UG CA S4-04

(DSE)

FOUR YEAR BACHELOR'S DEGREE COURSE

(CBCS in Accordance with NEP 2020)

SEMESTER - IV

OPEN SOURCE SOFTWARE

COMPUTER SCIENCE

Hiranya Chandra Bhuyan School of Science and Technology

KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Shikhar Kumar Sarma, Department of Information Technology, Gauhati University
Prof. Hiren Kumar Deva Sarma, Department of Information Technology, Gauhati University
Dr. Sanjib Kr. Kalita, Department of Computer Science, Gauhati University

Course Co-ordinator : **Dr.Tapashi Kashyap Das**, KKHSOU

SLM Preparation Team

Units	Authors
1, 2, 3, 4, 5, 6, 7	**Dr. Kishore Dutta**, Handique Girl's College and **Dr. Tapashi Kashyap Das,** KKHSOU
8, 9	**Mr. Anupam Gohain,** ADBU
10, 11	**Dr. Sruti Sruba Bharali,** KKHSOU
12, 13, 14	**Mr. Anupam Gohain**, ADBU and **Dr. Sruti Sruba Bharali,** KKHSOU

Editors

Content : **Dr. Sruti Sruba Bharali**, KKHSOU
Language : **Dr. Nava Khataniar**, Anundoram Borooah Academy Degree College

Structure & Format : Dr. Tapashi Kashyap Das, KKHSOU

April 2025

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State Open University is made available under a Creative Commons Attribution-Non Commercial-Share Alike 4.0 License (international) http://creativecommons.org/licenses/by-nc-sa/4.0/

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters: Patgaon, Rani Gate, Guwahati - 781 017
City Campus: NH 37, Resham Nagar, Khanapara, Guwahati - 781 022; www.kkhsou.in

DETAILED SYLLABUS

7 - 22 **UNIT 1: Installation of LaTeX**

Installation of LaTeX, Understanding Latex compilation.

23-42 **UNIT 2: Introduction to LaTeX**

Basic Syntax, Writing equations, Matrix, Tables.

43-68 **UNIT 3: Page Layout – I**

Page Layout – Titles, Abstract Chapters, Sections, References, Equation references, Citation, Table of contents.

69-83 **UNIT 4: Page Layout – II**

List making environments, Generating new commands, Figure handling, table & figure numbering, List of figures, List of tables, Generating index.

84-101 **UNIT 5: Packages**

Packages: Geometry, Hyperref, amsmath, amssymb, algorithms, algorithmic graphic, color, tikz listing.

102-130 **UNIT 6: Classes**

Classes: article, book, report, beamer, slides, letter.

131-171 **UNIT 7: Applications of LaTex**

Applications: Writing Resume, Writing question paper, Writing articles/ research papers, Presentation using beamer, inserting graphics, drawing graphics, writing complex equations.

172-180 **UNIT 8: INSTALLATION OF SCILAB**

Installation of Scilab (both windows & Linux)

UNIT 9: INTRODUCTION TO SCILAB

181-187 Introduction to Scilab, Basic syntax, Mathematical Operators, Predefined constants, Handling .sci files, Installation of additional packages e.g. optimization

188-198 **UNIT 10: FUNCTIONS IN SCILAB**

Built in functions, Complex numbers, Polynomials, Vectors, Matrix operations (functions like inv(), spec(), zeros(), ones(), eye(), rand(). Handling these data structures using built in functions.

199-213 **UNIT 11: CONDITIONAL STATEMENTS IN SCILAB**

Loops (for & while), Conditional statements

214-229 **UNIT 12: GRAPHICS IN SCILAB**

Graphics handling: 2D, 3D, Generating .jpg files, Function plotting, Data plotting, GUI in scillab, Plotting 2D graphs

230-240 **UNIT 13: APPLICATIONS OF SCILAB**

Applications: Numerical Linear Algebra (Solving linear equations, eigen values atc.) solving Ordinary Differential Equations, Numerical Analysis – iterative methods, Comparison with C/C++/Matlab

241-248 **UNIT 14: INTRODUCTION TO PYTHON**

Introduction to Python, Procedure to install Python, How to open Python console, Basic Python commands, Python programming.

COURSE INTRODUCTION

In the modern digital landscape, open source software has become a key force driving innovation and collaboration. With its free availability and flexibility for customization, it offers a valuable learning opportunity for students, educators, and professionals alike.

This course, titled **Open Source Software**, is designed as a self-learning material that introduces readers to three widely-used and powerful open-source tools—LaTeX, **Scilab,** and **Python.** These tools are commonly used in academics, research, and various industries. LaTeX is known for producing high-quality documents with scientific content; Scilab offers strong capabilities in numerical computations; and Python stands out as a simple yet powerful programming language used across many application areas.

The course is organized into fourteen easy-to-follow units. It begins with LaTeX, covering its installation, writing equations, document formatting, figure and table management, and its practical use in writing research papers, resumes, and presentations.

The second section introduces Scilab, explaining its installation, programming basics, use of functions, control structures, and graphical capabilities. You will also explore how Scilab helps solve problems in lineal algebra, differential equations, and numerical methods.

Finally, you will be introduced to **Python**, where you will learn how to install and run it, along with writing and executing basic programs.

By the end of this course, readers will gain hands-on experience with three versatile tools that are not only free to use but also vital for academic and professional growth in todays tech-driven world.

While going through a unit, you will notice some boxes, containing such information as would help you to understand some of the difficult, unfamiliar terms. At the end of each section, you will get “CHECK YOUR PROGRESS” questions. These have been designed to self-check your progress of study. It will be better if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with “ANSWERS FO CHECK YOUR PROGRESS” given at the end of each unit.

UNIT 1: INSTALLATION OF LATEX

UNIT STRUCTURE

1.1 Learning Objectives

1.2 Introduction

1.3 Installation of LaTeX

- 1.3.1 Installation of LaTeX in Windows
- 1.3.2 Installation of LaTeX in Linux
- 1.3.3 Installation of LaTeX Editor

1.4 Understanding LaTeX Compilation

1.5 Let Us Sum Up

1.6 Further Readings

1.7 Answers to Check Your Progress

1.8 Model Questions

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- perform installation of Latex software
- install the Latex packages in a computer
- start compilation of a simple Latex file

1.2 INTRODUCTION

The name LaTeX (pronounced as Lah-teck or Lay-teck) is derived from “Lamport TEX”. It is a system for typesetting documents, originally created by Leslie Lamport and is now maintained by a group of volunteers. LaTeX is based on TEX, a typesetting system designed by Donald Knuth in 1978 for high quality digital typesetting. TEX is a low-level language and since most people find it difficult to use, LaTeX has been developed to make it easier. The current version of LaTeX is LaTeX2e and it is widely used, particularly for complex and technical documents, such as those involving mathematics.

A LaTeX document can be typed in a simple text editor such as

Notepad. However, it is easier to use a dedicated LaTeX editor. While typing, the document structure can be marked with tags such as title, subtitle, chapters, sections, subsections etc. The file needs to be saved as a **`.tex`** file extension. For instance, when the file is saved as test.tex, after compilation it converts into another format. Thus it is obvious that, in order to create and compile a LaTeX file, a TEX editor and a complier need to be installed. In this unit, we shall describe what the basic requirements are and how to install the required libraries or packages.

1.3 INSTALLATION OF LATEX

1.3.1 Installation of Latex in Windows

In a computer with Windows operating system, the corresponding version of LaTeX is MiKTeX. If the computer has internet connection, then by typing www.miktex.org in the Internet browser, the MiKTeX official page can be seen which appear as shown in Figure 1.1. By clicking the **Downloads** button, one can see that different MiKTeX packages are available for different operating systems. By selecting the operating system that the computer has, one can choose and download the corresponding installer. MiKTeX has the ability to install missing packages automatically, i.e., this installer is suitable for computers connected to the Internet.

Figure 1.1: The MiKTeX webpage

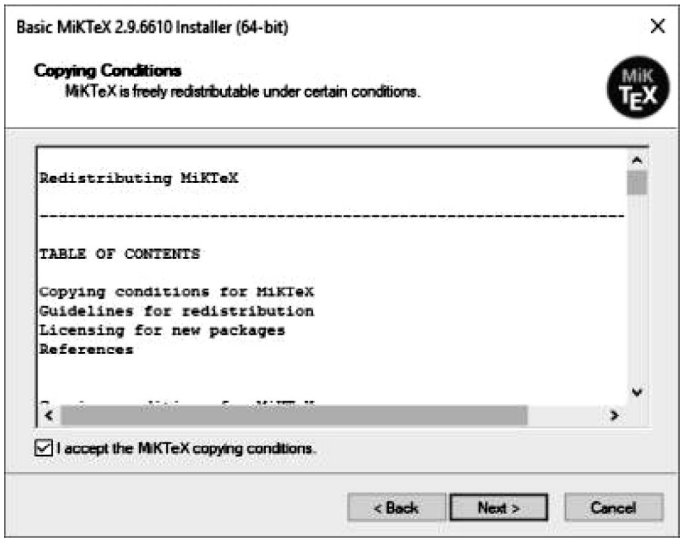


MiKTeX for Windows requires one of the following Windows operating systems:

- Windows 10
- Windows 8
- Windows 7
- Windows Server 2012
- Windows Server 2008
- Windows Server 2008 with Service Pack 2

The "Basic MiKTeX Installer" is used to set up a standard TeX/LaTeX system. The installer is available on the download page. A double-click on the downloaded file opens a wizard showing the MiKTeX copying conditions as shown in Fig. 1.2.

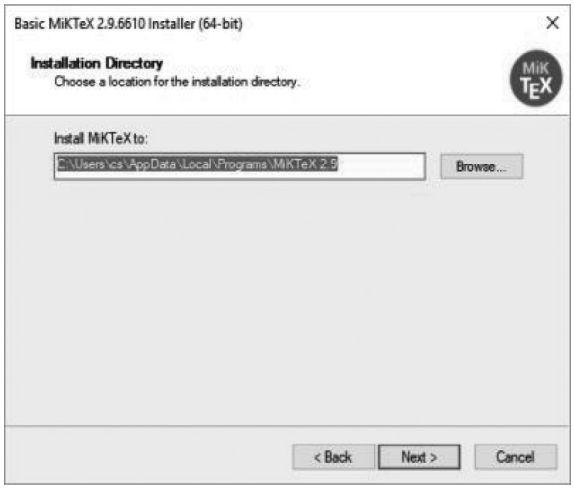
Figure 1.2: Copying condition for Basic MiKTeX installer



By clicking on the box showing **"I accept the MiKTeX copying conditions"** and then by clicking the **Next**, one can proceed for the next page. The next wizard is the installation scope wizard as shown in Figure 1.3. Two options appear here for installation scope:

- Set up a private (per-user) TeX installation.
- Set up a shared (system-wide) TeX installation.

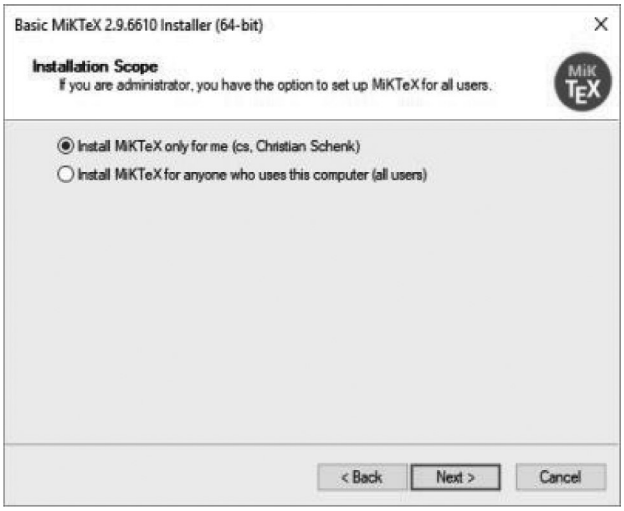
Figure 1.3: Asking for Installation scopes



Setting up a shared (system-wide) installation requires administrator privileges and we should keep in mind that maintaining a shared installation can become complicated and error prone. It is therefore highly recommended that we choose a private installation. After clicking on the option as shown in the wizard (Figure 1.3) and by clicking Next button, one can proceed for the next page.

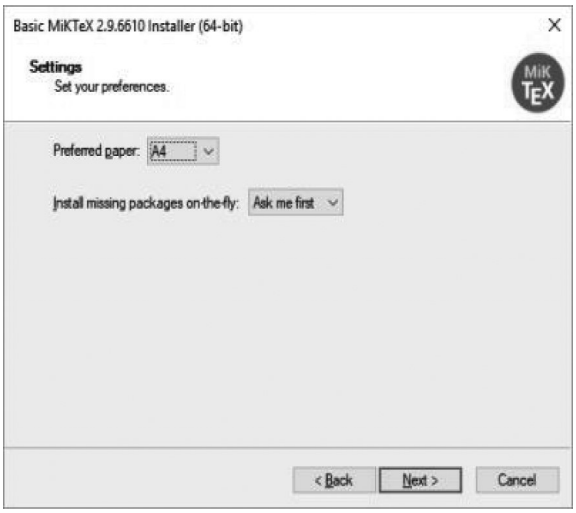
In the next wizard as shown in Figure 1.4, there appears an option for choosing the location in the computer where the MiKTeX can be installed. By clicking Browse button, it is possible to choose any location of the hard disks available in the computer. Once this is done, clicking the Next button would lead to the next page.

Figure 1.4: Asking for Installation directory



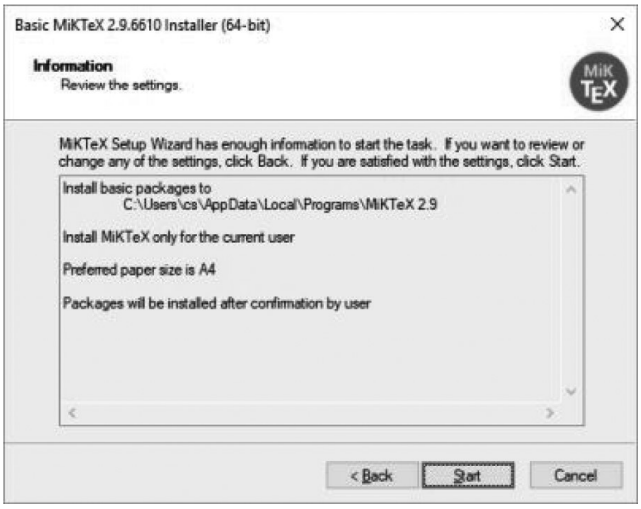
The next wizard appears as shown in Figure 1.5 which allows setting the preferred paper size. It also has an option to change the default behavior of the integrated package installer. We can select **"Ask me first**," if we want to confirm installation of required packages. Alternatively, we can select **Always**, if required packages are to be installed without confirmation and select **Never**, if we disallow the installation of missing packages.

Figure 1.5: Setting preferences



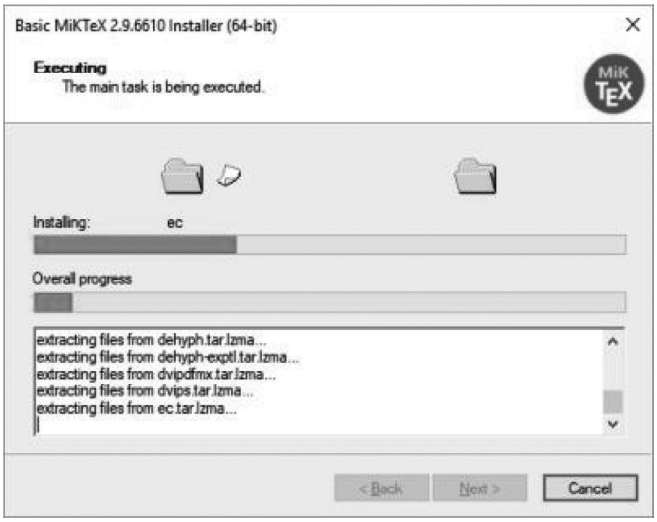
All these preferences can be changed later. By clicking Next button, we can proceed for next wizard page. Before the actual installation process begins, it gives us a chance to review our decisions as shown in Figure 1.6 below.

Figure 1.6: Asking for review or change any of the settings



If we are satisfied with the settings, then we can initiate the actual installation process by clicking **Start** button (Fig. 1.6). The installation will take a few minutes and the progress bar shows an approximate percentage of completion. At this stage, we can cancel the installation process by clicking the **Cancel** button as shown in Fig. 1.7.

Figure 1.7: The actual installation begins



When the installation is finished, we can click **Next** to open the last wizard page.

MiKTex is now installed in the computer and by clicking **Close** button as shown in Fig. 1.8, we can exit the setup wizard.

Figure1.8: Completing the MiKTeX setup wizard





CHECK YOUR PROGRESS

Q 1: Write down the name of the package for LaTeX which is compatible in a computer with windows operating system.

.....

.....

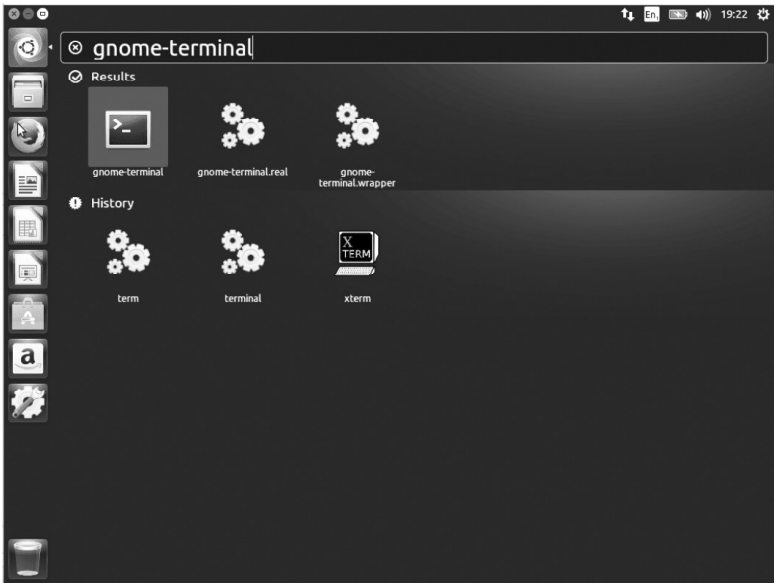
Q 2: Write down the source of MiKTeX from which the installer can be downloaded.

.....

.....

1.3.2 Installation of Latex in Linux

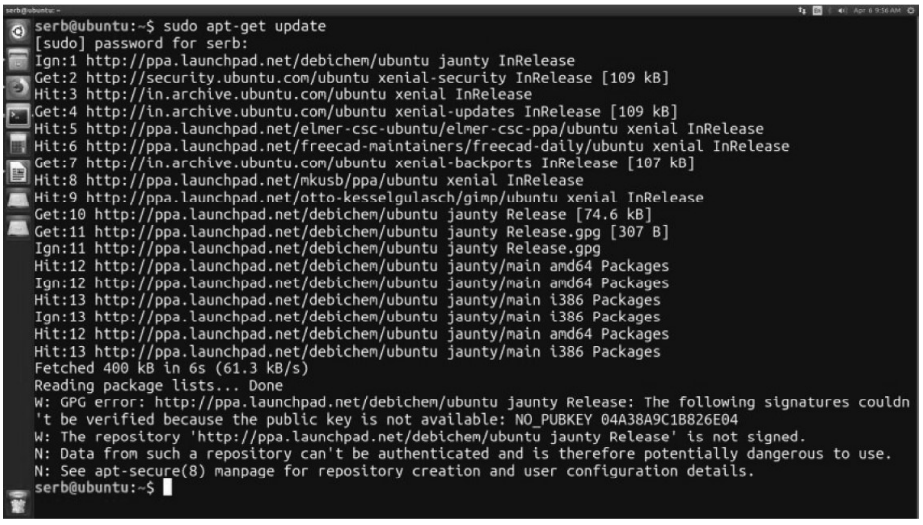
Installation of LaTeX is quite easier in computers with Linux operating system such as Ubuntu. It is because GNU/Linux operating system is almost always linked with **commands** and command line. If our computers have Internet connection available, software can be easily installed using command line. Here we shall describe how to install LaTeX using command line provided the computer has Internet connection.



First of all, we need to open a Terminal. There are more than one way how to open a Terminal in Linux. The simplest one is to Press Ctrl+Alt+T on the keyboard. This will open a Terminal. Alternatively, we can search for it by pressing “Windows” key and then typing “terminal”. It is important to keep in mind that commands in Linux are case sensitive (so upper- or lower-case letters matter). Another alternative way is to search in the Ubuntu Dash. We can start typing the word “term” or “genome-terminal” into the search box as shown in Fig. 1.9. As we type, there appears the terminal icon and we can open any one of them.

In the terminal appearing as shown below in Fig. 1.10, we need to type the command “sudo apt-get update” and then press Enter. This is an **update command** and it will update all the package information from all configured sources. It is useful to get information on an updated version of packages or their dependencies.

Figure 1.10: Running the update command in terminal

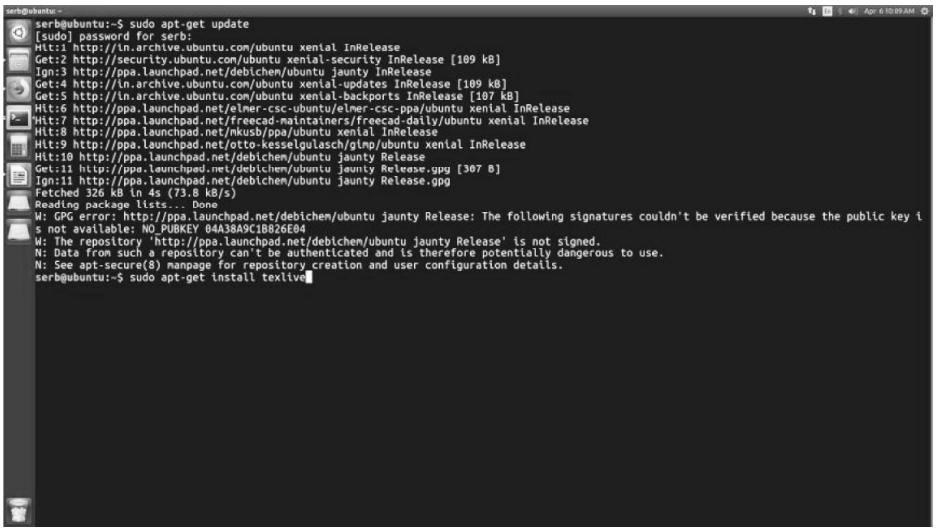


After update is completed, we can install LaTeX by typing the command

```
$ sudo apt-get install texlive-latex-extra
```

and pressing Enter (Fig. 1.11). It will start installing LaTeX in Ubuntu.

Figure 1.11: Showing the command for LaTeX installation





CHECK YOUR PROGRESS

Q 3: Before installation of LaTeX using command prompt in a Linux operating system, what is the essential requirement and how can we do it?

.....

.....

Q 4: What is the command for installing LaTeX in a machine with Linux operating system?

.....

.....

1.3.3 Installation of Latex Editors

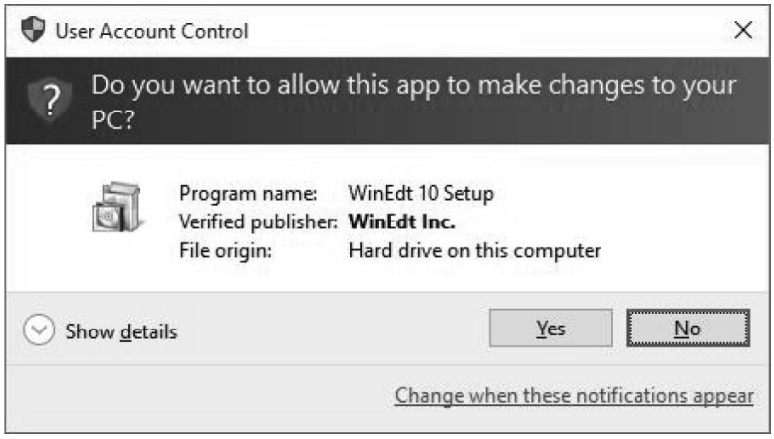
There are many different types of LaTeX editors. Some of them are WinEdt, TeXmaker, gedit, Emacs, LyX, Kile, etc.

- **Installing WinEdt**

  WinEdt is a powerful all-purpose text editor for Windows and can be used to create and compile LaTeX documents. The latest version WinEdt 10 runs on Windows 10, 8 and 7 (32-bit or 64-bit). If we have an older version of Windows, it is better to consider a non-unicode WinEdt 6 or even an older WinEdt 5.5. To install WinEdt in a Windows machine,

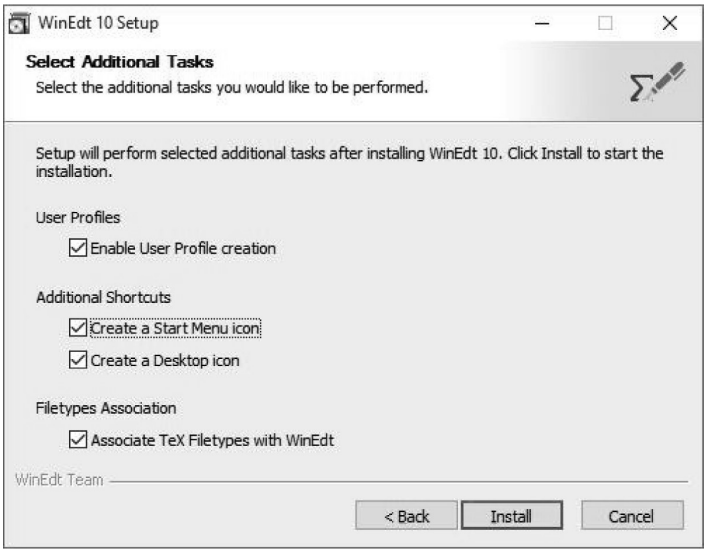
we need to download the installer from http://www.winedt.com/download.html and then execute the setup file winedt103-64.exe (or winedt103-32.exe if you choose to install a 32-bit version). As we execute the setup file, a User Account Control warning will be displayed as shown below (Fig. 1.12).

Figure 1.12: Executing WinEdt 10 setup file



We select “Yes” and it is now safe to proceed with the installation (Fig. 1.13). The Installer Wizard will guide us through the process. For most users the default options in the Wizard should be used. However, if we are creating a portable installation, we should disable User Profile Creation. This way WinEdt will create a Local folder inside its install folder and this folder will contain all our custom settings.

Figure 1.13: Installing WinEdt 10

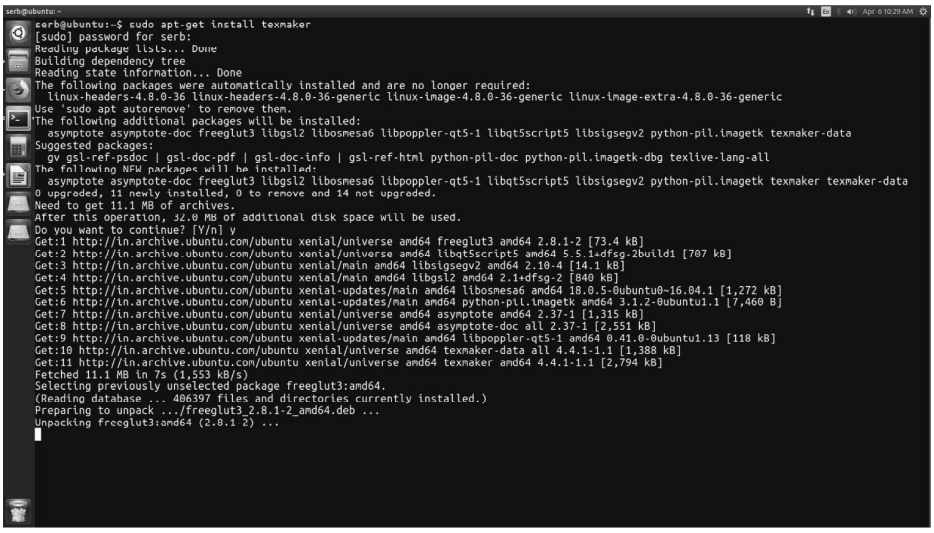


➢ Installing other editors in Linux

TeXmaker is one of the most user-friendly LaTeX editors that supports all the major platforms including Windows, Linux, and MacOS. It provides syntax highlighting, auto code completion, code folding, spell checking, etc. It can be installed in Linux operating system by typing the following command in terminal one by one (Fig. 1.14).

```
$sudo apt-get install update
$ sudo apt-get install texmaker
```

Figure 1.14: Running the command`$ sudo apt-get install texmaker`



Another most commonly used editor is **gedit** which is the default text editor of the GNOME desktop environment and designed as a general-purpose text editor. It can be installed in Linux operating system by typing the following commands.

```
$ sudo apt-get update
$ sudo apt-get install gedit
```



CHECK YOUR PROGRESS

Q 5: Write down the name of some common LaTeX editors.

.....

.....

Q 6: How will you install gedit in linux using command prompt?

.....

.....

1.4 UNDERSTANDING LATEX COMPILATION

LaTeX documents are plain documents with a .tex extension. Suppose you had saved your document as “mylatexfile.tex” and you want to compile it. There are three output formats available in all TeX distributions. These are DVI, PS, and PDF formats. To generate a specific output, the document has to be compiled by running a command in a terminal (Linux and macOS) or the command prompt (windows). We can use the command:

```
$ latex mylatexfile.tex
```

This will compile the file. If there are errors, we can quit the compiler by pressing “Ctrl+ X” and then Enter key. Once the errors are corrected in the file mylatexfile.tex, it successfully compiles, we will get a file named **mylatexfile.dvi.** It can further be converted either into a Postscript (PS) or PDF file, which may be done by using the commands

```
$ dvips mylatexfile.dvi
$ dvipdf mylatexfile.dvi.
```

Alternatively, the command

```
$ pdflatex mylatexfile.tex
```

will directly generate “**mylatexfile.pdf**”. For more clarification, an example is shown below. Figure 1.15 displays a simple latex file written in **gedit** text editor and saved as “mylatexfile.tex.” In order to create a PDF file out of it, we need to open the terminal and run the command as shown in Fig. 1.16.

Figure 1.15: A Simple LaTeX file written in gedit

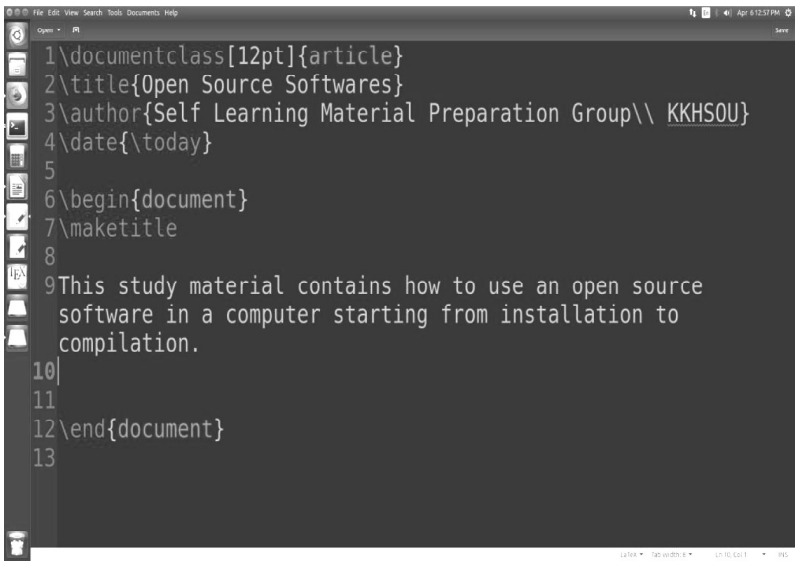
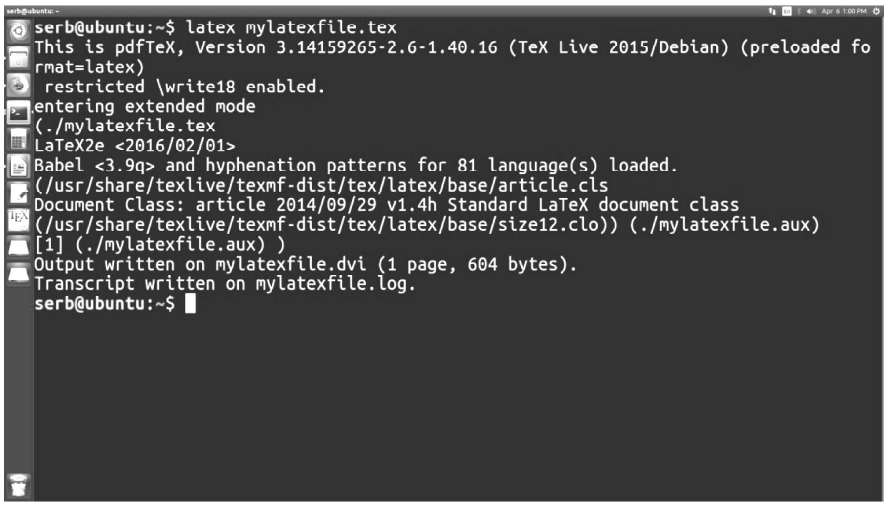
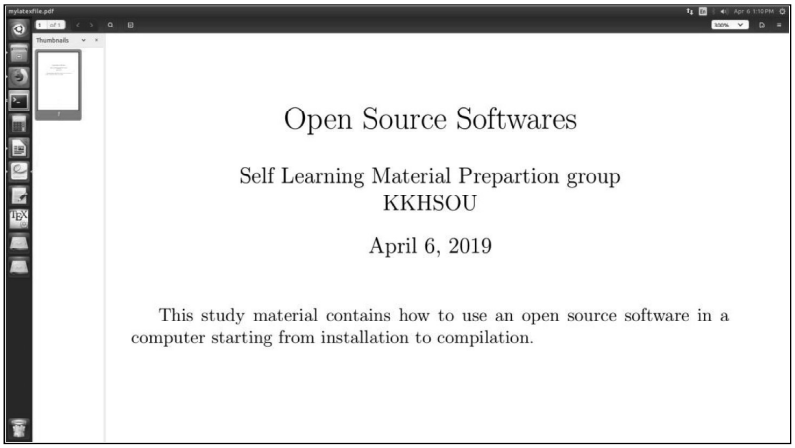


Figure 1.16: Compiling the LaTeX file



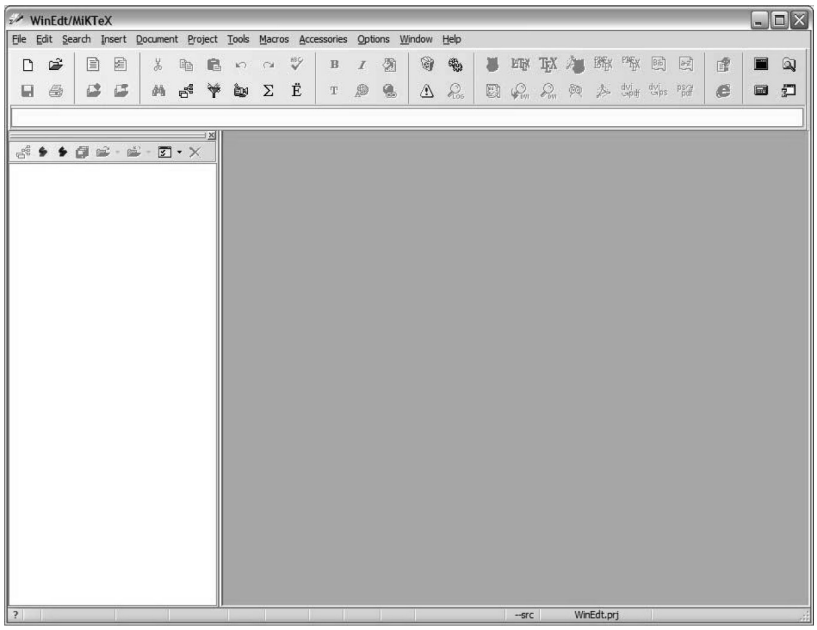
While compiling by the command **`$ latex mylatexfile.tex`**, it shows towards the last "Output written mylatexfile.dvi (1 page, 604 bytes)" and "Transcript written on mylatexfile.log". In the folder containing the file, we will see these files together with a mylatexfile.pdf file. If we open and see this pdf file, it appears as shown below (Fig. 1.17).

Figure 1.17: The output pdf file of the input file shown in Fig.



While compiling in WinEdt, the WinEdt screen appears as shown in Fig. 1.18. We can construct a file like **mylatexfile.tex** and run it. It will generate a similar pdf file as above.

Figure 1.18: Pdf file generated in WinEdt





CHECK YOUR PROGRESS

Q 7: What is the command for compiling a LaTeX file using command prompot?

.....

.....

Q 8: How can we create a pdf file out of **`test.tex`** file?

.....

.....



1.5 LET US SUM UP

- Latex is basically a system for typesetting documents.
- A LaTeX user writes an input file containing text and interspersed commands.
- The text file has an extension `.tex` which, after compilation, generates the desired output files such as pdf, ps, or dvi.
- LaTeX as open software, has different version depending on the operating system of the computer.
- The most frequently used versions, namely, MiKTeX for windows and Texlive for Linux, are available for installation.



1.6 FURTHER READING

1) LaTeX Project, http://www.latex-project.org/,

2) LaTeX Wikibook, http://en.wikibooks.org/wiki/LaTeX/



1.7 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: The LaTeX package which is compatible in a windows operating system is MiKTeX.

Ans to Q No 2: MiKTeX is freely available in the website www.miktex.org.

Ans to Q No 3: Before installation of LaTeX, the essential requirement is to update all the package information from all configured sources. This can be done by typing the command **`$ sudo apt-get update`** and then press Enter.

Ans to Q No 4: The command for installing LaTeX in a machine with

linux operating system is **$ sudo apt-get install texlive**

Ans to Q No 5: Some commonly used LaTeX editor are WinEdt, TeXmaker, gedit, Emacs, LyX, Kile, etc.

Ans to Q No 6: The following commands need to be run one by one in order to install gedit.

```
$ sudo apt-get update
$ sudo apt-get install gedit
```

Ans to Q No 7: The command for compiling a `.tex` file, for instance, mylatex.tex file is **$latex mylatex.tex**

Ans to Q No 8: The command that is used to create a pdf file from mylatex.tex file is **$pdflatex mylatex**



1.8 MODEL QUESTIONS

Q 1: What is LaTeX? What are the basic requirements to install and compile a LaTeX file?

Q 2: Write down the name of the most frequently used version of LaTeX suitable for a windows operating system.

Q 3: Write down the name of some commonly used LaTeX editors.

Q 4: What can we do in an editor? How can we compile the file generated using an editor?

Q 5: In a Windows machine, how can we generate a pdf file out of a LaTeX file?

Q 6: How can we open a command prompt in a computer working with Linux operating system?

Q 7: What is the command for compiling a LaTeX file?

Q 8: Write down the commands for generating a pdf file from a LaTeX input file.

Q 9: What are the possible file formats that can be generated from a LaTeX file?

Q 10: What is gedit? How does it differ from winedt?

*** ***** ***

UNIT 2 : INTRODUCTION TO LATEX

UNIT STRUCTURE

2.1 Learning Objectives

2.2 Introduction

2.3 Basic Syntax

- 2.3.1 Basic Structure
- 2.3.2 Font Sizes
- 2.3.3 Font Styles

2.4 Writing Mathematical Expressions

2.5 Writing Equations

2.6 Matrices

2.7 Tables

2.8 Let Us Sum Up

2.9 Further Readings

2.10 Answers to Check Your Progress

2.11 Model Questions

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- prepare a document in LaTeX format
- describe the structure of a LaTeX document
- use different font styles and font sizes in writing a LaTeX document
- write mathematical expressions and use various frequently used mathematical symbols
- write mathematical equations in LaTeX
- generate matrices and tables of different format

2.2 INTRODUCTION

As discussed in Unit 1, in LaTeX document processing we create a **`.tex`** file and LaTeX macro reads this to produce the final document.

It has some disadvantages in comparison to Microsoft Word. For instance, it can't display the final result directly and it requires the knowledge of specific commands for LaTeX markup. However, it has been widely used because it is easier to type any mathematical expressions or equations and also easy to generate the table of contents, list of figures, list of tables, references, indices etc.

In this unit we shall describe the basic syntax used in LaTeX, how to generate various mathematical symbols and operators, how to write mathematical expressions and equations, and how to construct matrices and tables. In the next unit, we will explore the page layout structure in LaTeX.

2.3 BASIC SYNTAX

2.3.1 Basic Structure

The basic structure of creating a text file consists of a number of command lines.

Let us begin with creating a simple article. The first line of the document is–

```
\documentclass {article}
```

This informs LaTeX what kind of document will be compiled. There are other document classes which are introduced in detail in **unit 6**. The second line of the document is

```
\begin {document}
```

After writing this line, one can type the text that will appear in the output files.

Once, the writing is completed, the document needs to be closed by the command

```
\end {document}
```

An example is shown in Fig. 2.1.

Figure 2.1(a): A simple .tex file in gedit showing the structure of a LaTeX document

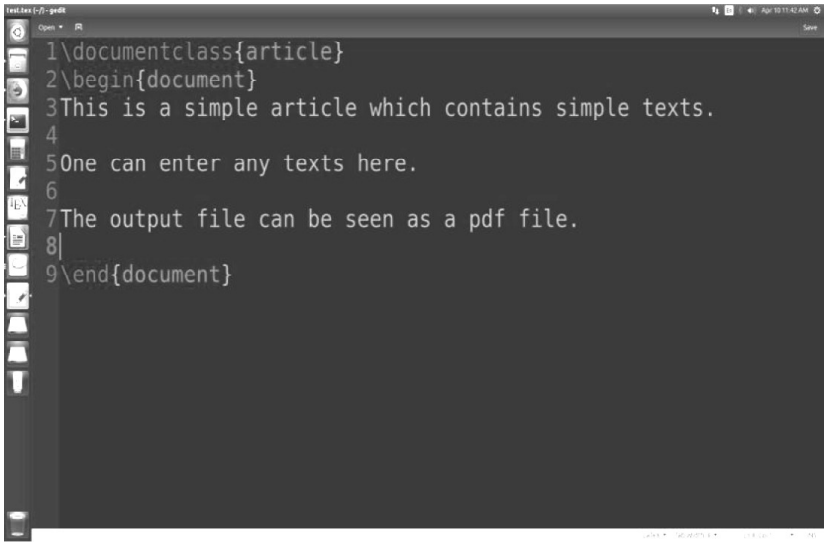
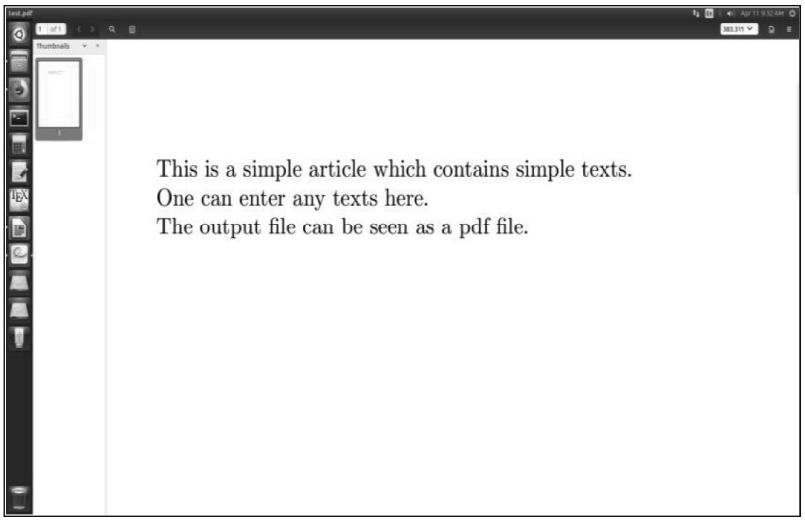


Figure 2.1(b): The corresponding output pdf file of Fig. 2.1(a)



2.3.2 Font Sizes

The default font size of the above texts is 10pt. One can define the font size in the first line as

```
\documentclass[11pt]{article}
```

This will change the font size to 11pt. In addition, there are standard type size commands supported by LaTeX. In Table 2.1, we display some of the commonly used commands. It is also shown how the texts appeared when written using these commands.

Table 2.1: Some useful commands for changing font sizes

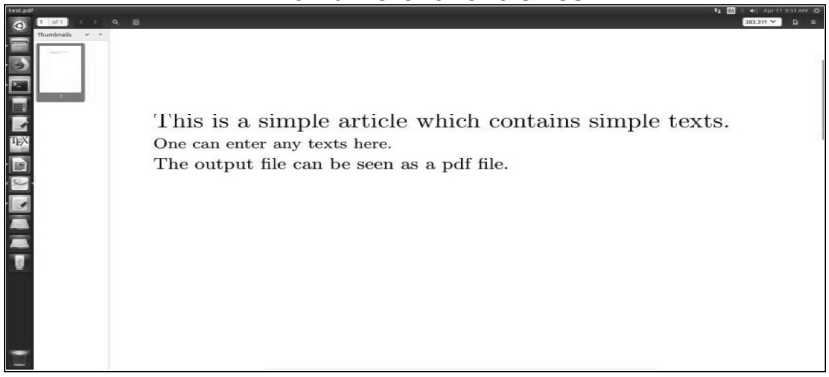
Command	Font size
\tiny	Sample text
\scriptsize	Sample text
\footnotesize	Sample text
\small	Sample text
\normalsize	Sample text
\large	Sample text
\Large	Sample text
\LARGE	Sample text
\huge	Sample text
\Huge	Sample text

A simple example is shown in Fig. 2.2 where different font sizes are used in each line. We use **\large** in the first line, **\footnotesize** in the second line and **\small** in the third line of the text.

Figure 2.2 (a): A sample .tex file with font size commands

```
\documentclass{article}
\begin{document}
{\large This is a simple article which
contains simple texts.}
{\footnotesize One can enter any texts
here.}
{\small The output file can be seen as
a pdf file.}
\end{document}
```

Figure 2.2(b): The output file of Fig. 2.2(a) showing each line with different font sizes



2.3.3 Font Styles

For font styles, there exists a number of style commands which can be used in LaTeX. In order to add some emphasis on a word or a phrase, **\emph{text}** command can be used which italicizes the text. Italics may be specified explicitly with **\textit{text}**. Alternatively one can use equivalent switches **{\em . .}** or {**\it . .**} respectively.

Table 2.2 (a): Examples for changing font styles

How to write	How it appears
I want to \emph{emphasize} a word	**I want to *emphasize* a word.**
I want to {\em emphasize} a word	
\emph{I want to emphasize a sentence, where a \emph {word} appears upright.}	***I want to emphasize a sentence, where a* word *appears upright.***
{\em I want to emphasize a sentence, where a {\em word} appears upright.}	

The **\emph** command is dynamic, i.e., if we emphasize a word which is already in an emphasized sentence, it will be reverted to the upright font as shown in the table.

Sometimes texts are emphasized more heavily by the use of boldface instead of italics. Bold texts can be generated with the **\textbf{}** command as shown below.

How to write	How it appears
I use \textbf **{Bold text}** to heavily emphasize a very important word.	I use **Bold text** to heavily emphasize a very word.

The following table lists the commands typically used to change the font shapes.

Table 2.2(b): Basic font styles

LaTeX Command	Equivalent switch	Output style	Remarks
`\textnormal{}`	`{\normalfont..}`	Document	This is the default or normal font
`\emph{}`	`{\em ..}`	*emphasis*	Typically italics
`\textrm{}`	`{\rmfamily ..}`	Roman font family	
`\textsf{}`	`{\sffamily ..}`	Sans serif font family	
`\texttt{}`	`{\ttfamily ..}`	Teletype font family	This is a fixed width or monospace font.
`\textit{}`	`{\itshape ..}`	Italic shape	
`\textsl{}`	`{\slshape ..}`	*Slanted shape*	Similar to but slightly different from italics
`\textbf{}`	`{\bfseries ..}`	**Bold**	
`\textsc{}`	`{\scshape ..}`	SMALL CAPITALS	

➢ **Creation of paragraphs, newlines, comments**

- LaTeX wraps text in adjacent lines as if they were part of the same paragraph. To start a new paragraph, insert an extra “return”.
- To get a newline without starting a new paragraph, use \\.
- To get a comment, use the percent sign **%** at the beginning of a line. The rest of that particular line will be commented out.



CHECK YOUR PROGRESS

Q 1: Write down the basic structure of the simplest LaTeX file.

.....

.....

Q 2: What are the different types of font size commands in LaTeX?

.....

.....

Q 3: What is the difference between the output font styles while using {\em software} and \emph{software}?

.....

.....

Q 4: In LaTeX how do we make the following sentence in small capital font style? The Earth revolves around the Sun.

.....

.....

2.4 WRITING MATHEMATICAL EXPRESSIONS

In order to write a mathematical expression or an equation in the middle of text, math mode can be used. The math mode is just the dollar sign `$`. Any mathematical expression can be written within `$…..$`. For instance `$E=Mc^2$` produces $E = Mc^2$.

Everything between the two `$` signs will be considered as math formula. Subscript and superscript are done using _ and ^ characters. If we want multiple characters in the superscript or subscript, we have to write them within curly braces. For example, `$e^i\pi=-1$` gives $e^{i\pi}=-1$ and `$e^{i\pi}=-1$` gives $e^{i\Pi}=-1$. Fractions are done using `$\frac{1}{2}$` which appears as $\frac{1}{2}$. Table 2.3 gives a list of mathematical commands used in LaTeX.

Table 2.3: Mathematical symbols

A. Greek letters

\alpha	α	\delta	δ	\lambda	λ	\rho	ρ
\beta	β	\eta	η	\mu	μ	\sigma	σ
\gamma	γ	\theta	θ	\pi	π	\Sigma	Σ
\Gamma	Γ	\psi	ψ	\omega	ω	\nu	ν

\Phi	Φ	\Psi	Ψ	\Omega	Ω	\epsilon	ϵ
\chi	χ	\xi	ξ	\zeta	ς	\tau	τ
\kappa	κ	\Xi	Ξ	\phi	ϕ	\varepsilon	ε

B. Binary operation and relation symbols: Some symbols require amssymb package

\times	$\times$	\gg	$\gg$	\leq	$\leq$	\forall	$\forall$
\div	$\div$	\geq	$\geq$	\ni	$\ni$	\in	$\in$
\pm	$\pm$	\nless	$\not<$	\subset	$\subset$	\exists	$\exists$
\mp	$\mp$	\ngtr	$\not>$	\supset	$\supset$	\approx	$\approx$
\neq	$\neq$	\ngeq	$\not\geq$	\cup	$\cup$	\sim	$\sim$
\star	$\star$	\nleq	$\not\leq$	\cap	$\cap$	\equiv	$\equiv$
\circ	$\circ$	\perp	$\perp$	\subseteq	$\subseteq$	\propto	$\propto$
\ll	$\ll$	\paral lel	$\parallel$	\supseteq	$\supseteq$	\infty	$\propto$

C. Variable-sized math symbols

\sum	$\sum$	\iint	$\iint$	\bigwedge	$\bigwedge$	\bigoplus	$\bigoplus$
\int	$\int$	\iiint	$\iiint$	\bigvee	$\bigvee$	\biguplus	$\biguplus$
\prod	$\prod$	\bigcap	$\bigcap$	\bigodot	$\bigodot$	\bigsqcup	$\bigsqcup$
\oint	$\oint$	\bigcup	$\bigcup$	\bigotimes	$\bigotimes$	\coprod	$\coprod$

D. Arrow symbols: Some symbols require amssymb package

\leftarrow	$\leftarrow$	\Longleftarrow	$\Longleftarrow$	\leftleftarrows	$\leftleftarrows$
\Leftarrow	$\Leftarrow$	\longrightarrow	$\longrightarrow$	\curvearrowleft	$\curvearrowleft$
\rightarrow	$\rightarrow$	\Longrightarrow	$\Longrightarrow$	\curvearrowright	$\curvearrowright$
\Rightarrow	$\Rightarrow$	\longleftrightarrow	$\longleftrightarrow$	\circlearrowleft	$\circlearrowleft$
\leftrightarrow	$\leftrightarrow$	\Longleftrightarrow	$\Longleftrightarrow$	\circlearrowright	$\circlearrowright$
\Leftrightarrow		\nearrow	$\nearrow$	\twoheadleftarrow	$\twoheadleftarrow$
\uparrow	$\uparrow$	\nwarrow	$\nwarrow$	\twoheadrightarrow	$\twoheadrightarrow$
\Uparrow	$\Uparrow$	\searrow	$\searrow$	\dagger	$\dagger$
\downarrow	$\downarrow$	\swarrow	$\swarrow$	\ddagger	$\ddagger$
\Downarrow	$\Downarrow$	\upuparrows	$\upuparrows$	\looparrowleft	$\looparrowleft$
\updownarrow	$\updownarrow$	\downdownarrows	$\downdownarrows$	\looparrowright	$\looparrowright$
\Updownarrow	$\Updownarrow$	\rightleftarrows	$\rightleftarrows$	\leftarrowtail	$\leftarrowtail$
\longleftarrow	$\longleftarrow$	\rightrightarrows	$\rightrightarrows$	\rightarrowtail	$\rightarrowtail$

E. Math accents

\hat{a}	$\hat{a}$	\acute{a}	$\acute{a}$	\vec{a}	$\vec{a}$
\dot{a}	$\dot{a}$	\ddot{a}	$\ddot{a}$	\tilde{a}	$\tilde{a}$

F. Extra text symbols

\copyright	©	\S	§	\pounds	

G. Special text characters

\#	#	\&	&	\$	$	@	@	\%	%

H. Miscellaneous symbols: Some symbols require amssymb package

\vdots	$\vdots$	\Delta	Δ	\Re	$\Re$
\cdots	$\cdots$	\nabla	∇	\Im	$\Im$
\sqrt	$\sqrt{}$	\angle	$\angle$	\ell	ℓ
\ddots	$\ddots$	\measuredangle	$\measuredangle$	\emptyset	$\emptyset$
\partial	∂	\sphericalangle	$\sphericalangle$	\nexists	$\nexists$
\hbar	$\hbar$	\hslash	$\hbar$	\surd	$\surd$

I. Trigonometric functions

\sin x	$\sin x$	\cos x	$\cos x$	\tan x	$\tan x$
\sec x	$\sec x$	\cosec x	$\operatorname{cosec} x$	\cot x	$\cot x$



CHECK YOUR PROGRESS

Q 5: Write down the expression in LaTeX to produce the following:

(i) $\angle 180° = \pi$

(ii) $\sqrt[5]{y^3}$

(iii) $A \cup (B \cap C) = (A \cap B) \cap (A \cup C)$

(iv) Golden Ratio $\approx 1.618.$

2.5 WRITING EQUATIONS

In order to type a mathematical expression on its own line and centered, double dollar `$$` signs are used. For example, `$$E=Mc^2$$` produces

$$E=Mc^2$$

To give an equation a number, equation environment can be used

as

```
\begin{equation}
   E=Mc^2
   \sin^2\theta+\cos^2\theta=1
\end{equation}
```

which will produce an equation number as

$E=Mc^2$ (2.1)

$\sin^2\theta+\cos^2\theta=1$ (2.2)

In Table 2.4, we shall show how to produce some of the common mathematical expressions. In the left-hand column of the table, we provide the LaTeX version of the expressions that appear in the corresponding right-hand column. As already mentioned above, any mathematical expressions need to be written in between two `$` signs or in between `\begin{equation}` and `\end{equation}`. Thus, while practicing in a LaTeX editor, the expressions appear in the left-hand column should be expressed within a mathematical environment.

Table 2.4: Some common mathematical expressions

How to write	How it appears
`\frac{1}{x^2}`	$\frac{1}{x^2}$
`\sqrt{\frac{a}{b}}`	$\sqrt{\frac{a}{b}}$
`\sqrt[n]{x}`	$\sqrt[n]{x}$
`\ln{x}`	$\ln x$
`e^x=\exp{x}`	$e^x = \exp x$
`\sqrt[n]{1+x+x^2+x^3+\dots+x^n}`	$\sqrt[n]{1+x+x^2+x^3+\cdots+x^n}$
`\sin x =x- \frac{x^3}{3!}+ \frac{x^5}{5!}+\cdots`	$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots$
`\frac{d^2y}{dx^2}-6\frac{dy}{dx}+7x=0`	$\frac{d^2y}{dx^2} - 6\frac{dy}{dx} + 7x = 0$
`ax^2+bx+c=0`	$ax^2 + bx + c = 0$
`\sum_{i=1}^n x_i^2`	$\sum_{i=1}^{n} x_i^2$

`\prod_{i=1}^n x_i^2`	$\prod_{i=1}^{n} x_i^2$
`\frac{n!}{k!(n-k)!} = \binom{n}{k}`	$\frac{n!}{k!\,(n-k)!} = \binom{n}{k}$
`\lim\limits_{x\to\infty}\exp(-x)=0`	$\lim_{x\to\infty} \exp(-x) = 0$
`\lim_{x\rightarrow\infty} \exp(-x)=0`	$\lim_{x\to\infty} \exp(-x) = 0$
`f(x)=\int (3x^2+6x+1) dx`	$f(x) = \int (3x^2 + 6x + 1)dx$
`f(x)=\int_0^\infty x^5 e^{-x} dx`	$f(x) = \int_0^\infty x^5 e^{-x} dx$
`x=\frac{-b\pm\sqrt{b^2 -4ac}}{2a}`	$x = \frac{-b \pm \sqrt{b2 - 4ac}}{2a}$
`for all $\epsilon > 0$ for all $x \in D$`	for all $\epsilon > 0$ and and for all $x \in D$

Let us look at some examples showing how to write a mathematical equation in LaTeX.

Example 2.1 Type $cos2\theta = cos^2\theta - sin^2\theta$

$$= 2cos^2\theta - 1$$

Solution:

```
\begin{eqnarray*}
\cos 2\theta &=& \cos^2 \theta - \sin^2 \theta \\
             &=& 2 \cos^2 \theta - 1.
\end{eqnarray*}
```

The eqnarray* environment produces equations that are aligned and with no equation numbers. Equation number will be generated when we write only eqnarray instead of eqnarray*.

Example 2.2: Type

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

Solution:

```
\[\frac{\partial u}{\partial t}
= \frac{\partial^2 u}{\partial x^2}
```

```
        + \frac{\partial^2 u}{\partial y^2}
        + \frac{\partial^2 u}{\partial z^2} \]
```

Example 2.3: Type

$$\sum_{k=1}^{n} k^2 = \frac{1}{2} n(n+1).$$

Solution:

```
\[ \sum_{k=1}^n k^2 = \frac{1}{2} n (n+1).\]
```

Example 2.4: Type

$$i\hbar \frac{\partial \psi}{\partial t} = \frac{-\hbar^2}{2m} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \psi + V\psi.$$

Solution:

```
\[ i\hbar\frac{\partial \psi}{\partial t}
  = \frac{-\hbar^2}{2m} \left(
        \frac{\partial^2}{\partial x^2}
        + \frac{\partial^2}{\partial y^2}
        + \frac{\partial^2}{\partial z^2}
      \right) \psi + V \psi.\]
```

In this example, we use the `\left` and `\right` commands. These commands must come in matching pairs. Here, `\left( ……\right )` have the matching delimiters that produces the parenthesis. However, the matching delimiters need not be the same. An invisible delimiter, for instance, can be created by entering a dot ('.') after the `\left` and `\right` command. The following example 2.5 illustrates this.

Example 2.5: Type

$$|x| = \begin{cases} x & \text{if } x \geq 0; \\ -x & \text{if } x < 0 \end{cases}$$

Solution:

It can be produced using array environment as follows.

```
\[ |x| = \left\{ \begin{array}{ll}
x & \textnormal{if $x\geq0$}, \\
x & \textnormal{if $x<0$}
\end{array} \right. \]
```

In fact, the above example can also be produced using the `cases` environment as

follows.

```
\[  |x|  =  \begin{cases}
x  &  \mbox{if  }  x  \geq  0;\\
-x  &  \mbox{if  }  x  <  0.\end{cases}\]
```

Example 2.6: Type

$$f(x) = \int \frac{sinx}{x} dx \qquad (2.1)$$

Solution:

```
\begin{equation}
      f(x)  =\int\frac{\sin  x}{x}  dx
\end{equation}
```

Example 2.7: Type

$$x^2 + y^2 = R^2$$
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Solution:

```
\begin{align}
      x^{2}&+  y^{2}=R^{2}\\
      \frac{x^2}{a^2}&+\frac{y^2}{b^2}=  1\\
      \frac{x^2}{a^2}&+\frac{y^2}{b^2}=  1
\end{align}
```

Here, the alignment is done by using the `\begin{align}` ……`\end{align}` environment. It can also be done using `\begin{eqnarray}`….`\end{eqnarray}` environment as shown in the first example.



CHECK YOUR PROGRESS

Q 6: Write down the following expressions in LaTeX.

(i) $|\delta(t)| \approx e^{\lambda t}|\delta Z_0|$ (ii) $F = \frac{Gm_1m_2}{r^2}$

(iii) $\nabla.E = \frac{\rho}{\epsilon_0}$ (iv) $\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = rV - rS\frac{\partial V}{\partial S.}$

2.6 MATRICES

There exists several environments for matrices within the **amsmath** package. In order to access these environments, we need to add in th preamble a line as \usepackage{amsmath}. As we shall see below, each command has different features in the context of the brackets used in representing a matrix. For instance, if we use the matrix environment, no brackets will appear in the matrix as shown below.

```
\[\begin{matrix}
5  &  0  &  1\\
3  &  -1  &  0\\
4  &  5  &  1+i
\end{matrix}\]
```

This produces the matrix $\begin{matrix} 5 & 0 & 1 \\ 3 & -1 & 0 \\ 4 & 5 & 1+i \end{matrix}$

If we use `bmatrix` environment, then the matrix is enclosed in square brackets, as shown bellow.

```
\[\begin{bmatrix}
5  &  0  &  1  \\
3  &  -1  &  0  \\
4  &  5  &  1+i
\end{bmatrix}\]
```

This produces the matrix $\begin{bmatrix} 5 & 0 & 1 \\ 3 & -1 & 0 \\ 4 & 5 & 1+i \end{bmatrix}$

The `pmatrix` environment encloses the matrix in parenthesis. For example:

```
\[\begin{pmatrix}
5  &  0  &  1  \\
3  &  -1  &  0  \\
4  &  5  &  1+i
\end{pmatrix}\]
```

This produces the matrix $\begin{pmatrix} 5 & 0 & 1 \\ 3 & -1 & 0 \\ 4 & 5 & 1+i \end{pmatrix}$

The `vmatrix` command encloses the matrix in vertical lines as shown below.

```
\[\begin{vmatrix}
5 & 0 & 1 \\
3 & -1 & 0 \\
4 & 5 & 1+i
\end{vmatrix}\]
```

$$\begin{vmatrix} 5 & 0 & 1 \\ 3 & -1 & 0 \\ 4 & 5 & 1+i \end{vmatrix}$$

The Vmatrix command encloses the matrix in double vertical lines. For example,

```
\[\begin{Vmatrix}
5 & 0 & 1 \\
3 & -1 & 0 \\
4 & 5 & 1+i
\end{Vmatrix}\]
```

produces the matrix as $\begin{Vmatrix} 5 & 0 & 1 \\ 3 & -1 & 0 \\ 4 & 5 & 1+i \end{Vmatrix}$

Without using the above mentioned commands, matrices can also be produced using array environment. For instance,

```
\[
\left( \begin{array}{ll}
x & 1+x \\ 1-x & 1+ x^2
\end{array}\right)
\]
```

produces the matrix $\left(\begin{array}{ll} x & 1+x \\ 1-x & 1+ x^2 \end{array}\right)$

```
\[
\left[ \begin{array}{ll}
x & 1+x \\ 1-x & 1+ x^2
\end{array}\right]
\]
```

produces the matrix $\begin{bmatrix} x & 1+x \\ 1-x & 1+x^2 \end{bmatrix}$.



CHECK YOUR PROGRESS

Q 7: Write down the following matrices in LaTeX

(i) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (ii) $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ (iii) $\begin{Vmatrix} 5-2 \\ 0 \quad 4 \end{Vmatrix}$

2.7 TABLE

To arrange things in tabular form, a tabular environment is used. Here we show an example of a table with three columns and with this example, let us describe various features of tabular environment.

```
\begin{center}
      \begin{tabular}{  |c|c|c|  }
            \hline
a1  &a2  &a3  \\
a4  &a5  &a6  \\
a7  &a8  &a9  \\
            \hline
      \end{tabular}
\end{center}
```

In this example the tabular environment is introduced in between `\begin{center}` and `\end{center}` environment. This is because normally a table appears by default in left side of the page. To display the table in the centre of the page, `\begin{center}` ...`\end{center}` can be used.

In the command line, `\begin{tabular}{ccc}`, `{ccc}` declares that the table have three columns. Here `c` is used to instruct the table to display the content of the column at centre place of the column. Instead of `c`, we can use `r` to align the text to the right and `l` for left alignment. In the above example, instead of using `\begin{tabular}{ccc}`, we use `\begin{tabular}{|c|c|c|}`. The only difference is it declares that the

table has three columns, each column are separated by a vertical line.

We have also used `\hline` for inserting a horizontal line on top of the table and at the bottom. There is no restriction on the number of times we can use `\hline`. Further, each `&` sign in `a1 &a2 &a3` act as a separator and the double-backslash `\\` sets the end of the row. We have provided another example below where, in order to introduce a caption of the table, everything is written in between `\begin{table} … \end{table}` environment.

```
\begin{table}
\caption{Size of papers in inches.}
\label{paper}
\begin{tabular}{|c|r|}\hline
&Size& Width $\times$ Height (inches)\\
A0 & $66.2\times 93.6$ \\\hline
A1 & $23.4\times 33.1$ \\\hline
A2 & $16.5\times 23.4$ \\\hline
A3 & $11.7\times 16.5$ \\\hline
A4 & $8.3\times 11.7$ \\\hline
A5 & $5.8\times 8.3$ \\\hline
A6 & $4.1\times 5.8$ \\\hline
A7 & $2.9\times 4.1$ \\\hline
A8 & $2.0\times 2.9$ \\\hline
A9 & $1.5\times 2.0$ \\\hline
A10 & $1.0\times 1.5$ \\\hline
\end{tabular}
\end{table}
```

The table appears as follows.

Table 1. Size of papers in inches.

Size	Width × Height(inches)
A0	66.2 × 93.6
A1	23.4 × 33.1
A2	16.5 × 23.4

A3	11.7 × 16.5
A4	8.3 × 11.7
A5	5.8 × 8.3
A6	4.1 × 5.8
A7	2.9 × 4.1
A8	2.0 × 2.9
A9	1.5 × 2.0
A10	1.0 × 1.5



2.8 LET US SUM UP

- LaTeX has a particular environment following which we can create the documents of required style.
- There are commands for font styles and font sizes. The commands for different font sizes are \tiny, \small, \scriptsize, \footnotesize, \normalsize, \large, \Large, \LARGE, \huge, \Huge, etc. Similarly, different font style commands are available in LaTex a few of them are described in Table 2(b).
- Numerous mathematical symbols including Greek letters, binary operators and inequalities, variable-sized math operators, arrows, accents can be easily produced in LaTeX as displayed in Table 3. Some of them require amssymb package.
- In LaTeX, various equation environments are available which can be used to display equations in different styles and formats as described in Sec. 2.5.
- Matrices of different types can be easily produced in LaTeX by using either the available commands as described in Sec. 2.6 or using `array` environment.
- Tables can be created in LaTeX using `tabular` environment.



2.9 FURTHER READINGS

1) F. Mittelbach, M. Goossens, *The LaTeX Companion,* 2nd ed. Addison- Wesley (2004), ISBN 0-201-36299-6.

2) L. Lamport: *LATEX, A Document Preparation System, User's Guide and Reference Manual,* 2nd ed., Addison-Wesley (1994), ISBN 0-201-52983-1.



2.10 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1:
```
\documentclass{article}
\begin{document}
The texts appear here.
\end{document}
```

Ans to Q No 2: `\tiny, \small, \scriptsize, \footnotesize, \normalsize, \large, \Large, \LARGE, \huge, \Huge.`

Ans to Q No 3: They produce the same output: *software*.

Ans to Q No 4: By writing as {\sc The Earth revolves around the Sun}, we can change the font-size to small capitals as: THE EARTH REVOLVES AROUND THE SUN.

Ans to Q No 5: (i) `$\angle 180^\circ=\pi$`. (ii) `$\sqrt[5]{y^3}$` (iii) `$A\cup(B\cap C)=(A\cup B)\cap(B\cup C)$` (iv) `Golden ratio $\approx 1.618$.`

Ans to Q No 6:

(i) `$|\delta Z(t)|=e^{\lambda t}|\delta Z_0|$`

(ii) `$F=\frac{Gm_1m_2}{r^2}$`

(iii) `$\nabla\cdot E=\frac{\rho}{\epsilon_0}$`

(iv) `$\frac{\partial V}{\partial t}+ \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial^2 S}=rV-rS\frac{\partial V}{\partial S}$.`



2.11 MODEL QUESTIONS

Q 1: How can we produce $\frac{df}{dx} = \lim_{n\to\infty} \frac{f(x+h)-f(x)}{h}$ using LaTeX?

Q 2: Write down the codes for producing the matrix $\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$

Q 3: Try to produce the above matrix using array environment.

Q 4: Write down the codes for producing the matrix using array environment

$$\begin{matrix} 4 & 5 & 9 \\ -2 & 7 & 0 \\ 0 & 4 & 1 \end{matrix}$$

Q 5: Try to produce the expression $x = \frac{-5\pm\sqrt{9-6i}}{3}$.

Q 6: Write in LaTeX to produce the Fourier series, given by

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$$

Q 7: How can we write in LaTeX to produce the following equation:

$$(x+a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k}.$$

*** ***** ***

UNIT 3 : PAGE LAYOUT – I

UNIT STRUCTURE

3.1 Learning Objectives

3.2 Introduction

3.3 Page Layout

3.4 Titles

3.5 Abstract

3.6 Chapters

3.7 Sections and Subsections

3.8 References

- 3.8.1 Referencing within a LaTeX file
- 3.8.2 Preparing a separate BibTeX file

3.9 Equation References

3.10 Table of Contents

3.11 Let Us Sum Up

3.12 Further Readings

3.13 Answers to Check Your Progress

3.14 Model Questions

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- customize the layout of a LaTeX document
- write an article or a research paper
- create chapters and sections and subsections in Latex
- create a reference list
- generate a table of contents

3.2 INTRODUCTION

In the previous unit we have learnt about the basic syntax used in LaTeX. We have learnt how to generate various mathematical symbols and operators and how to write mathematical expressions and equations.

We have also learnt how to construct matrices and tables.

In this unit we shall discuss about the main page layouts for producing documents of various styles. A standard short article or a research manuscript comprises of a specific layout involving sections, subsection, chapters, references, tables and figures. In LaTeX, it is easier to display everything using the available commands. Here we shall describe how to prepare the layout of some standard documents and how to change page margins and sizes, headers and footers, and the proper placement of figures and tables on the page.

3.3 PAGE LAYOUT

As we have already described in Unit 2, a LaTeX document begins with a `\documentclass` command and any text to be printed must be included in between `\begin{document}` and `\end{document}` environment. The `\begin{document}` command is sometimes preceded by commands that set the page-style and set up user-defined control sequences. A typical LaTeX input file is as follows:

```
\documentclass{article}
\begin{document}
Write here the content of the article
\end{document}
```

The command begins with `\documentclass` and ends with the names of one of the available styles, enclosed in curly brackets. The available styles are `article, report, book, letter, beamer`,etc. Typing simply `\documentclass {article}` will produce a document in `10pt` type size. It is important to remember that a "`pt`" is a length unit which is approximately 1/72.27 inch, i.e. about 0.0138 inch. If we only declare `\documentclass{article},` the printed output will not be nicely positioned on A4 paper, since the default size is intended for a different (American) paper size. Thus, we can declare it as `\documentclass[12pt, a4paper]{article}.` This will create the output in an A4 size paper with 12pt font size. This is a simple example of customizing the layout. There are many commands that can be used

effectively to customize the layout of the page. Below we shall describe how to customize the page layout using a few easy-to-remember commands. Here are some important points to remember before customizing a page in LaTeX.

- The normal text area (Body) contains the running text including footnotes, tables and figures. The headings, footer and margin notes do not belong to the text area. The text area has the width `\textwidth` and the height `\textheight`.
- The left margin is either `\oddside margin` or `\evensidemargin` plus 1 inch. It is called `\evensidemargin` on even numbered pages and `\oddsidemargin` on odd-numbered pages. In one-sided documents `\oddsidemargin` is used for either.
- The top margin is the sum of `\topmargin`, `\headheight`, and `\headsep` plus 1 inch.
- The right margin is the paper width minus the left margin and the text area.
- The bottom margin is the paper height minus the top margin and the text area.
- The page headers and footers are defined by the `\pagestyle` and `\pagenumbering` commands. The command `\pagestyle` defines the general contents of the headers and footers while `\pagenumbering` defines the format of the page number. LaTeX uses four standard page styles as given bellow.

  `empty` no headers or footers; no page number will be printed.

  `plain` no header; footer contains page number at the center.

  `headings` footer; header contains name of chapter/section and/or subsection and page number

  `myheadings` no footer; header contains page number and user supplied information

The `\pagenumbering{}` command defines the layout of the page number. It has a number of parameter such as

arabic	Arabic numerals
roman	lower case roman numerals
Roman	upper case roman numerals
alph	lower case letters
Alph	upper case letters

The command \onecolumn produces a single-column output and \twocolumn produces two column output. In a \twocolumn layout, the text area is split into two columns. The distance between the two columns can be set using the command \columnsep. If we define in the preamble \setlength{\columnsep}{50pt}, it will change the separation between two columns as 50pt. The default value is "35pt". The width of a column can be set by using the command \columnwidth. In one-column mode, this is equal to \textwidth. In two-column mode, by default LaTeX sets the width of each of the two columns to be half of \textwidth minus \columnsep. The height of the box that contains running head can be set by using \headheight. The default value in the article, report, and book classes is "**12pt**" for all type sizes. The vertical distance between the bottom of the header line and the top of the main text can be set by using \headsep. The default value in the article and report classes is 25pt. In the book class, the default is: 0.25in for a document of type size 10pt and 0.275in if the document is set at 11pt or 12pt. The distance from the baseline of the last line of the text to the baseline of the page footer can be set by using \footskip. The default in the article and report classes is 30pt. In the book class the default is: 0.35in for a document type size 10pt, 0.38in for a document of type size 11pt and 30pt for a document of 12pt. The normal vertical height and full horizontal width of the page body can be set using \textheight and \textwidth, respectively. In addition to the above, there are some other commands for setting lengths. Below we list the above mentioned commands for lengths in tabular form.

Table 3.1 Basic length commands for customizing page layout.

Commands	Description
\evensidemargin	Margin of even pages, commonly used in books
\oddsidemargin	Margin of odd pages, commonly used in books
\columnsep	Distance between columns
\columnwidth	The width of a column
\textheight	Height of the text area in the page
\textwidth	Width of the text area in the page
\headsep	Separation between the bottom of the header line and the top of the main text
\headheight	The height of the box that contains running head
\footskip	Distance from the baseline of the last line of the text to the baseline of the page footer
\parskip	Vertical space between paragraphs
\topmargin	Length of the top margin

Here is an example showing how to customize some basic lengths.

```
\documentclass[11pt,a4paper]{article}
\topmargin 1cm
\setlength{\headheight}{0cm}
\setlength{\headsep}{0cm}
\setlength{\textheight}{22cm}
\setlength{\textwidth}{15cm}
\setlength{\oddsidemargin}{0.8cm}
```

There is a standard package called `fancyhdr` macro package that allows customizing page headers and footers in a easier way. It is particularly useful to define

- decorative lines in headers and footers
- wider headers and footers than the width of the text
- multi-line headers and footers
- separate headers and footers for even and odd pages
- different headers and footers for chapter pages

Before including this package in a LaTeX document, a style file `fancyhdr.sty` is required to install and it has to store in a directory/ folder where TEX can find it. Now it can be included in the preamble of the document after `\documentclass {...}` command as

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

Here is a simple example to show how to use `\pagestyle {fancy}` under `\usepackage{fancyhdr}` macro package. The layout is given below:

Left header	Centered header	Right header
	Page body	
Left footer	Centered footer	Right footer

The corresponding commands are `\lhead, \chead, \rhead, \lffot, \cfoot, \rfoot` for left header, centered header, right header, left foot, centered foot, and right foot. For instance, let us consider the following layout of writing a letter by Johnny Depp to the Editorial Team on the publication of a magazine article. This can be produced by using `\pagestyle{fancy}` as follows:

		Publication of an article
	Page body	
From: Johnny Depp	To: The Editorial Team	5

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\lhead{}
\chead{}
```

```
\rhead{\bf Publication of an article}
\lfoot{From: Johnny Depp}
\cfoot{To: The Editorial Team}
\rfoot{\thepage}
```

Here `\thepage` command is used to display the current page number at the position of right foot.



CHECK YOUR PROGRESS

Q 1: Write down the commands that are used for changing the height and width of the text area in a page. Set the height and width of the text as 9.0 inch and 6.1 inch, respectively.

.....

.....

Q 2: How to change the font size of the text from `11pt` to `12pt`?

.....

.....

Q 3: How to use the package `fancyhdr`?

.....

.....

Q 4: What are the commands in the package `fancyhdr` that are used for setting lengths?

.....

.....

Q 5: Which command is used in `fancyhdr` to display the current page number at the position of right foot?

.....

.....

3.4 TITLES

The title of a document can be displayed by declaring it in the preamble of the LaTeX file. Here we show a simple structure of a document which will display its title and the authors.

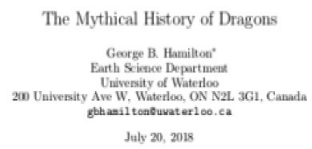
```
\documentclass[12pt,a4paper]{article}
\usepackage{amsmath,amssymb}
\begin{document}
\title{titleline1\\ titleline2}
\author{name\\
addressline1\\
addressline2\\
\date{date}
\maketitle
\end{document}
```

Here we have used the `article` as document class. In fact, there exist other document classes in LaTeX such as `book`, `letter, report`, etc. We shall discuss them separately in details in Unit 6. Here we shall only discuss how to insert title in a document. As we see in the above example, we use the command `\title{..}` where we can enter the title of the document. New lines can be added here by putting newline command (`\\`). The command `\author{..}` is used to write the authors name and address. Multiple lines can be added here just like in `\title{..\\..}`. The command `\date{..}` is used to display the date. Depending on the class of the document, LaTeX may automatically generate the date when the document is compiled. In case if we do not like to display date, we need to specify an empty date as `\date{}`.We can write `\date{\today}` to display the current date. All these belong to the *top matter* of a standard article that contains title, authors, and date.

All these font matters are then printed in the output file as it is by inserting `\maketitle` command as we have done above. It is important to remember that the `\maketitle` command must come after the `\begin{document}` command. An example is shown below.

```
\title{The Mythical History of Dragons}
\author{George B. Hamilton\thanks{Research supported by the National science foundation under grant number~93569.}\\Earth Science Department\\University of Waterloo\\200 University
```

```
Ave W, Waterloo, ON N2L 3G1, Canada\\
\texttt{gbhamilton@uwaterloo.ca} \date{July 20, 2018}
\maketitle
```



The Mythical History of Dragons

George B. Hamilton*
Earth Science Department
University of Waterloo
200 University Ave W, Waterloo, ON N2L 3G1, Canada
gbhamilton@uwaterloo.ca

July 20, 2018

*Research supported by the National science foundation under grant number 93569.

In the above example, the `\maketitle` command is not going to print the font matters in a separate page. Without using `\maketitle` command, a title page can be created in a separate page. The title page includes as usual the title, authors name and address, and the date. It can be done by including all the commands for *top matter* inside the `\begin{titlepage}` and `\end{titlepage}` commands as follows.

```
\documentclass[12pt,a4paper]{article}
\usepackage{amsmath,amssymb}
\begin{document}
\begin{titlepage}
\title{titleline1\\ titleline2}
\author{name\\
addressline1\\
addressline2\\
\date{date}
\end{titlepage}
\end{document}
```

While writing the author's name inside the field `\author{..}`, at the same time we can create a footnote to display some important

information about the author.

For instance, the author's email or alternative email can be added in the footnote using the command \thanks.



CHECK YOUR PROGRESS

Q 6: How to write author's name, affiliation, and address in a LaTeX file?

.....

.....

Q 7: What is the command for producing the title page in the output file of LaTeX?

.....

.....

Q 8: Write down the command for displaying the current date?

.....

.....

3.5 ABSTRACT

The abstract of an article can be included in the `abstract` environment just after declaring the *front matter* as follows.

```
\documentclass[12pt,a4paper]{article}
\usepackage{amsmath,amssymb}
\begin{document}
\title{titleline1\\titleline2}
\author{name\thanks{support}\\
addressline1\\
addressline2\\
\date{date}
\maketitle
\begin{abstract}
Write the abstract here
\end{abstract}
\end{document}
```

An example is as follows.

```
\begin{abstract}
It is argued that dragons evolved from rainbows through the concept of the rainbow serpent, a concept that itself extends far back into the Pleistocene. In this perspective many seemingly arbitrary traits which are widely associated with dragons are seen to have a physical explanation. This article presents a brief historical record related to the archeological survey on dragons and the scientific explanations for their existence.} \end{abstract}
```



The Mythical History of Dragons

George B. Hamilton*
Earth Science Department
University of Waterloo
200 University Ave W, Waterloo, ON N2L 3G1, Canada
gbhamilton@uwaterloo.ca

July 20, 2018

Abstract

It is argued that dragons evolved from rainbows through the concept of the rainbow serpent, a concept that itself extends far back into the Pleistocene. In this perspective many seemingly arbitrary traits which are widely associated with dragons are seen to have a physical explanation. This article present a brief historical record related to the archeological survey on dragons and the scientific explanations for their existence.

*Research supported by the National science foundation under grant number 93569.

It is important to remember that the abstract environment is only defined in the `article` and `report` document classes.



CHECK YOUR PROGRESS

Q 9: What environment is required to write abstract in a document?

.....

.....

Q 10: What are the document classes that accept abstract environment?

.....

.....

3.6 CHAPTERS

Creating and handling chapters in LaTeX is very easy. In the document classes `report` and `book` we can easily structure the document

into chapters. Chapters can be written using the command `\chapter[toctitle]{title}`.

In addition to providing the heading `title` in the main text, the chapter title can appear in two other places: (i) the table of contents and (ii) the running head at the top of the page. Some books or reports use a style such that the same text does not display in these places as in the main text. This can be handled easily by using the optional argument `toctitle`. Whatever is written in the place of `toctitle`, it appears in these other places. For instance, if we declare a chapter as `\chapter[The Origin]{The Origin of Supernova Explosion}`, It displays the title as in the main text.

3. The Origin of Supernova Explosion

However, in the table of contents and in the running head of the book or report, it only writes `The Origin` as the chapter's name. LaTeX takes care of numbering chapters. It automatically generates the numbers. In the above example, the number 3 indicates that it is the third chapter of the book. If we want a chapter heading without a number, we have to add an asterisk to the command. The chapter number can be used for citation by using a label key. For instance, if we write `\chapter[The Origin]{The Origin of Supernova Explosion}\label{ch:supernova}`, then we can use the command `\ref{ch:supernova}` to cite the chapter wherever required. Here, `ch:supernova`is a key for the chapter which is written as an argument to the command `\label{}`. The same key is then used to refer in the main text using the command `\ref{}`.

It is important to note that in the document class `article`, we cannot use `\chapter` command. It is applicable only for `book` and `report`.

3.7 SECTIONS AND SUBSECTIONS

Sections, subsections, and sub-subsections can be created in LaTeX using the commands `\section[toctitle]{title}`, `\subsection[toctitle]{title}`, `\subsection[toctitle]{title}`.

Structurally, all these commands have the same general form as in chapters. These commands automatically number and write the section or subsection title in the appropriate format according to the document class and font size specified in the preamble. As in chapters, the corresponding numbers generated in sections, subsections, or sub-subsections can also be cited using the `\label{}`and `\ref{}`commands.



CHECK YOUR PROGRESS

Q 11: How to generate a chapter with title “The Cosmos”?

.....

.....

Q 12: What is the method of citation for a particular section?

.....

.....

3.8 REFERENCES

Usually, referencing is a very strenuous and time consuming procedure. Latex includes features that allow easily citing references and creating bibliographies in a document. In fact, there are two different styles in LaTeX to incorporate reference of a document: One is one can add a list of reference towards the end of the .tex file and the other one is the creation of a separate BibTeX file and compile it with the LaTeX file. Here, we shall explain how to do all these with some examples.

3.8.1 Referencing within a LaTeX file

Let us look at the following texts of a book:

```
Euclidean geometry is a mathematical system
attributed to Alexandrian Greek mathematician
Euclid [1]. It consists in assuming a small set
of intuitively appealing axioms, and deducing
many other theorems from these.
```

Here the number [1] cites a reference `Werner Oechslin, *Geometry: Euclid and Beyond*, Taschen (2016).`

How to cite it in LaTeX? In order to cite it, we need a citation key. Using this citation key, we can cite it wherever it is required. In order to cite the above reference, we need to write \cite{Werner} where Werner is the citation key. Now towards the end of the document, we have to add this reference within the bibliography environment. This can be done as follows

```
\begin{thebibliography}{9}
\bibitem{Werner}
Werner Oechslin,
\newblock {\emGeometry: Euclid and Beyond},
\newblockTaschen (2016).
\end{thebibliography}
```

This will display a list of reference towards the end of the output (PDF) file as follows:

Reference

[1] WernerOechslin,*Geometry: Euclid and Beyond,* Taschen (2016).

Thus, we can add all the references by giving unique key to each reference. Let us now understand how it works. The `thebibliography{widest_label}` environment produces a bibliography or a reference list. In the `article` class, the Reference list is labelled **'References'** while in the report class, it is labelled as **'Bibliography'.** The mandatory `widest_label` argument is text that, when typeset, is as wide as the widest item label produced by the `\bibitem` commands. It is typically given as 9 for bibliographies with less than 10 references, 99 for ones with less than 100, etc.

The `\bibitem[label]{cite_key}` command generates an entry labeled by `label`. If the label argument is missing, a number is automatically generated. The `cite_key` is a citation key comprises of letters, numbers, and punctuation symbols without containing a comma.

3.8.2 Preparing a separate BibTeX file

A separate data file containing information for all the references required for a document is called a BibTeX file. In order to use such a BibTeX file, the following two lines need to be added towards the end of the LaTeX file or at places where we want to display it in the output pdf file.

```
\bibliographystyle{plain}
\bibliography{ref}
```

Here `ref` is the name of BibTeX file`ref.bib` that we need to create separately. The .bib file has all the information regarding the citation material e.g. title, journal, volume, issue, pages, etc. All these information must be included inside the `.bib` file in a particular format so that LaTeX can draw information as required from the particular citation style.

Let us assume that we have a `.tex`file with name `test.tex` and a separate `.bib` file named as `ref.bib`. While compiling, we have to run the following commands one-by-one

```
$latex test.tex
$bibtextest.bib
$pdflatextest.tex
```

It is important to remember that although the name of the `.bib` file is different (`ref.bib`) from that of the `.tex` file (`test.tex`), during compilation we should run the command `bibtextest.bib` but not the bibtexref.bib. Another important thing to remember is that the LaTeX file and the BibTeX file should be kept in the same folder.

The `.bib` file is plain text - it can be edited using Notepad or any other LaTeX editors (e.g. gedit). We need to enter each of our references in the BibTeX file in the following format:

```
@article{Hamilton2001,
author = {R. Hamilton and S. Smith},
title = {The Game of Group Hunting in Ecosystem},
```

```
journal = {Behavioral Ecology and Sociobiology},
volume = {50},
pages = {9-19},
year = {2001} }
```

Each reference starts with the reference type. In the above example, the reference type is `@article`. Other reference types include `@book`, `@incollection` for a chapter in an edited book and `@inproceedings` for papers presented at conferences.

The reference type declaration is followed by the citation key. Each reference's citation key must be unique - in the above example it is `Hamilton2001`.The remaining lines contain the reference information which need to be entered against the available field format: `field name={field contents}`.

One noticeable thing here is that, BibTeX automatically uncapitalises any capital letters within the journal article title. For instance, in the above example, the title of the article is entered as "`The Game of Group Hunting in Ecosystem`". However, in the output pdf file, it will appear as`The game of group hunting in ecosystem`. In order to capitalize any letters in a journal article title, it has to be surrounded by curly brackets `{…}`Thus, if we write `title = {The {G}ame of {G}roup {H}unting in {E}cosystem}`, it will be printed as at is. Usually LaTeX has several styles with numerical in-text citations such as :

- **`plain:`** This style displays the citation as a number in square brackets (e.g. [1]). The bibliography is ordered alphabetically starting with the author surname. All of the authors' names are written in full.
- **`abbrv:`** It is the same as plain except the authors' first names are abbreviated to an initial.
- **`unsrt:`** It is the same as plain except the references in the bibliography appear in the order that the citations appear in the document.

- **alpha:** It is the same as plain except the citation is an alphanumeric abbreviation based on the author(s) surname(s) and year of publication, surrounded by square brackets (e.g. [Kop10]).

In the following section you see how different BibTeX styles look in the resulting PDF. The style is defined in the `\bibliographystyle{style}` command where `style` is to be replaced with one of the following styles (e.g. `alpha`, etc.). The following bibliography inputs were used to generate the result:

```
@book{latex-companion,
author      = "Michel Goossens and Frank
              Mittelbach and Alexander Samarin",
title       = "The \LaTeX\ Companion",
year        = "1993",
publisher   = "Addison-Wesley",
address     = "Reading, Massachusetts"
}

@booklet{booklet,
title       = {The Miracle Morning for
              Entrepreneurs},
author      = {Hal Elrod and Cameron Herold},
howpublished = {Paperback},
address     = {Kindle Edition},
month       = 7,
year        = 2018,
note        = {An optional note}
}

@conference{conference,
author      = {Peter Draper},
title       = {The title of the work},
booktitle   = {The title of the book},
year        = 1993,
```

```
editor      =  {The  editor},
volume      =  4,
series      =  5,
pages       =  213,
address     =  {The  address  of  the  publisher},
month       =  7,
organization  =  {The  organization},
publisher   =  {The  publisher},
note        =  {An  optional  note}
}

@incollection{incollection,
author      =  {Peter  Farindon},
title       =  {The  title  of  the  work},
booktitle   =  {The  title  of  the  book},
publisher   =  {The  name  of  the  publisher},
year        =  1993,
editor      =  {The  editor},
volume      =  4,
series      =  5,
chapter     =  8,
pages       =  {201-213},
address     =  {The  address  of  the  publisher},
edition     =  3,
month       =  7,
note        =  {An  optional  note}
}

@manual{manual,
title       =  {The  title  of  the  work},
author      =  {Peter  Gainsford},
organization  =  {The  organization},
address     =  {The  address  of  the  publisher},
edition     =  3,
```

```
month       =  7,
year        =  1993,
note        =  {An  optional  note}
}

@mastersthesis{mastersthesis,
author      =  {Peter  Harwood},
title       =  {The  title  of  the  work},
school      =  {The  school  of  the  thesis},
year        =  1993,
address     =  {The  address  of  the  publisher},
month       =  7,
note        =  {An  optional  note}
}

@misc{misc,
author      =  {Peter  Isley},
title       =  {The  title  of  the  work},
howpublished  =  {How  it  was  published},
month       =  7,
year        =  1993,
note        =  {An  optional  note}
}

@phdthesis{phdthesis,
author      =  {Peter  Joslin},
title       =  {The  title  of  the  work},
school      =  {The  school  of  the  thesis},
year        =  1993,
address     =  {The  address  of  the  publisher},
month       =  7,
note        =  {An  optional  note}
}
```

```
@proceedings{proceedings,
title      = {The title of the work},
year       = 1993,
editor     = {Peter Kidwelly},
volume     = 4,
series     = 5,
address    = {The address of the publisher},
month      = 7,
organization  = {The organization},
publisher  = {The name of the publisher},
note       = {An optional note}
}

@techreport{techreport,
author     = {Peter Lambert},
title      = {The title of the work},
institution = {The institution that published},
year       = 1993,
number     = 2,
address    = {The address of the publisher},
month      = 7,
note       = {An optional note}
}

@unpublished{unpublished,
author     = {Peter Marcheford},
title      = {The title of the work},
note       = {An optional note},
month      = 7,
year       = 1993
}
```



CHECK YOUR PROGRESS

Q 13: Write down different types of reference that are used in creating bibliography.

.....

.....

Q 14: Write down the different styles used in writing a bibliography.

.....

.....

Q 15: What is the environment for creating a reference list in the LaTeX file?

.....

.....

Q 16: Write down the commands that are required in order to compile a `.tex` and a `.bib` file together.

.....

.....

3.9 EQUATION REFERENCES

As described in Unit 2, there are three different way of displaying equations: in-line with text, as a numbered or unnumbered equation on its own line, or as a block of equations. Cross referencing an equation is possible only if the equation is written within mathematical environment.

In the equation environment of LaTeX, we can generate equation number and, whenever required, we can cite it at appropriate places. For instance,

```
\begin{equation}
E=mc^2 \label{eq:E}
\end{equation}
Eq. \ref{eq:E} is one of the classic inventions in the
history of science.
```

As we see here, there is a label for the equation with `eq:E` as the label key. This means the equation number that is going to generate

by this equation environment can be referred by writing `\ref{eq:E}`.

In the above example, instead of \begin{equation} and \end{equation}, we can simply write the equation within a pair of delimiter \[\] which can also do the same thing. We can also use `\begin{eqnarray}\end{eqnarray}` environment for referencing an equation.

3.10 TABLE OF CONTENTS

In LaTeX, a table of content can be automatically created just by using the command `\tableofcontents` after the `\begin{document}` command. One example is shown below.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\title{Fibonacci Series}
\author{Michel Jin Justin}
\date{}
\begin{document}
\maketitle
\tableofcontents
```

We can put the command `\tableofcontents` where we want the table of contents to go. While compiling the `.tex` file, a `.toc` file will be generated that contains the information for table of contents.

The `\tableofcontents` command produces a heading, but it does not automatically start a new page. If we want a new page after the table of contents, we need to write a `\newpage` command after the `\tableofcontents` command.



3.11 LET US SUM UP

- In LaTeX, the page layout can be customized using a few easy-to-understand commands as mentioned in Section 3.3.
- The front matter of a document containing the title of the document,

author's name, and date can be created using commands in the title page environment.

- An abstract can be created by writing the text in an abstract environment. This is possible only in the `article` and `report` document classes.
- A chapter in a book can be created by using `\chapter{…}` command. In the document class `article`, we cannot use `\chapter` command; it is applicable only for `book` and `report`.
- Sections, subsections, or sub-subsections can be created by the commands `\section{..}`, `\subsection{…}` and `\subsubsection{…}`.They can be referred by using `\label{key}` and `\ref{key}` commands.
- Reference lists can be created in LaTeX either by writing in the `.tex` file or by creating a separate BibTeX file.
- Equations can be referred by using `\label{key}` and `\ref{key}` commands.
- A table of contents can be created automatically in LaTeX using `\tableofcontents` command.



3.12 FURTHER READINGS

1) Knuth, D. E., & Bibby, D. (1984). *The texbook (Vol. 3)*. Reading: Addison-Wesley.

2) Lamport, L. (1994). *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley.

3) Heck, A. (2005). *Learning LATEX by Doing*. AMSTEL Institute.



3.13 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: `\textheight}`and`\textwidth}`are the commands for setting the height and width of the text area in a page. The

commands for changing the height of the text to 9.0 inch and width of the text to 6.1 inch are

```
\setlength{\textheight}{9.0in}
\setlength{\textwidth}{6.1in}
```

Ans to Q No 2: The font size of the text can be changed from `11pt` to `12pt` by replacing the `11pt` appeared in `\documentclass [11pt]{article}` to 12pt.

Ans to Q No 3: The package `fancyhdr`can be used in LaTeX by declaring the following two line after `\documentclass{}`:

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

Ans to Q No 4: In `fancyhdr`package, `\lhead`, `\chead`, `\rhead`, `\lffot`, `\cfoot`, `\rfoot` are the commands for setting left header, centered header, right header, left foot, centered foot, and right foot, respectively.

Ans to Q No 5: In `fancyhdr`, `\thepage` command is used to display the current page number at the position of right foot.

Ans to Q No 6: Within the argument of the command `\author{}`, we can write the author's name, affiliation, and address. We can use new line command `\\` to create a separate line for each field.

Ans to Q No 7: In the title page, the available fields are `\title{}`, `\author{}`, and `\date{}`. After putting information in all these fields, we can use `\maketitle` command to display them. Alternatively, the title page can be created by putting the information in the environment `\begin{titlepage} …. \end{titlepage}`.

Ans to Q No 8: The current date can be displayed in the title page by writing `\date{\today}`.

Ans to Q No 9: An abstract can be created by using the environment `\begin{abstract}….\end{abstract}`.

Ans to Q No 10: Only the `article` and `report` document classes accept `abstract` environment.

Ans to Q No 11: A chapter with title "The Cosmos" can be generated by typing `\chapter{The Cosmos}`.

Ans to Q No 12: The corresponding number generated by a `\section{}` command can be cited using the `\label{key}` and `\ref{key}` commands.

Ans to Q No 13: `@book`, `@booklet`, `@conference`, `@incollection`, `@manual`, `@masterthesis`, `@proceedings`, `@techreport`, etc.

Ans to Q No 14: `plain`, `abbrev`, `unsrt`, `alpha` arethe styles used in writing a bibliography.

Ans to Q No 15: There are two different environments for creating a reference list in the LaTeX file: one is by writing the details of the reference in the LaTeX file within `\begin{thebibliography}…..\end{thebibliography}` environment and the other one is by preparing a separate BibTeX file and declaring the commands

```
\bibliographystyle{plain}
\bibliography{name of .bib file}
```

in the LaTeX file where we want to display the reference list.

Ans to Q No 16:

```
$latex filename
$bibtex filename
$pdflatex filename
```



3.14 MODEL QUESTIONS

Q 1: Prepare a list of commands used in customizing header and footer of a document in A4 size paper.

Q 2: What are the commands used within the `fancyhdr` style?

Q 3: How to add the name of two authors having different address and affiliations?

Q 4: Write down the structure of LaTeX file which can create an abstract of an article.

Q 5: Which command is used in `fancyhdr` to display the current page number at the position of right foot?

Q 6: How to create a chapter of a book? In which document classes chapters can be created?

Q 7: How to create sections and subsections of an article or a book?

Q 8: Write down the structure of LaTeX file which can create a reference list.

Q 9: What is BibTeX? How to create a BibTeX file?

Q 10: Suppose you have a LaTeX file named mylatex.tex and a separate BibTeX file named reference.bib. How to produce a pdf file out of them that can display the cited references?

Q 11: Explain how the styles `plain, abbrev, unsrt, alpha` work while used in bibliography.

Q 12: What is the command to create a table of contents?

Q 13: Write down the structure of a simple LaTeX file that can create a table of contents of a book.

*** ***** ***

UNIT 4 : PAGE LAYOUT– II

UNIT STRUCTURE

4.1 Learning Objectives

4.2 Introduction

4.3 List Making Environments

- 4.3.1 Enumerate
- 4.3.2 Itemize
- 4.3.3 Description

4.4 Generating New Commands

4.5 Figure Handling

4.6 Table and Figure Numbering

4.7 List of Figures and List of Tables

4.8 Generating Index

4.9 Let Us Sum Up

4.10 Further Readings

4.11 Answers to Check Your Progress

4.12 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- create and display a list of items in different style
- handle figures and tables in a document
- generate a list of figures and tables as a front matter of a long document
- generate index

4.2 INTRODUCTION

In the previous unit, we discussed about the main page layouts for producing documents of various styles. We have learnt how to create a specific layout involving sections, subsections, chapters and references. We have also learnt how to prepare the layout of some standard

documents and how to change page margins and sizes, headers and footers, and the proper placement of figures and tables on the page.

In this unit, we shall describe how to create and display a list of items. LaTeX is designed to create a logically consistent document structure just by using commands. We shall also discuss how to handle figures and tables in a LaTeX file. For long articles or for a book, a tedious job is to produce a list of tables or a list of figures. In this unit, we shall describe how they become easier in LaTeX just by inserting a few easy-to-remember commands. Further, we shall also describe how to generate a list of index.

4.3 LIST MAKING ENVIRONMENTS

The list environment is a generic environment for constructing more specialized lists:

```
\begin{list} ... \end{list}
```

It is most commonly used to create lists of different types via the following list environments:

- *enumerate* for numbered lists,
- *itemize* for unnumbered lists,
- *description* for description lists

4.3.1 Enumerate

Numbered lists are produced using

```
\begin{enumerate} ... \end{enumerate}
```

All the items to be appeared in the list need to be enclosed between `\begin{enumerate}` and `\end{enumerate}`. For example,

```
\begin{enumerate}
\item First thing
\item Second thing
\end{enumerate}
```

```
1. First thing
2. Second thing
    • A sub-thing
    • Another sub-thing
3. Third thing
```

Here, the control sequence `\item` automatically generates the

number labelling the item. The format of the label numbering depends on the nesting level of this environment. The default top-level numbering is '1', '2'. etc. Each enumerate list environment described in the `.tex` file must contain at least one item. If there remains a blank enumerate environment, a LaTeX error message will appear during compilation as:

```
1. First thing
2. Second thing
```

'Something's wrong- - -perhaps a missing \item'.

4.3.2 Itemize

An unordered bulleted list can be generated using

`\begin{itemize} ... \end{itemize}`

A list can be nested to produce sub-lists. Each itemize list environment described in the .tex file must contain at least one item. If there remains a blank itemize environment, a LaTeX error message will appear during compilation as:

'Something's wrong- - - -perhaps a missing \item'.

By typing the following, one can produce a numbered list with a bulleted sub-list:

```
\begin{enumerate}
\item First thing
\item Second thing
\begin{itemize}
\item A sub-thing
\item Another sub-thing
\end{itemize}
\item Third thing
\end{enumerate}
```

First This is the first item
Second This is the second item

It is easy to change the bullet symbol using square brackets after the `\item`, for example, `\item[-]` will give a dash as the bullet. You can even use words as bullets, for example, `\item [One].`

The following code:

```
\begin{itemize}
\item[-] Fruits
\item[-] Vegetables
\begin{itemize}
\item[$] Green vegetables
\item[#] Yellow vegetables
\end{itemize}
\item[-] Other grocery
\end{itemize}
```

```
- Fruits
- Vegetables
$ Green vegetables
# Yellow vegetables
-Other grocery
```

4.3.3 Description

Description lists (for glossaries etc.) are produced using

`\begin{description} ... \end{description}`

It produces a customizable list. For example,

```
\begin{description}
\item[One] This is the first item
\item[Two] This is the second item
\end{description}
```

One This is the first item

Two This is the second item

Here is another example

```
\begin{description}
\item{[First]} This is the first item
\item{[Second]} This is the second item
\end{description}
```

In addition to the above mentioned environments for producing lists, there exist other options of creating lists. For instance, displayed quotations can be embedded in text using the quote and quotation environments.

`\begin{quote} ... \end{quote}`

The quote environment is recommended for short quotations: the whole quotation is intended in the quote environment,

but the first lines of individual paragraphs are not further indented. The input file

```
\begin{quote}
Change will not come if we wait for some other person or some other time. We are the ones we've been waiting for. We are the change that we seek. \hfill{Barack Obama}
\end{quote}
```

The `verse` environment is designed for poetry. The margins are indented on the left and the right, paragraphs are not indented, and the text is not justified. It separates the lines of each stanza with `\\`, and uses one or more blank lines to separate the stanzas.

```
\begin{verse}
Line 1 \\
Line 2 \\
…
\end{verse}
```

The `verbatim` environment prints its text in typewriter-style type and reproduces new lines and spaces exactly as they are in the input file. For instance,

```
\begin{verbatim}
I would like to see how verbatim environment works. I would like to put a big space here. And some characters % & $ ^ ! #.
\end{verbatim}
```

produces

```
I would like to see how verbatim environment works. I would like to put a big space here. And some characters % & $ ^ ! #.
```

Comments & Spacing

Comments are created using %. When LATEX encounters a % character while processing a `.tex` file, it ignores the rest

of the line until the [Return] key has been pressed to start a new line. The commented lines are often used to write notes in the input file that will not appear in the printed version.

For instance,

```
History reminds us the consequences of war % independent of the causality
on the evolution of human civilizations from ethnic period to civilized period.
```

produces:

```
History reminds us the consequences of war on the evolution of human civilizations from ethnic period to civilized period.
```



CHECK YOUR PROGRESS

Q 1: Write down the command for preparing the following list

1. BCA courses

(a) Theory courses

— Data structure

— Object oriented programming

(b) Practical courses

— C++

— Java

2. MCA courses

Q 2: Which environment produces a customizable list?

.....

.....

Q 3: Which list environment is useful for writing poetry?

.....

.....

4.4 GENERATING NEW COMMANDS

In LaTeX, `\newcommand` and `\renewcommand` define and redefine a command, respectively. They are very helpful in preparing a long document. They are written in the preamble of the `.tex` file as

```
\newcommand{\cmd}{defn}
\renewcommand{\cmd}{defn}
```

Here, `\cmd` is the command name. For `\newcommand`, it must not be already defined and for `\renewcommand`, it must already be defined. The `defn` is the text to be substituted for every occurrence of `\cmd`. For instance, if in a document the name A. P. J. Abdul Kalam appears several times, we need not to write it repeatedly if we create a new command for it. If we insert a line in the preamble as `\newcomand{\apj}{A. P. J. Abdul Kalam}`, we can produce this name in the output file just by typing `\apj` in the running text. Thus a `\newcommand` helps in replacing longer text by a short command. Sometimes some predefined commands appear repeatedly in a document. This can also be handled easily by using `\renewcommand`. For instance, if we declare in the preamble of the `.tex` file `\renewcommand{\beq}{\begin{equation}}`, just writing `\beq` will do the same job as `\begin{equation}`. This way, we can create a number of commands of our own and use them wherever required.



CHECK YOUR PROGRESS

Q 4: Create new-commands for `\begin{enumerate}`, `\begin{itemize}`, `\end{enumerate}`, and `\end{itemize}` and use them to create the list of items given in Q1.

.....

.....

Q 5: How do we use \newcommand? Create a new command for \frac{1}{r^2}.

.....

.....

Q 6: In a LaTeX file, where we need to define the `\newcommand` or `\renewcommand`?

.....

.....

4.5 FIGURE HANDLING

In order to insert an image in a LATEX document, we need to add in the preamble the `graphicx` or *`graphics`* package. The `\usepackage{graphics}` command instructs LaTeX to read the file `graphics.sty`. These packages can read the image files which could be in the PDF, PNG, JPEG, GIF, or EPS formats. Here is an example showing how to add a figure in a LaTeX document. Here we show adding an image called image.eps.

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{image.eps}
\caption{Here is an image}
\label{fig:image}
\end{figure}
```

Here `[h]` is the placement specifier which means put the figure approximately here. Other options are “t” (at the top of the page), “b” (at the bottom of the page) and “p” (on a separate page for figures). We can also add [!], which will put the figure where you want. If the image does not fit where it was written in the `.tex`, LaTeX will move the float to the next page and shift some text on top of it to minimise empty spaces.

`\centering` centers the image on the page, if not used images are left-aligned by default. It’s a good idea to use this as the figure captions are centered.

`\includegraphics{…}` is the command that actually resizes, transforms and positions the image in the document. The image file should be saved in the same folder as the .tex file.

`[width=1\textwidth]`is an optional command that specifies the width of the picture. The definition “`1\textwidth`” indicates that

the width is same as the text. The width could also be given in centimeters (cm). Alternatively, we can also use `[scale=0.5]` which scales the image by the desired factor, in this case reducing by half.

`\caption{...}`defines a caption for the figure. If this is used LaTeX will add “Figure” and a number before the caption.

`\label{...}`creates a label to allow us to refer to the table or figure in our text.



CHECK YOUR PROGRESS

Q 7: What packages are required to insert an image file in LaTeX?

.....

.....

Q 8: Write the codes to produce a figure named `phi.eps` at the center of the page and reduce its size to 0.5 of text width and 0.8 of text height.

.....

.....

4.6 TABLE AND FIGURE NUMBERING

Including the figure number in the source is poor practice since if that number changes as the document evolves, then we must remember to update this reference by hand. Instead, LaTeX has a label like `\label{key}` and refers to it with equation~`\ref{key}`. The `\label{key}` assigns a reference number to key. In ordinary text, the `\label{key}` assigns to key the number of the current sectional unit. Inside an environment with numbering, such as a table or theorem environment, `\label{key}` assigns to key the number of that environment. In order to retrieve the assigned number, the `\ref{key}` command is used.

A key name can be any sequence of letters, digits, or common punctuation characters. Upper and lowercase letters are distinguished, as usual. A common convention is to use labels consisting of a prefix and

a suffix separated by a colon or period. This helps us to avoid accidentally creating two labels with the same name, and makes your source more readable. Some commonly-used prefixes are

ch for chapters
sec for sections
fig for figures
tab for tables
eq for equations

For instance, `\label{fig:Einstein}` is a label for a figure with a portrait of Einstein; `\label{eq:Einstein}` is a label for an equation of Einstein. LaTeX writes the information from the labels to a file with the same name as the file containing the `\label{...}` but with an `.aux` extension.

In the above example, we have used a command called `\label{…}`. It creates a label to allow us to refer to the table or figure in our text. The figure is labelled using the key written between {..}. This can be used to refer to the figure number by simply writing `\ref{key}` anywhere in the document. It makes it easier to refer to the figure and the number will change automatically as the figure number changes.

The most common side effect of the prior paragraph happens when the document has a forward reference, a `\ref{key}` that appears earlier than the associated `\label{key}`. However, LaTeX gets the information for references from the .aux file. While compiling for the first time, A LaTeX warning message will appear as:

Label(s) may have changed. Rerun to get cross references right.

In addition, we will see in the output that the reference will appear as two question marks **'??', i**n boldface. The solution in either case is just to compile the document a second time.

4.7 LIST OF FIGURES AND LIST OF TABLES

One of the important lists of a long document that must appear before the main text is the table of contents. We have already described

in Section 3.10 about the table of contents. Just like `\tableofcontents` command, the commands `\listoffigures` and `\listoftables` produce a list of figures and a list of tables from `.lof` and `.lot` files, respectively. Everything works exactly the same as for the table of contents. Whatever is included in the caption of the tables and figures using `\caption{}` command, they appear in the list of tables and list of figures as font matters.

4.8 GENERATING INDEX

Generating an index is very easy using LaTeX commands. In order to create a list of index, we need to use the package `makeidx.` This means we need to load the package `makeidx` with the command `\usepackage{makeidx}` and put the command `\makeindex` in the preamble of the LaTeX file. It enables creating indexes. At the place where we want the index to appear, we have to put the command `\printindex.` Throughout the text, we can create index of a word or phrase using the command `\index{entry}`where `entry` is the index entry.

In order to generate the index we run the following command

```
$latex filename
$makeindex filename
$latex filename
```

Here while running `$ latex filename` command, it generates a file `filename.idx` that contains all the index entries and page numbers. The command `$makeindex filename` processes `filename.idx` and produces a file `filename.ind` that contains the LaTeX commands to format the index. Running the `$latex filename` command once again, reads the file `filename.ind` and produces the resulting output where it finds the `\printindex` command.

As an example, let us consider that in a running text the word Geometry appears and at that place we want to generate an index entry for ‘Geometry’ that says ‘see Euclid’. In order to do this, we need to write using a vertical bar as follows:

```
\index{Geometry|see{Euclid}}.
```



CHECK YOUR PROGRESS

Q 9: What package is required to generate index?

.....

.....

Q 10: What commands are used in the preamble to create index?

.....

.....

Q 11: What are the commands for compiling the files to generate an output pdf file with a list of index?

.....

.....



4.9 LET US SUM UP

- Different styles of lists can be easily prepared in latex using the environments listed in Sec. 4.3.
- Figures can be inserted in LaTeX using the figure environment as described in Sec. 4.5. There are options to handle figures using various optional keys.
- Figures and tables can be numbered and can be cited in the main text whenever required just by using the `\label{key}` and `\ref{key}`, as described in Sec. 4.6.
- In the font matter of a long document, a list of figures and a list of tables can easily be generated using the commands `\listoffigures` and `\listoftables`, respectively.
- Indexes can be easily generated using the package `makeidx`.



4.10 FURTHER READINGS

1) Knuth, D. E., & Bibby, D. (1984). *The Tex book (Vol. 3)*. Reading: Addison-Wesley.

2) Lamport, L. (1994). *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley.



4.11 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1:

```
\begin{enumerate}
\item BCA courses
   \begin{itemize}
   \item[(a)] Theory courses
      \begin{itemize}
      \item[–] Data structure
      \item[–] Object oriented programming
      \end{itemize}
   \item[(b)] Practical courses
      \begin{itemize}
      \item[–] C++
      \item[–] Java
      \end{itemize}
   \end{itemize}
\item MCA courses
\end{enumerate}
```

Ans to Q No 2: `\begin{description} .. \end{description}` environment creates a customizable list.

Ans to Q No 3: For writing poetry, `\begin{verse} … \end{verse}` environment is useful.

Ans to Q No 4:

```
\renewcommand{\ben}{\begin{enumerate}}
\renewcommand{\een}{\end{enumerate}}
\renewcommand{\bi}{\begin{itemize}}
\renewcommand{\ei}{\begin{itemize}}
\ben
\item BCA courses
```

```
\bi
\item[(a)] Theory courses
   \bi
   \item[–] Data structure
   \item[–] Object oriented programming
   \ei
\item[(b)] Practical courses
   \bi
   \item[–] C++
   \item[–] Java
   \ei
\ei
\item MCA courses
\een
```

Ans to Q No 5: The `\newcommand` is used as `\newcommand {\cmd} {defn}` where `\cmd`is the command name and`defn` is the text to be substituted for every occurrence of `\cmd`. `\newcommand{\a2}{\frac{1}{r^2}}` where \a2 is the new-command for\ frac{1}{r^2}.

Ans to Q No 6: We need to define the `\newcommand` or `\renewcommand` in the preamble of the LaTeX file.

Ans to Q No 7: We need to load either the command `\usepackage{graphics}`or `\usepackage{graphicx}` in order to insert an image file in LaTeX.

Ans to Q No 8:

```
\begin{figure}[h]
\centering\includegraphics[width=0.5\textwidth,height=0.8\tex
theight]{phi.eps}
\end{figure}
```

Ans to Q No 9: `makeidx` package is required to generate index.

Ans to Q No 10: The command `\makeindex` is required in the preamble of the LaTeX file to create index.

Ans to Q No 11: If the file name is xyz.tex, then following commands need to run

```
$latex  xyz
$makeindexxyz
$latex  xyz
```



4.12 MODEL QUESTIONS

Q 1: How can we create a numbered list and a bulleted list?

Q 2: Which list environment will be useful to create a customizable list of few items?

Q 3: How do you create a new-command of your own for the commands `\begin{description}` and `\end{description}`?

Q 4: Create new-commands for `\rightarrow`and `\leftarrow`.

Q 5: Write down the codes for including a picture named as pic.jpg in your LaTeX file.

Q 6: What are the available options for customizing figures within `\includegraphics` command?

Q 7: Explain briefly the use of `\label{key}` and `\ref{key}` in a LaTeX file.

Q 8: A table is labelled as `\label{tab:data}`. How can we cite the table in running text?

Q 9: Explain how we can refer an equation in LaTeX.

Q 10: What necessary modifications are required in order to create a table of contents as a front matter of an article?

Q 11: Write down the LaTeX commands that create a list of figures and a list of tables.

Q 12: What are the necessary commands for creating an index list of a document?

*** ***** ***

UNIT 5 : PACKAGES

UNIT STRUCTURE

5.1 Learning Objectives

5.2 Introduction

5.3 Packages

5.4 Geometry

5.5 Hyperref

5.6 The amsmath and amssymb packages

5.7 Algorithms and Algorithmic Packages

5.8 TikZ Package

5.9 Let Us Sum Up

5.10 Further Readings

5.11 Answers to Check Your Progress

5.12 Model Questions

5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- use geometry package for customizing layout of a document
- use hyperref for displaying live links in the pdf file
- use various ams packages
- write flowchart and algorithms using standard packages of LaTeX
- draw geometric figures using TikZ package

5.2 INTRODUCTION

In the previous unit we discussed about the main page layouts for producing documents of various styles. We have learnt how to create a specific layout involving sections, subsection, chapters, references, tables and figures. We have also learnt how to handle figures and tables in a LaTeX file.

In this unit we will learn about different packages. A wide variety of additional features are provided by a large number of available LaTeX

packages. Although there are many packages available in LaTeX, in this unit we shall describe some of the important packages that are frequently used in preparing documents of different styles. The packages that we are going to describe in this unit are `geometry, hyperref, amsmath, amssymb, algotithm2e,` and `TikZ.` In the next unit we will explore the concept of different classes like article class, book class, beamer class etc.

5.4 PACKAGES

A package generally comprises of a number of files that contain extra definitions and macros. The file names usually have the extension `.sty`. We can load a package with the `\usepackage` command which has to be loaded immediately after the `\documentclass` command in our input file. The command has the general form:

```
\usepackage[options]{package}
```

We can use one `\usepackage` command to load several packages by separating their names with commas. For example, if need to insert `graphics` and customization of page layout in our document, the packages `graphicx` and `geometry` provide the commands to do this. The beginning of our input file might look like:

```
\documentclass[11pt]{article}
\usepackage{graphicx, geometry}
```

The `\usepackage` command above instructs LaTeX to read the files `graphicx. sty` and `geometry.sty.`

5.4 GEOMETRY

The `geometry` package provides a flexible and easy interface to change page dimensions. We can easily change the layout of a page such as the paper size, margins, footnote, header, orientation, etc by changing the setting of the parameters. In order to use `geometry` package, we need to download `geometry.sty` and then to load it using the command `\usepackage{geomtery}.`

The page layout dimensions defined in the `geometry` package

contains a total body (printable area) and margins. The total body consists of body (text area) with an optional `header, footer, and marginal notes`. There are four margins: `right, left, top and bottom`. Each margin is measured from the corresponding edge of a paper. For instance, the right margin measure the distance between the right edge of the paper and that of the total body. Thus the left and top margins defined in geometry are different from the native dimensions `\leftmargin` and `\topmargin`.

There are many different ways of handling the page layout in the geometry package. Here we shall describe some commonly useful once. For instance, in order to create a document in a4paper with text area 6 inches in width and 8 inches in height, we need to include in the preamble:

```
\usepackage[a4paper, total={6in, 8in}]{geometry}
```

Here `a4paper` sets the paper size and the `total` parameter determines the size of the text area. Other page elements such as paper size, orientation and margins can also be changed as our requirements. There are two different ways to do this. For instance, if we want to create a document with legal paper size, landscape orientation and 1.2 inches margin all around, we can either write:

```
\usepackage[legalpaper,landscape,margin=1.2in]{geometry}
```

or, we can define them as

```
\usepackage{geometry}
\geometry{legalpaper, landscape, margin=1.2in}
```

The first parameter is the paper size where we set `legalpaper`. Other options are `a4paper, a5paper, b4paper, letterpaper, executivepaper`, etc.The second parameter is the orientation where we use `landscape`; the default value is `portrait`. The third parameter is the `margin` which we set as 1.2 inch.

As we see above, `margin` is a length parameter. All such parameters are described in the form: `parameter=value`, where the value is expressed in standard LaTeX units such as pt, mm, cm, in.

If we use

```
\usepackage[total={6in,8in},top=1.2in,left=0.9in,includefoot]
{geometry}
```

this will set the total allowable text area as 6 inches wide

and 8 inches high. The top margin on each page is 1.2 inches from the top edge of the page while the left margin is 0.9 inch from the left edge. This also includes footer with page number at the bottom of the text area.

In order to describe the total body, following options can be used.

- `hscale` is the ratio of width of total body to `\paperwidth`. For instance, `hscale=0.8` is equivalent to `width=0.8\paperwidth`. The default value is 0.7.
- `vscale` is the ratio of height of total body to `\paperheight`. For example, `vscale=0.9` is equivalent to `height=0.9\paperheight`. The default value is 0.7.
- `total` specifies width and height of total body. It is specifies either as `total={<width>,<height>}` or `total=<length>`.
- `textwidth` specifies `\textwidth`, the width of body (the text area): `textwidth=<length>`.
- `textheight` specifies `\textheight`, the height of body (the text area): `textheight=<length>`.

In the latest version (version 5) there are commands like `\newgeometry{..}` and `\restoregeometry` that allow us to change page dimensions in the middle of the document. Unlike \geometry in the preamble, `\newgeometry` command can be defined only after `\begin{document}`. It resets all the options ever specified except for the paper-size related options: `landscape`, `portrait`, and paper size options such as `a4paper`, `a5paper`, and so on. The command `\restoregeometry` restores the page layout specified in the preamble. It is important to note both the `\newgeometry` and `\restoregeometry` insert `\clearpage` where they are defined. Below we provide an example showing how to change layout in the middle of the document.

```
\usepackage[margin=4cm]{geometry}
\begin{document}
\newgeometry{left=3cm,right=1cm,bottom=0.1cm}
% Layout L1 will appear here
\restoregeometry
Layout L1 will be restored here
```

```
\newgeometry{margin=1cm,includefoot}
% Layout L2 will appear here
\end{document}
```

Here, the layout L1 specified with margin=4cm (left, right, top, and bottom margins are 4cm long) is changed to L2 with left=3cm, right=1cm and bottom=0.1cm. The layout L1 is restored with `\restoregeometry`.

We can define and reuse more different layouts in the document simply by using the commands `\savegeometry{<name>}` and `\loadgeometry{<name>}`. Below we show an example where these commands are used.

```
\usepackage[hmargin=3cm]{geometry}
\begin{document}
\newgeometry{left=3cm,right=1cm,bottom=0.1cm}
\savegeometry{L2}
\restoregeometry
\newgeometry{margin=1cm,includefoot}
\loadgeometry{L2}
\end{document}
```



CHECK YOUR PROGRESS

Q 1: How do we insert a package in a LaTex file?

.....

.....

Q 2: What is a .sty file?

.....

.....

Q 3: What are the margins used in Geometry package?

.....

.....

Q 4: How do we define paper size and orientation in a LaTeX file using Geometry package?

.....

.....

Q 5: Set a paper with allowable text area as 6.2 inches wide and 9 inches high. Set the top margin on each page as 1.0 inches from the top edge of the page.

.....

.....

5.5 HYPERREF

There is a package called `hyperref` which can be used to display live links in the pdf files. By loading `\usepackage{hyperref}` in the preamble, we can automatically get bookmarks corresponding to our sections, and hyperlinks from our table of contents and `\cite` and `\ref` references. This package has a number of options and we can effectively use them as our requirements. For instance, `colorlinks` makes the text of the link colored instead of framed with a box. It is also possible to choose the color of the links with the `linkcolor`, `urlcolor`, and `citecolor` options.

If we declare

```
\usepackage[colorlinks,linkcolor=blue,urlcolor=red,citecolor=green]{hyperref}
```

the links of any text in the pdf files appear blue whereas the url links appear red and cite colors green.



CHECK YOUR PROGRESS

Q 6: What is the use of `hyperref` package?

.....

.....

Q 7: How do we choose different colors for different links using `hyperref`?

.....

.....

5.6 THE AMSMATH AND AMSSYMB PACKAGES

The amsmath and amssymb packages were developed by American mathematical society (AMS). Previously, the amsmath package was named "amstex" that included some of the font-related features. The font-related features are now separated from amsmath and they are called amssymb and amsfonts packages. We can invoke them in the preamble as \usepackage{amsmath,amssymb,amsfonts}. They provide mathematical symbols and typeface changing commands with numerous options. There are several user-friendly environments within the amsmath package which we can use effectively as our requirements.

In Unit 2 we show several environments for displaying matrices. All these environments such as matrix, pmatrix, bmatrix,Bmatrix,vmatrix, Vmatrix, belong to amsmath package. In addition, the amsmath package has a number of environments for alignment. We can use align environment for creating multiline display equations instead of the eqnarray environment.

```
\begin{align*}
x^2+y^2 &< 1 \\ y &= \sqrt{1-
x^2}
\end{align*}
```

with

```
\begin{eqnarray*}
x^2+y^2 &<& 1 \\ y &=&
\sqrt{1-x^2}
\end{eqnarray*}
```

Compare

$$x^2 + y^2 < 1$$
$$y = \sqrt{1 - x^2}$$

with

$$x^2 + y^2 < 1$$
$$y = \sqrt{1 - x^2}$$

Here we can see the difference between the eqnarray and align environment although they produce the same thing. The difference is in the alignment points. For alignment in eqnarray environment, two ampersand (&) characters is used surrounding the part that needs to be aligned. The align environment, on the other hand, uses a single ampersand to mark the alignment point.

The package `align*` is the same as `align` but it aligns at a single place without automatic numbering of the formula. For alignment at several places, we can use the `alignat` environment or `alignat*`.

```
\begin{alignat*}{2}
F_0 &= 0 & \qquad F_1 &= 1 \\
F_2 &= 1 & \qquad F_3 &= 2 \\
F_4 &= 3 & \qquad F_5 &= 5
\end{alignat*}
```

The `split` environment can be used to split a large formula into multiple lines.

```
\[ \begin{split}
(x+y)^n &= \sum_{k=0}^{n}\binom{n}{k}
x^{k}y^{n-k} \\ &= x^n + nx^{n-1}y +
\frac{n(n-1)}{2}x^{n-2}y^2\\ &\quad +
\cdots + nxy^{n-1} + y^n\\
\end{split} \]
```

Some mathematical symbols are available in `amsmath`. However, there are symbols for which `amssymb` package is essential. For instance, `\mathbb`, `\mathfrak`, `\boldsymbol`, `\text` are not available in `amsmath`. They only work if we load `amssymb`.

5.7 ALGORITHMS AND ALGORITHMIC PACKAGES

To create algorithms in LaTeX, one can use `algorithm2e`, `algorithmic` or Listings environment. The `algorithm2e` package allows typesetting algorithms with a relatively large number of customization in comparison to algorithmic package. Here we shall describe how `algorithm2e` package is used to write algorithms in LaTeX.

To enable algorithm environment, we need to write `\usepackage{algorithm2e}`in the preamble of the `.tex` file which will load `algorithm2e` package. The algorithm has to be written in between `\begin{algorithm}`and `\end{algorithm}`.Below we give an example where we have used the `algorithm2e` package.

```
\documentclass[11pt,a4paper]{article}
\usepackage[ruled,vlined,linesnumbered]{algorithm2e}
\begin{algorithm}[H]
\KwData{Enter the data}
\KwResult{how to write algorithms with \LaTeX2e}
initialization\;
\While{this inequality is satisfied}
{
readdata\;
\eIf{understand}{
Look at the next condition\;
The new condition becomes this one\;
}{
go back to the beginning of conditions\;
}
}
\caption{How to write algorithms}
\end{algorithm}
```

Algorithm 1: How to write algorithms

Data: Enter the Data

Results: How to write algorithms with LaTeX2e

```
1 Initialization;
2 While this inequality is satisfieddo
3 read data
4 if understand then
5 look at the next condition
6 the new condition becomes this one
7 else
8 go back to the beginning of conditions
```

In this example, while declaring the package in preamble, we have used various options which give us more nice formatting of algorithm. The keywords of the options are described below.

- `ruled`: It is used to make caption as a header name of the

algorithm and to display it in the beginning of algorithm.

- `vlined`: It is used to draw vertical line from starting position of a state to the end of that state.
- `linesnumbered`: This is used to display algorithm with line number.
- `\KwData{this text}`: This is a predefined keyword variable in algorithm. It shows "enter the data" as data in algorithm.
- `\KwResult{how to write algorithm with \LaTeX2e}`: This is also predefined keyword variable in algorithm. It will show "how to write algorithm with LaTeX2e" as result in algorithm.
- `\While{<need to satisfy>}{<code>}`: The while command takes two arguments. First one represents the code which needs to satisfy by while loop and second argument takes execution code for whole loop. Similarly other codes will work. For example `\If{<need tosatisfy>}{<code>}`,`\For{<need to satisfy>}{<code>}`.
- `\eIf{<need to satisfy for if>}{ <if-code> }{<else-code>}`

It is also possible to define own keyword text using `\SetKwInOut` command. We can also make algorithm such that none of the line will contain semicolon using `\DontPrintSemicolon`.

- **The Algorthmic Package**

  The algorithmic package is used to write algorithms. The pseudo codes are written in between

  ```
  \begin{algorithmic}
  \end{algorithmic}
  ```

  There is an option to give an optional argument of a positive integer to the command `\begin{algorithmic}`. It indicates where to appear the line numbering in multiple of that number. For instance, `\begin{algorithmic}[10]` will create an algorithmic environment and number every tenth line. Let us take the following example.

  ```
  \begin{algorithmic}
  \STATE $i\gets 100$
  \IF {$i\geq 10$}
  ```

```
\STATE $i\gets i-1$
\ELSE
\IF {$i\leq5$}
\STATE $i\gets i+2$
\ENDIF
\ENDIF
\end{algorithmic}
This will produce the following results
```

$i \leftarrow 100$
if $i \geq 10$ **then**
$i \leftarrow i - 1$
else
if $i \leq 5$ **then**
$i \leftarrow i + 2$
end if

end if



CHECK YOUR PROGRESS

Q 8: What is an `algorithm2e` package?

.....

.....

Q 9: What are the options available with `algorithm2e` package? Straight lines are represented by are some simple examples.

.....

.....

5.8 TIKZ PACKAGE

Here we provide a brief introduction showing how to draw in LaTeX using the **TikZ** package. TikZ can produce portable graphics in both PDF and postscript formats. In order to use TikZ we must include the package as `\usepackage{tikz}` somewhere in the preamble. Inside the document we can now create a new picture using

`\begin{tikzpicture}` and `\end{tikzpicture}`. Everything in between these two commands is interpreted as a portable graphics image. There is a series of commands that are used in various purposes in TikZ. These are `\draw`, `\fill`, `\filldraw`, `\pattern`, `\shade`, `\shadedraw`, `\clip`, etc.

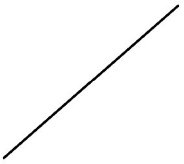
There is a `\draw` command to specify the coordinates of a line which are in centimeters by default.The `\draw` command also specifies to draw a straight line.

Straight lines are represented by some simple examples.

```
\documentclass{article}
\usepackage{tikz}
\begin{tikzpicture}
\draw (1,0) — (0,0) — (0,1);
\end{tikzpicture}
\end{document}
```



```
\begin{tikzpicture}
\draw (0,0) — (2,4);
\end{tikzpicture}
```



Two points can be connected by straight lines that are only horizontal and vertical.

There are two options:

- (p) -| (q) connect points (p) and (q) using horizontal and vertical line traveling horizontal first and then vertical
- (p) |- (q) connect points (p) and (q) using horizontal and vertical line traveing vertical first and then horizontal.

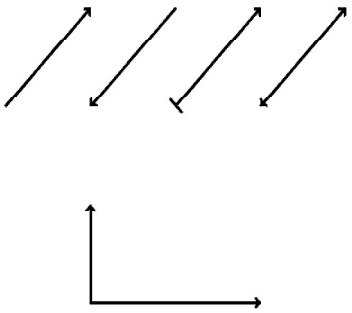
```
\begin{tikzpicture}
\draw (1,0) -| (0,1);
\draw (2,0) |- (0,2);
\end{tikzpicture}
```



The draw command accepts several different options that change

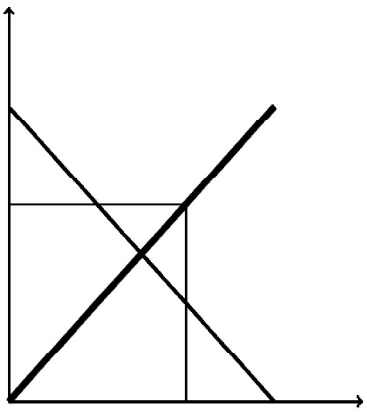
the appearance of the line endings. For example, arrows of different style can be easily drawn as follows:

```
\begin{tikzpicture}
      \draw [->] (0,2) – (1,3);
      \draw [<-] (1,2) – (2,3);
      \draw [|->] (2,2) – (3,3);
\draw [<->] (3,2) – (4,3);
\draw [<->] (1,1) – (1,0) – (3,0);
\end{tikzpicture}
```



The thickness of every single line can be changed according to our requirements. We can use the operation `ultrathin, very thin, thin, semi thick, thick, very thick` and `ultrathick.`

```
\begin{tikzpicture}
\draw [<->] (0,4) – (0,0)
\draw [thick] (0,3) –
\draw [ultrathick] (0,0)
\draw [ultrathin] (2,0)
\end{tikzpicture}
```



We can use color in drawing line. For instance

```
\begin{tikzpicture}
\draw[red,thick] (-2,0)
\end{tikzpicture}
```


We see here a red thick line whose end points are (–2,0) and (2,0). There are several predefined colours, but you can also use your own by adding `\definecolor{mycolor}{rgb}{0,0.6,0.5}` in the preamble of your documents.



```
\documentclass{article}
\usepackage{tikz}
\definecolor{mycolor}{rgb}{0,0.6,0.5}
```

```
\begin{document}
\begin{tikzpicture}
\draw [thick, black] (0,1)
\draw [thick, mycolor] (0, 0.5)
\draw [thick, blue] (0,0)
\end{tikzpicture}
\end{document}
```

The `\filldraw` command is used to draw elements and fill them with some specific colour.

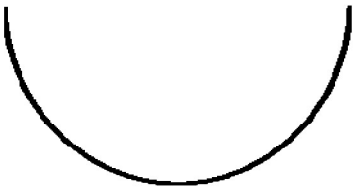
```
\begin{tikzpicture}
\draw[red,thick] (-2,0)
\filldraw [blue] (0,0) circle (2pt);
\end{tikzpicture}
```



We see here a small blue circle centered at `(0,0)` with radius `2pt`. We can also draw curves using `control` operation. For instance,

```
\begin{tikzpicture}
\draw (-2,-2) ..controls (0,0) .. (2,
\end{tikzpicture}
```

This draws a curve with four points: `(-2,2)` and `(2,2)` are the end points, `(-1,0)` and `(1,0)` are the control points determine how curve it is.



User defined path can be created using the **to** operation. We can also use the options in and out . For instance `\draw(0,0) to[out=90,in=180] (2,2)` creates a path that leaves at an angle of 90 degree at the first coordinate and arrive at an angle of 180 degree at the second cooordinate.

```
\begin{tikzpicture}
\draw (0,0) to (3,2);
\draw (0,0) to[out=90,in=180] (
\draw (0,0) to[bend right] (
\end{tikzpicture}
```



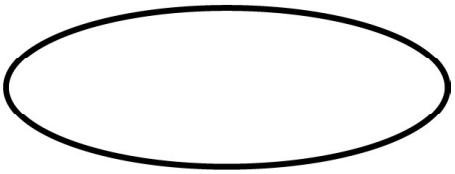
We can easily draw shapes like rectangle, circle, stars, etc.

`\draw [blue] (0,0) rectangle (2,1)` This will draw a rectangle where from which the pencil starts drawing and diagonally opposite point.



To draw an ellipse, we have to define a center point and the horizontal and vertical radii. For instance,

`\draw [blue] (0,0) ellipse (2.5 and 1.5)`



This creates an ellipse whose center is at `(0,0)`, the horizontal radius `2.5 and` the vertical radius `1.5`.

We can also use mixture color gradient as follows.

```
\shade[top color=blue, bottom color=red](0,0) rectangle (2,1)
```



This will create a rectangle filled with a mixture color gradient with blue as the top color and red as the bottom color.



CHECK YOUR PROGRESS

Q 10: List the commands that are used in various purposes in TikZ.

.....

.....

Q 11: Write down the LaTeX commands for drawing an arrow from the point (0,0) to (5,10).

.....

.....

Q 12: What are the options available in TikZ for changing the line thickness?

.....

.....



5.9 LET US SUM UP

- LaTeX packages provide additional features that are useful for creating documents of different styles and forms.
- The `Geometry` package provides a flexible and easy interface to change page dimensions.
- The `hyperref` package is used to display live links in the pdf files.
- The `amsmath` and `amssymb`packages provide mathematical symbols and typeface changing commands with numerous options.
- The `algorithm2e` and `algorithmic`packages are used for typesetting algorithms.
- The TikZ package is used to produce portable graphics in both PDF and postscript formats.



5.10 FURTHER READING

1) Lamport, L. (1994). *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley.

2) Goossens, M., Rahtz, S. P., Rahtz, S., & Mittelbach, F. (1997). *The LATEX graphics companion: illustrating documents with TEX and PostScript (Vol. 1)*. Addison-Wesley Professional.

3) Kottwitz, S. (2011). *LaTeX beginner's guide*. Packt Publishing Ltd.



5.11 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Using the command line `\usepackage{package name}` immediately after declaring `\documentclass`command in our input file.

Ans to Q No 2: A `.sty` file is usually a specific package file used in LaTeX.

Ans to Q No 3: There are four margins: `right`, `left`, `top` and `bottom`.

Ans to Q No 4: The paper size and orientation in a LaTeX file can be declared in the \usepackage command as:

```
\usepackage[legalpaper,landscape]{geometry}
```

where `legalpaper` is the paper size and `landscape` is the orientation.

Ans to Q No 5:

```
\usepackage[total={6.2in,9.0in},
top=1.0in]{geometry}
```

Ans to Q No 6: The `hyperref` package is used in LaTeX to create live links in the output pdf files.

Ans to Q No 7: Different colors for different types of links can be given within the `hyperref` package by using the `linkcolor`, `urlcolor`, and `citecolor` options. For instance, if we declare `usepackage[colorlinks,linkcolor=blue, urlcolor=red, citecolor= green]{hyperref}` the links of any text in the pdf files appear blue whereas the url links appear red and cite colors as green.

Ans to Q No 8: The `algorithm2e` package allows typesetting algorithms. It has a relatively large number of customization in comparison to `algorithmic` package

Ans to Q No 9: `ruled`, `vlined`, `linesnumbered`.

Ans to Q No 10: `\draw`, `\fill`, `\filldraw`, `\pattern`, `\shade`, `\shadedraw`, `\clip`.

Ans to Q No 11:

```
\begin{tikzpicture}\draw [->] (0,0) -- (5,10); \end{tikzpicture}
```

Ans to Q No 12: `ultrathin`, `very thin`, `thin`, `semi thick`, `thick`, `very thick` and `ultrathick`.



5.12 MODEL QUESTIONS

Q 1: What is the default field value for paper orientation in Geometry package? How do you change the paper orientation from portrait to landscape?

Q 2: Explain what will happen if we declare `hscale=0.9` and `vscale =0.95`?

Q 3: Explain how we use `\savegeometry` and `\restoregeometry` commands to change page dimensions in the middle of a document.

Q 4: List the various pre-defined key variables used for formatting an algorithm in`algorithm2e` package.

Q 5: Write a simple algorithm using algorithmic package.

Q 6: Using TikZ create a path that leaves at an angle of 90 degree at the first coordinate (0,0) and arrive at an angle of 180 degree at the second cooordinate (5,3).

Q 7: Using TikZ create an ellipse whose center is at (0,0), the horizontal radius 6.2 and the vertical radius 2.6.

Q 8: Write the commands to convert an arrow of normal thickness to a very thin arrow.

*** ***** ***

UNIT 6 : CLASSES

UNIT STRUCTURE

6.1 Learning Objectives

6.2 Introduction

6.3 Classes

6.4 Article Class

6.5 Book Class

6.6 Report Class

6.7 Beamer Class

6.8 Slides

6.9 Letter Class

6.10 Let Us Sum Up

6.11 Further Readings

6.12 Answers to Check Your Progress

6.13 Model Questions

6.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- use different document class in writing a document
- describe the basic structure of all the important classes
- create a standard article or book chapters of your own
- create slides using beamer class
- write letters in LaTeX

6.2 INTRODUCTION

In the previous unit we discussed about packages like geometry, hyperref, amsmath, amssymb, algotithm2e, and TikZ. We often use widely different types of documents like letter, short article, project report, book chapter, and thesis. Creating such different formats is time consuming.

In LaTeX, there are various options to produce different types of document just by using those predefined options that are available in

LaTeX. Some of the more common document classes that can be easily created in LaTeX are article, report, book, letter, thesis, slides, proceedings, etc. Each of them is represented by different packages in LaTeX. While such a package is defined in the preamble section of the document, it will automatically take that format and create the document that we actually want to see. Further, the existing format can easily be changed to another format by changing the document class. In this unit we shall discuss some commonly used document classes that are available as a separate package in LaTeX which will enable you to create output files of your own. In the next unit we will explore the different applications of LaTeX.

6.3 CLASSES

The document class determines the overall layout of the document. There are several standard document classes distributed with LaTeX. The most frequently used classes are article, report, book, letter, and beamer class. Each of these classes has special commands suitable for producing specific sections. For example, the article, report, and book classes include a variety of commands to format section headings such as `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, etc. These classes also possess commands to produce a title page and a table of contents. The article class is designed for short articles and does not have chapters. The letter class provides special commands to produce the salutation, address, and closing. As already described in the previous units, a document class can be incorporated in a document by using the command `\documentclass [option,option,option]{class}`. We can insert specific options within the square bracket with multiple options separated by commas. The standard class options include font sizes (`10pt, 11pt,12pt`), paper sizes (`a4paper, letterpaper`), `titlepage`, number of columns `(onecolumn, twocolumn)`, print style `(oneside,twoside)`, formula style `(leqno, fleqn)` etc. We shall now go through different types of classes in detail.

6.4 ARTICLE CLASS

The article class is used for short documents including journal articles and short reports. As described above, in between the `\documentclass` and the name of the document style, one may place a list of options. These options are separated by commas and the list of options is enclosed in square brackets. For instance, one can define `\documentclass[a4paper,12pt,twocolumn]{article}` The options available include the following:

- **Font sizes:**
  - `11pt`: It specifies a size of type known as eleven-point, which is ten percent larger than the ten-point type normally used.
  - `12pt`: It specifies a twelve-point type size, which is twenty percent larger than tenpoint.

  If no option is specified, `10pt` is assumed by default.
- **Paper sizes:**
  - `a4paper`: This ensures that the page is appropriately positioned on A4 size paper.
  - `a5paper`: This ensures that the page is appropriately positioned on A5 size paper.
  - `b5paper`: This ensures that the page is appropriately positioned on B5 size paper.
  - `legalpaper`: This ensures that the page is appropriately positioned on legal size paper.

  The default size is `letterpaper`.
- **Number of columns:**
  - `onecolumn`: It instructs LaTeX to produce an one-column output.
  - `twocolumn`: It instructs LaTeX to typeset the document in two columns.
- **Print style:**
  - `oneside`: It instructs LaTeX to generate a single-sided output. The classes article and report are single sided by default.
  - `twoside`: It instructs LaTeX to generate a double-sided output.

The book class is double-sided by default.

Note that this option concerns the style of the document only. The option two-side does not tell the printer you use that it should actually make a two-sided printout.

- **Formula style:**
  - `Fleqn`: It displays formulas left-aligned instead of entered.
  - `leqno`: It places the numbering of formulae on the left-hand side instead of right.

Pages will be automatically numbered at the bottom of the page, unless we specify otherwise. This can be done using the `\pagestyle` command. This command should come after the `\documentclass` command and before the `\begin{document}` command. This command has the syntax `\pagestyle{option}`, where the option is one of the following:

- **`plain`:** The page number is at the foot of the page. This is the default page style for the article and report document styles.
- **`empty`:** No page number is printed.
- **`headings`:** The page number (and any other information determined by the document style) is put at the top of the page.
- **`myheadings`:** Similar to the headings `pagestyle`, except that the material to go at the top of the page is determined by `\markboth` and `\markright` commands.

For example, the input file

```
\documentclass[12pt, a4paper]{article}
\pagestyle{empty}
\begin{document}
The main body of the document is placed here.
\end{document}
```

produces a document without page numbers. As we mentioned, it uses the 12-point font size.

The top part of an article contains title, authors, and date. An example is shown below.

```
\title{The Mythical History of Dragons}
```

```
\author{Wild Frank\thanks{Research supportedby the National science foundation under grant number~93569.} \\Earth Science Department\\University of Waterloo\\200 University Ave W, Waterloo, ON N2L 3G1, Canada\\\ texttt{gbhamilton@uwaterloo.ca} \date{July 20, 2018} \maketitle
```

There are certain rules that need to be remembered while preparing the front matter (title page information). These rules are:

- If the title is only one line long, there should not be any `\\` in the argument of the `\title` command; otherwise, the lines of the separate title should be separated with `\\`.
- The lines of the address should be separated with `\\`. There should not be any `\\` at the end of the last line.
- A footnote at the bottom of the first page can be placed by writing `\thanks`. It can be avoided if not needed. In the above example, we write `\thanks{Research supported by the National science foundation under grant number~93569.}` The text that is written within the curly bracket of `\thanks` command will be displayed as a footnote in the corresponding page of the output file.
- The `\author` command contains all the information (name, address, support) about all the authors. Multiple authors are separated by `\and`.
- The `\title` command is the only compulsory command. The others are optional. A particular date, say March 14, 2018 can be displayed by typing `\date{March14, 2018}`. The current date will be shown if `\date` command is not given.

The top matters can be printed using the command `\maketitle` just after declaring `\begin{document}`. The `\maketitle` command prints the title, the authors, and date in the format enclosed in the `.tex file`.

The rest of the body of the article can be arranged in sections and subsections as follows.

\section{Introduction} \label{sec:intro}
A dragon is a large legendary creature that appears in the folklore of many cultures around the world. In most of the cultures, dragons are depicted as winged, horned, four-footed and capable of breathing fire while, in most of the eastern cultures, they are usually depicted as wingless, fourlegged, serpentine creatures with above-average intelligence.

The first mythological description of Dragon-like creatures appear in the art and literature of ancient Mesopotamia. In the classic book An Instinct for Dragons (2000) anthropologist David Jones proposed a hypothesis that humans, just like monkeys, have inherited instinctive reactions to snakes, big cats, and birds of prey. He cites a study which found that approximately 39 people in a hundred are afraid of snakes and notes that fear of snakes is especially prominent in children, even in areas where snakes are rare. The earliest attested dragons all resemble snakes or bear snakelike attributes. Jones therefore concludes that the reason why dragons appear in nearly all cultures is because of humans' innate fear of snakes and other animals that were major predators of humans' primate ancestors.

\subsection{Egyptian mythology }In Egyptian mythology, Apep is a giant serpent who resides in the theDuat, the Egyptian Underworld. The Bremner-Rhind papyrus, written in around 310 BC, preserves an account of a much older Egyptian tradition that the setting of the sun is caused by Ra

descending to the Duat to battle Apep. In some accounts, Apep is as long as...



The Mythical History of Dragons

Wild Frank*
Earth Science Department
University of Waterloo
200 University Ave W, Waterloo, ON N2L 3G1, Canada
gbhamilton@uwaterloo.ca

July 20, 2018

1 Introduction

A dragon is a large legendary creature that appears in the folklore of many cultures around the world. In most of the cultures, dragons are depicted as winged, horned, four-footed and capable of breathing fire while, in most of the eastern cultures ,they are usually depicted as wingless, four-legged, serpentine creatures with above-average intelligence. The first mythological description of Dragon-like creatures appear in the art and literature of ancient Mesopotamia. In the classic book An Instinct for Dragons (2000) anthropologist David Jones proposed a hypothesis that humans, just like monkeys, have inherited instinctive reactions to snakes, big cats, and birds of prey. He cites a study which found that approximately 39 people in a hundred are afraid of snakes and notes that fear of snakes is especially prominent in children, even in areas where snakes are rare. The earliest attested dragons all resemble snakes or bear snakelike attributes. Jones therefore concludes that the reason why dragons appear in nearly all cultures is because of humans' innate fear of snakes and other animals that were major predators of humans' primate ancestors.

1.1 Egyptian mythology

In Egyptian mythology, Apep is a giant serpent who resides in the the Duat, the Egyptian Underworld. The Bremner-Rhind papyrus, written in around 310 BC, preserves an account of a much older Egyptian tradition that the setting of the sun is caused by Ra descending

*Research supportedby the National science foundation under grant number 93569.

1

In the above example, we see that a section, named as “Introduction” is entered as

```
\section{Introduction} \label{sec:intro}
```

Here, introduction is the title of the section and `sec:intro` is used as the label of the section number. The same label should be used in order to refer this section whenever required. In order to refer this section, we have to insert the label within `\ref{}` command as `\ref{sec:intro}`. The number of a section is automatically assigned by LaTeX and the same number will be displayed when we cite the label using `\ref{}` command in the text.

Sections have subsections, and sub-sections have subsubsections, followed by paragraphs and subparagraphs. The corresponding commands are

`\subsection` `\subsubsection` `\paragraph` `\subparagraph`

- **Spacing and paragraphs**

In LaTeX, whitespace characters such as blank or Tab are treated uniformly as “space”. Several consecutive whitespace characters are treated as one single space. Similarly, several empty lines between two lines of text are treated as the one empty line. An empty line between two lines of text defines the end of a paragraph. Otherwise, one can begin a new paragraph by inserting the command ***\par.***

There are several ways to introduce line breaks. The command `\newline` breaks the line at the point of the command. Instead of `\newline` command, one can use `\\` also. The command `\break` breaks the line without filling the current line.

Horizontal and vertical spaces of arbitrary length can be inserted with the commands `\hspace{}` and `\vspace{}`, respectively. For instance, `\hspace{1cm}` inserts a horizontal space of length 1 cm; `\vspace{7mm}` inserts a vertical space of length 7 mm.

Paragraphs in LaTeX are fully justified, i.e. flush with both the left and right margins. This justification of a paragraph can be altered by using following three environments:
`center, flushleft` and `flushright` as follows.
`\begin{center}` `\end{center}`
`\centering\begin{flushright}` `\end{flush-right}.`
`\begin{flushleft}` `\end{flushleft}`

The command `\noindent` creates a non-indented paragraph. There exists a number of other commands which are used to set different lengths as follows:

Length	Description
`\baselineskip`	Vertical distance between lines in a paragraph
`\columnsep`	Distance between columns
`\columnwidth`	Width of a column
`\paper-width`	Width of the page
`\paperheight`	Height of the page

\parindent	Paragraph indention
\parskip	Vertical space between paragraphs
\tabcolsep	Separation between columns in a table
\textheight	Height of the text area in the page
\textwidth	Width of the text area in the page
\topmargin	Length of the top margin



CHECK YOUR PROGRESS

Q 1: Name the different types of document class that are distributed with the LaTeX package?

.....

.....

Q 2: What is the command line for inserting a document class in a latex file?

.....

.....

Q 3: What are the different options that can be defined in declaring command line of the document class?

.....

.....

Q 4: Write the command for preparing an article with 12 point font-size and legal paper-size.

.....

.....

Q 5: What are the fields that we can fill in the title page of an article? How to include the affiliation of the author in the title page of the article?

.....

.....

Q 6: What is the command for printing the title page?

.....

.....

Q 7: How can we create a footnote at the bottom of the title page?

.....

.....

6.5 BOOK CLASS

The book class is used for preparing a full-length book. It includes font matters such as a preface, a table of contents, a list of figures, a list of tables, and back matters such as an appendix and a list of index. One of the unique features of the book class is that it has a `\chapter` command for beginning a new chapter. Others (for instance, section, subsections) are the same as in `article` class. The command for including book class in the preamble is the same as for article, only the article needs to be replaced by book as follows.

```
\documentclass[12pt,a4paper]{book}
```

If we declare the commands `\listoffigures` and `\listoftables`, they generate a list of all the figures and tables respectively in the font matter. Further, we can generate a list of index as back matter of the book. It is relatively easy to generate an index by using LaTeX commands combined with the support program `MakeIndex`. In order to do this, we have to load the package `makeidx` with the command: `\usepackage{makeidx}` and place the command `\makeidx` in the preamble. The place at which we want the index to appear, we shall put the command `\printindex`. If we want any specific word or phrase of the book to be displayed in the index, we need to write that particular word or phrase using the command `\index{entry}`.

The front matter of a standard book includes preface and table of contents, the main matter is the body of the text and the back matter comprises bibliography and index.

The front matter has roman page numbering and suppresses the numbering of chapters. Back matter also has unnumbered chapters. The page number is reset for the main matter.

The table of contents is normally generated and printed after the title page and abstract with the command `\tableofcontents`. All entries

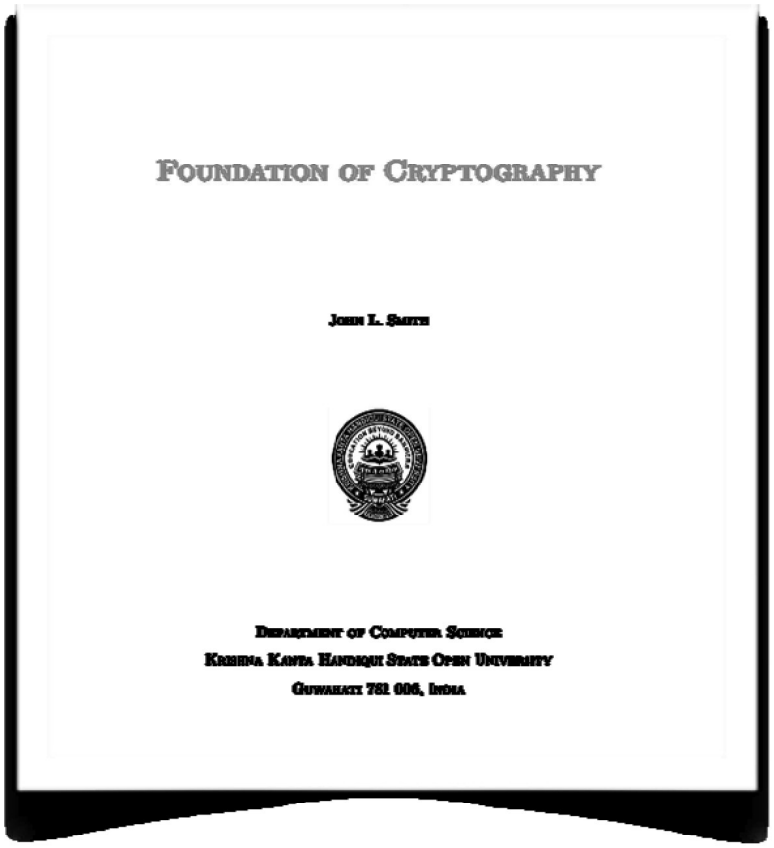
are created automatically, based on the sectioning commands such as \chapter{}, \section{}, \subsection{}, \subsubsection{}. If we do not want to include any chapter or section in the table of contents, the corresponding command will include a * mark such as \chapter*{}, \section*{}, \subsection*{}, \subsubsection*{}. The * mark behind the command produces an unnumbered version which will not be included in the table of contents.

One important difference between the book class and the report or article class is that the book class does not allow the abstract environment. The commands \begin{abstract} - - -\end{abstract} is available only for the document class report and article.

The simplest book template is shown below:

```
\documentclass[12pt]{book}
\usepackage{graphicx,amssymb,amsmath,pst-all,makeidx}
\makeindex
\begin{document}
\baselineskip=1.3\normalbaselineskip
\pagenumbering{roman}
\thispagestyle{empty}
\begin{center} \Huge{\sc\cyan Foundation of
Cryptography}\ end{center}
\vskip 3.0cm
\begin{center} \sc John L. Smith \end{center}
\vskip 2.0cm
\begin{center}
\includegraphics[width=3cm,height=3.5cm]{logo}\end{center}
\vskip 2.0cm
\begin{center}{\sc Department of Computer Science\\
Krishna KantaHandiqui State Open University\\Guwahati
781 006, India}\ end{center}
\newpage\thispagestyle{empty}
\chapter*{\hfill Preface\hfill\hfill}
\markright{\hspace{6cm} Preface}
\begin{verse}\textit{“Nature loves hiding”}\\\hfill—P.
W. Anderson.\end{verse}
```

```
%%%%%%% Write the preface here%%%%%%%
\newpage
\markright{}
\tableofcontents
\listoffigures
\pagenumbering{arabic}
\chapter{Introduction}
%%%% Write the Introduction chapter here
\section{An Overview on the Origin of Cryptography}
\subsection{Ancient Ciphers}
\subsection{Modern Ciphers}
\newpage
\leftmark\printindex
\newpage
\markright{Bibliography}
\bibliography{crypto}
\bibliographystyle{plain}
\end{document}
```





CHECK YOUR PROGRESS

Q 8: How do we create a chapter in LaTeX?

.....

.....

Q 9: Write the LaTeX commands for creating the first page of a book entitled "The Mythological History of Dragons" written by Wild Frank, University of Waterloo in August 2019.

.....

.....

Q 10: How do we create a Roman page number?

.....

.....

Q 11: What is the command for creating the first page of a book with-out a page number?

.....

.....

6.6 REPORT CLASS

The report class can be declared using the command:

```
\documentclass[11pt,a4paper]{report}
```

Before beginning the writing of the main text, we can prepare the first page separately so that we can design it independently and needn't have to compile the whole file each time. Once we prepare it separately, we can simply include it within the existing big document we are writing. For instance, let us take an example of a project report on "Cryptography and Data Hiding", prepared by Albert Smith and guided by Dr. K. K. Handique. We can prepare the title page by creating a separate small file, named `title_page.tex`. It will contain the following commands:

```
\begin{titlepage}
\begin{center}
\includegraphics[width=0.15\textwidth]{logo.jpg}\\[1cm]
\textsc{\LARGE Krishna KantaHandiqui State Open
```

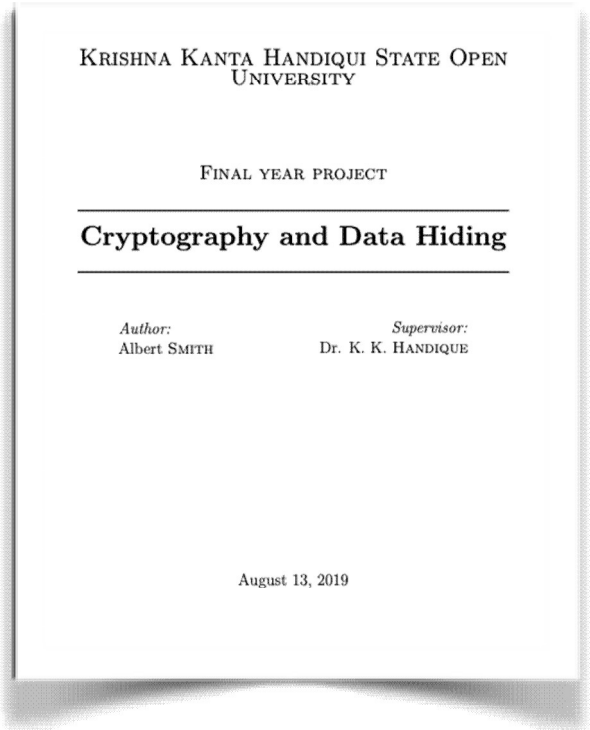
```
Universi-ty}\\[2.5cm]
\textsc{\Large Final year project}\\[0.5cm]
% Title
\HRule \\[0.4cm]
{ \huge \bfseries Cryptography and Data Hiding}\\[0.4cm]
\HRule \\[1.5cm]
% Author and supervisor
\begin{minipage}{0.4\textwidth}
\begin{flushleft} \large
\emph{Author:}\\
Albert \textsc{Smith}
\end{flushleft}
\end{minipage}
\begin{minipage}{0.4\textwidth}
\begin{flushright} \large
\emph{Supervisor:} \\
Dr. K. K. \textsc{Handique}
\end{flushright}
\end{minipage}
\vfill

% Bottom of the page
{\large \today}
\end{center}
\end{titlepage}
```

This LaTeX file, named as `title_page.tex` is now included in the main file, named as `project.tex` as follows:

```
\documentclass[12pt, a4paper]{report}
\usepackage{graphicx}
\newcommand{\HRule}{\rule{\linewidth}{0.5mm}}
\begin{document}
\input{title_page.tex}
\end{document}
```

Once we compile this file, it will generate the first page of the output file, named `project.pdf`, which looks like:


KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

FINAL YEAR PROJECT

Cryptography and Data Hiding

Author: Albert SMITH

Supervisor: Dr. K. K. HANDIQUE

August 13, 2019

As we see in this example, the title_page.tex file is created within the environment:

`\begin{titlepage} . . . .\end{titlepage}`

Following are the important things that we need to know in order to create such a title page for a report.

- **Alignment:** As shown above, we have used `\begin{center}. . .\end{center}` to center some text. We have also used `\begin{flushleft}. . . \end{flushleft}` environment for left-alignment of the text and `\begin{flushright}. . . \begin{flushright}` for rightright-alignment. Further, we can manage texts as we like using `\begin{minipage}…\end{minipage}` environment that create a small minipage. As shown above, within the mintage, we have used different alignments.
- **Images:** We have included the logo of Krishna Kanta Handiqui State Open University using the command: `\includegraphics[width=0.15\textwidth]{logo.jpg}.`

As we usually do, here we have not used any `\begin{figure}` environment simply because we do not want the figure to be floating; we want it where we placed it.

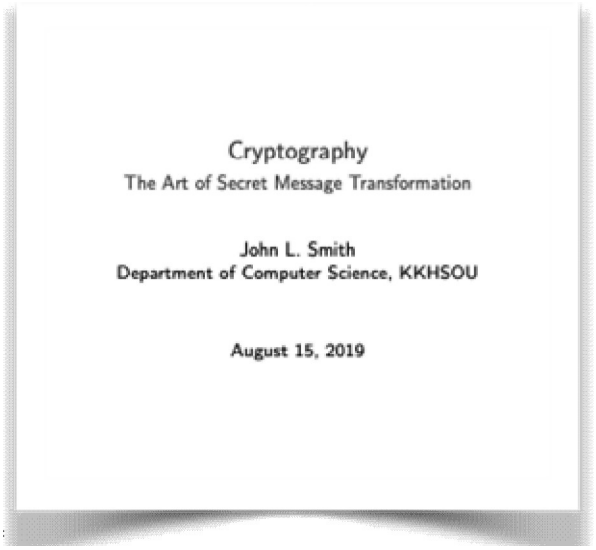
- **New lines:** As usual, we can force the start of a new line by using new-line command `\\`. If we want to add more vertical place, we don't need to use several newline commands. As we have done here, by writing `\\[1cm]` we can create a new line leaving 1 cm as an empty space.
- **Date:** We can insert the date of the current day using the command `\today`.
- **Filling the page:** The command `\vfill` is used to add empty space until the page is full Just like in the document class article, the commands `\begin{abstract} - - -\end{abstract}` is also available for the document class `report`. This environment is used to write the abstract. By default, LaTeX will use the word "Abstract" as a title for the abstract. However, we can change it if we want. For instance, if we want to write "Summary" instead of Abstract, we can do it by adding the following line in the preamble: `\renewcommand{\abstractname}{Summary}`

6.7 BEAMER CLASS

An excellent way of making a presentation using LaTeX is the implementation of `beamer` class. The presentation slides can be prepared just like a normal LaTeX document and the corresponding output after compilation is a pdf slide-show similar to Powerpoint, with the advantage of superior mathematical documentation capabilities. The very first step for creating such a presentation is to declare `\documentclass{beamer}` in the preamble (before the `\begin{document}`). With just this command we can now create a very basic looking presentation. In order to create each slide we use the command:

```
\begin{frame}
Insert slide contents here
\end{frame}
```

A simple example is given below.




```
\documentclass{beamer}
\usepackage{graphicx}
\title{Cryptography}
\subtitle{The Art of Secret Message Transformation}
\author{John L. Smith\\ Department of Computer Science, KKHSOU}
\end{figure}
\date{\today}
\begin{document}
\frame{\titlepage}
\section[Outline]{}
\frame{\tableofcontents}
\section{Introduction}
\subsection{Overview of Cryptography}
\frame
{
\frametitle{Caeser Cipher}
\begin{itemize}
\item<1-> Julius Caeser (49 BC - 44 BC)\includegraphics[width=0.3\textwidth]{Julius}
\item<2-> \includegraphics[width=0.5\textwidth]{Caeser}
\item<3->Caeser Cipher \includegraphics[width=0.5\textwidth]{Caeser_cipher}
\end{itemize}
}
\end{document}
```

As shown in the above example, titles and subtitles can be added

to the frame using the commands

```
\frametitle{Title}
\framesubtitle{Subtitle }
```

just after the `\begin{frame}` command. The default font in the beamer is "Sans Serif." To change the fonts, the command `\usefonttheme{font theme name}` needs to be added in the preamble.

Some common font themes that Beamer recognizes are:

`professionalfonts,structureitalicserif,structurebold, structuresmallcapsserif`

To change the colour of the background, use the command, `\setbeamercolor{normal text}{bg=colour}` in the preamble. If we define our own colour, the commands will be similar to

```
\setbeamercolor{normal text}{bg=MyGreen}
\definecolor{MyGreen}{RGB}{60, 140, 60}
```

Using the command, `\begin{frame}[plain]` will result in the frame having none of the loaded backgrounds or decorations. To enhance our presentations, there are different themes available in Beamer that drastically changes the visual presentation.

We can use the command, `\usetheme{theme name}` in the preamble. Some standard themes that Beamer recognizes are:

```
AnnArbor    Berkeley   Berlin     Boadilla   Copenhagen
Dramstadt   Dresden    Frankfurt  Goettingen Hannover
IImenau     Madrid     Malmoe     Marburg    PaloAlto
Pittsburgh  Rochester  Singapore  Szeged     Warsaw
```

While using any one of these themes, the colour of the themes appear blue. The colour can be changed by using the command, `\usecolortheme[named=colour name]{structure}` in the preamble, where the colour name is one of the standard colours (or a defined colour, in which case the `\definecolor` command must be placed before the `\usecolortheme` command). We can also use a colour theme by using the command,

```
\usecolortheme{colour theme name}
```

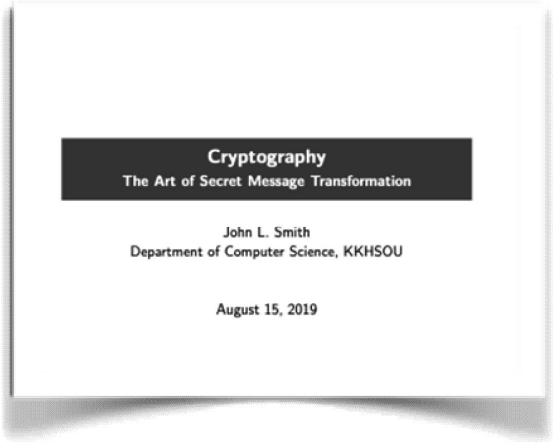
in the preamble. Some standard colour themes of Beamer class are:

```
Albatross beaver beetle crane dolphin dove fly lily
orchid seahorse whale  wolverine
```

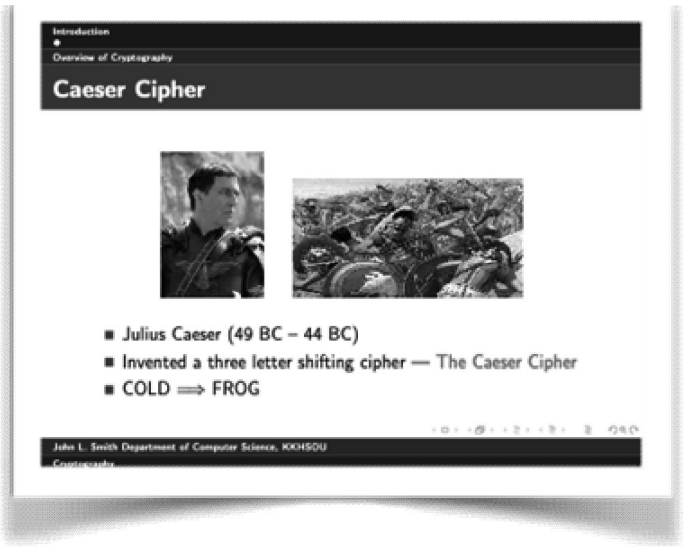
For instance, when we define

```
\usecolortheme{whale}
\usefonttheme{structurebold}
```

In the preamble of the above example, the first page of the output file looks like:



It is important to note that in the above example we have not used any theme. If we use, for example the theme Berlin, then the output will show where you are and who is presenting the slides. Below we show how any one of the slides look like when we use the command `\usetheme{Berlin}`:



While using Beamer, we can still use the basic commands of LaTeX. The commands for itemize, enumerate and description are the very same as in any LaTeX document. A nice feature of Beamer is that all of the information on the slide does not need to come up at once.

Use the command `\pause` to stop the information on the slide at that point. For example,

```
\begin{itemize}
\item Step 1
\pause
\item Step 2
```

will show Step 1, then the slide show must be prompted to show Step 2. The above commands will actually create 2 slides. The first slide will simply have Step 1, and the second will have both Step 1 and Step 2. The file that is created after compiling will have two slide counts. One will be the total number of slides (taking into account each time the `\begin{frame}` and `\pause` commands are used), and the other will be the total number of frames (only taking into account each time the `\begin{frame}` command is used).

To insert graphics into a Beamer presentation, first we call the package graphicx,

`\usepackage{graphicx}` then use the command,

`\includegraphics[*]{picture name}` where * includes the usual commands about the width, height, trimming, etc. It is nice to be able to include text with graphics. To do this, we can create columns in our frame using the `columns` and `column` commands.

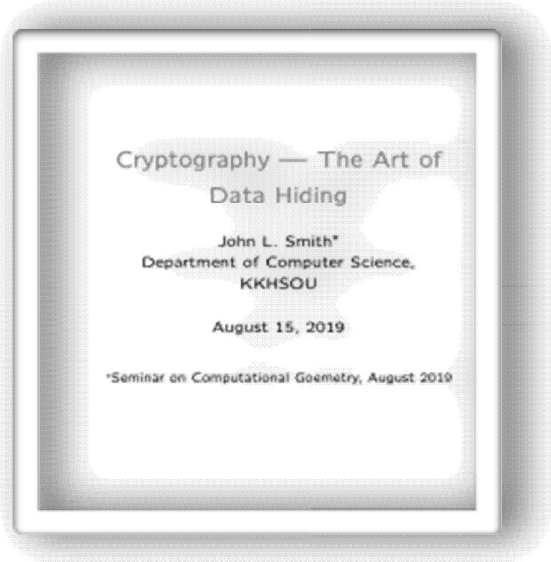
```
\begin{columns}
\begin{column}{0.5\textwidth}
Text here
\end{column}{0.5\textwidth}
\begin{column}{0.5\textwidth}
\includegraphics[width=0.9\textwidth]{picture name}
\end{column}{0.5\textwidth}
\end{columns}
```

The result of these commands will produce a frame with two equal sized columns, text on the left and graphics on the right. The size of the columns can be changed by replacing the 0.5 with other values between 0 and 1 exclusive. More than two columns can be made in the same fashion. Note that when the command `width=0.9\textwidth` is used for the graphics, this is the width of the column (not the width of the frame). The `\pause` command can be used just as before when using columns.

6.8 SLIDES

Below we have shown an example for preparing slides using the document class `slide`.

```
\documentclass{slides}
\usepackage{geometry,color}
\geometry{screen}
\title{\color{cyan}Cryptography --- The Art of Data Hiding}
\author{John L. Smith\footnote{Seminar on Computational Goe-metry, August 2019}\\Department of Computer Science, KKHSOU}
\date{\today}
\maketitle
{\color{magenta} Introduction -- From History to Number Theory}
\begin{enumerate}
\item[--] Julius Caesar invented Caeser cipher for transmit-ting secret message among commanders in battle fields.
\item[--] The Caeser Cipher was a shifting Cipher where each letter is shifted by successive three letters of the alpha-bet.
\end{enumerate}
\end{document}
```



6.9 LETTER CLASS

The Letter class allows for very quick letter writing. For this purpose, we need to load the document class package `letter`. Each letter consists of four main parts —- opening, main body, closing, and signature. In order to understand its structure, we shall use the example given below.

```
\documentclass[a4paper,12pt]{letter}
\name{AnupBorphukan}
\signature{}
\address{Department of Economics, \\Sikkim
University\\ Sikim 983 002}

\begin{document}
\begin{letter}{Academic Registrar,\\ K KHandiqui
State Open University,\\ Housfed Complex, Dispur\\
Guwahati 06}

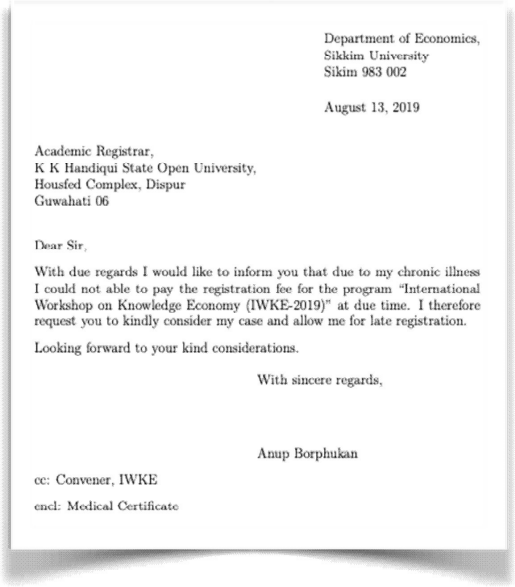
\opening{Dear Sir,}
With due regards I would like to inform you that due
to my chronic illness I could not able to pay the
registration fee for the program ``International
Workshop on Knowledge Economy (IWKE-2019)" at due
```

```
time. I therefore request you to kindly consider my
case and allow me for late registration.

Looking forward to your kind considerations.

\closing{With sincere regards,}

\cc{Convener, IWKE}
\encl{Medical Certificate}
\end{letter}
\end{document}
```



Department of Economics,
Sikkim University
Sikim 983 002

August 13, 2019

Academic Registrar,
K K Handiqui State Open University,
Housfed Complex, Dispur
Guwahati 06

Dear Sir,

With due regards I would like to inform you that due to my chronic illness I could not able to pay the registration fee for the program "International Workshop on Knowledge Economy (IWKE-2019)" at due time. I therefore request you to kindly consider my case and allow me for late registration.

Looking forward to your kind considerations.

With sincere regards,

Anup Borphukan

cc: Convener, IWKE

encl: Medical Certificate

As we show in the above example, the typical structure of writing a letter in LaTeX is as follows.

```
\documentclass{letter}
\name{sender's name}
\signature{space for sender's hand-written signature}
\address{sender's address}
\begin{document}
\begin{letter}{recipient's address}
\opening{salutation}
letter body
\closing{Yours sincerely or with sincere regards}
```

```
\end{letter}
\end{document}
```

As shown above, the argument recipient's address may contain multiple lines. In order to separate each line, we can use the new-line command `\\`. Just like the recipient's address, the sender's address often contains multiple lines separated by a double backslash `\\`. LaTeX will put the sender's name under the closing, after a vertical space for the traditional hand-written signature; it also can contain multiple lines.

Any additional material can be added after the \closing command. For instance, in some official letters, whoever is receiving a copy of the letter is mentioned towards the end of the letter. This can be easily done in LaTeX by using command like `\cc{Secretary, Dean's Office \\ Registrar, George Alfred University}`. For a list of enclosures, there is a `\encl{}` command.



CHECK YOUR PROGRESS

Q 12: Create the first page of a project report using `report` document class. The title of the project is Data Mining, submitted by a Third Semester Student of BCA courses in KKHSOU.

.....

.....

Q 13: What are the main parts of a letter that can be generated using `letter` document class?

.....

.....

Q 14: Write down the fields that are required to create a letter in LaTeX.

.....

.....

Q 15: How do we write the sender's address in a letter?

.....

.....



6.10 LET US SUM UP

- Different types of documents can easily be created using the pre-defined `documentclass` templates available in LaTeX.
- Each document class differs from each other in styles and arrangements.
- While creating a report or an article, we can include an abstract (or a summary) in the title-page by using the `abstract` environment.
- The table of contents, list of figures, and list of tables can automatically be generated by using commands `\tableofcontents`, `\listoffigures`, and `\listoftables`.
- Chapters can only be created in report and book class using the command `\chapter`. We can easily generate list of index in `book` class by using the package `makidx`.
- The title-page in a `report` or in a `book` class can be prepared in a separate file which can be included in the master file using the command `\input{filename}`. This way, each chapter of a book can also be prepared separately.
- `Beamer` class contains widely different themes that can be used effectively to produce slides for presenting a topic in different styles.
- Writing a letter is quite easier in LaTeX using `letter` template. The essential fields are `\name{}`, `\signature{}`, `\address{}`, `\opening{}`, `\closing{}`, `\cc{}`, and `\encl{}`.



6.11 FURTHER READINGS

1) Goossens, M., Rahtz, S. P., Rahtz, S., & Mittelbach, F. (1997). *The LATEX graphics companion: illustrating documents with TEX and PostScript (Vol. 1)*. Addison-Wesley Professional.

2) Knuth, *D. E., & Bibby, D. (1984). The texbook (Vol. 3)*. Reading: Addison-Wesley.

3) Mittelbach, F., Goossens, M., Braams, J., Carlisle, D., & Rowley, C. (2004). *The LATEX companion. Addison-Wesley Professional*.



6.12 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Different types of document class that are distributed with the La-TeX package are `article, letter, report, book, beamer,` etc.

Ans to Q No 2: The command line for inserting a document class in a latex file is `\documentclass [option, option, option] {class}.`

Ans to Q No 3: The options that can be defined in declaring command line of the document class are font sizes (`10pt, 11pt,12pt`), paper sizes (`a4paper, letterpaper`), `titlepage,` number of columns `(onecolumn, twocolumn)`, print style `(oneside,twoside),` formula style `(leqno, fleqn)` etc.

Ans to Q No 4: The command for preparing an article with font size 12 point and paper size legal is `\documentclass[12pt,legalpaper]{article}`

Ans to Q No 5: The fields that we can fill in the title page of an article are `\title{}, \author{},\date.` Within the command `\author{},` we can insert the affiliation of the author by using newline command `\\`. For instance, `\author{John L. Smith\\ Department of Computer Science \\ Krishna KantaHandiqui State Open University\\ Housefed Complex, Guwahati 781 006}.`

Ans to Q No 6: The command for printing the title page is \maketitle which has to be inserted after \begin{document}.

Ans to Q No 7: The footnote can be created by using the command `\thanks{}.`

Ans to Q No 8: A chapter can be created only when we choose the document class as either book or report. The command for creating a chapter is `\chapter{}`.

Ans to Q No 9: The LaTeX commands are as follows:

```
\documentclass[12pt]{book}
\usepackage{graphicx}
\begin{document}
\pagenumbering{roman}
\thispagestyle{empty}
\begin{center} \Huge{The Mythological History of
Dragons}\
end{center}
\vskip 3.0cm
\begin{center} \sc Wild Frank\end{center}
\vskip 2.0cm
\begin{center}{\sc University of
Waterloo}\end{center}
\end{document}
```

Ans to Q No 10: The command for creating a page with Roman page number is:

```
\pagenumbering{roman}
```

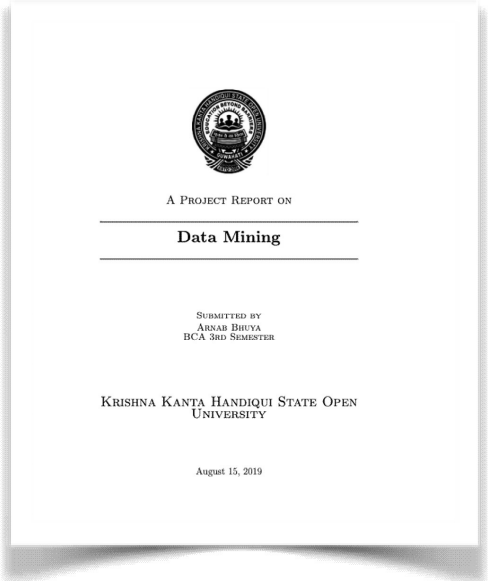
Ans to Q No 11: The command for creating the first page of a book without a page number is: `\thispagestyle{empty}`

Ans to Q No 12:

```
\documentclass[12pt,a4paper]{report}
\usepackage{graphicx}
\newcommand{\HRule}{\rule{\linewidth}{0.5mm}}
\begin{document}
\begin{titlepage}
\begin{center}

\includegraphics[width=0.3\textwidth]{logo}\\[1cm]
\textsc{\Large A Project Report on}\\[0.5cm]
```

```
\HRule \\[0.4cm]
{ \huge \bfseries
Data
Mining}\\[0.4cm]
\HRule \\[2.5cm]
{ \ l a r g e \ s c
Submitted by
\\ ArnabBhuya\\ BCA
3rd
Semester}\\[3cm]
\ t e x t s c { \ L A R G E
Krishna
KantaHandiqui State Open
University}\\[2.5cm]
{\large \today}
\end{center}
\end{titlepage}
\end{document}
```



A Project Report on

Data Mining

Submitted by
Arnab Bhuya
BCA 3rd Semester

Krishna Kanta Handiqui State Open University

August 15, 2019

Ans to Q No 13: There are four main parts of a letter that can be generated using `letter` document class. These are: opening, main body, closing, and signature.

Ans to Q No 14: The essential fields that are required to create a letter in LaTeX are: `\name{}`, `\signature{}`, `\address{}`, `\opening{}`, `\closing{}`, `\cc{}`, and `\encl{}`.

Ans to Q No 15: The sender's address in a letter can be written using the field `\address{}`.



6.13 MODEL QUESTIONS

Q 1: Make a list for the various document classes available in LaTeX.

Q 2: Write down the general command line for declaring the document class in LaTeX.

Q 3: What are the different types of options that can be inserted along with the declaration of document class?

Q 4: How article class differs from report class?

Q 5: Create the first page of an article entitled “Application of Geometry in Computation”, written by J. L. Chuang, University of Oxford. Include the date as 12 February, 2008.

Q 6: Write down the template for `report` class and create the first page for a report on “Graphics Design Using Free CAD” submitted for the partial fulfillment of MCA degree at KKHSOU.

Q 7: What are the different themes available in beamer class? How can you change the color of the slides in beamer?

Q 8: Produce three slides showing the essential topics that are covered in BCA course at KKHSOU. Write the heading of the slides as “BCA Course at KKHSOU”.

Q 9: What are the essential fields that are required in writing a letter using `letter` template?

Q 10: Using the `letter` template write a letter to the Deputy Commissioner of your district requesting for a correction of your name that appeared in NRC list.

*** ***** ***

UNIT 7 : APPLICATIONS OF LATEX

UNIT STRUCTURE

7.1 Learning Objectives

7.2 Introduction

7.3 Applications of LaTeX

7.4 Writing Resume

7.5 Writing Question Paper

7.6 Inserting Graphics

7.7 Plot Functions

7.8 Writing Complicated Equations

7.9 Let Us Sum Up

7.10 Further Readings

7.11 Answers to Check Your Progress

7.12 Model Questions

7.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- apply LaTeX in a more efficient way in creating resume or CV
- use LaTeX in creating question papers of different styles
- write a research paper or article
- use Beamer for presentation
- use graphics package

7.2 INTRODUCTION

In the previous units we have described how to use some of the standard document classes that are available with LaTeX in creating documents. We have also covered how to write a standard article using `article` class and how to create presentation files or slides using `beamer` class. In this unit, you will be introduced to various other applications of LaTeX and how it can be used in an effective way in writing resume, question papers, inserting and drawing graphics, and

writing complicated equations. In this unit, a number of examples along with the `tex` files are described so that they can be used for practicing purposes.

7.3 APPLICATIONS OF LATEX

There are various application of LaTeX, particularly in academic and official activities. For example, it can be used as one of the most effective and efficient tools in creating documents of widely different styles. Once we understand the structure of LaTeX documentations, we can easily use it in preparing question papers or writing resumes or curriculum vitae as we shall describe below. We shall also describe how LaTeX can be used in inserting and drawing graphics using commands. Towards the end of this unit we shall describe how LaTeX can be used in writing complicated mathematical expressions without any hassle.

7.4 WRITING RESUME

A **resume** is a document containing background, skills, and accomplishments of a person. Resumes can be used for a variety of reasons, but most often they are used to secure new employment. A resume is a summary version of one's education and experience while a curriculum vitae (CV) is a comparatively longer containing a more detailed information of the person.

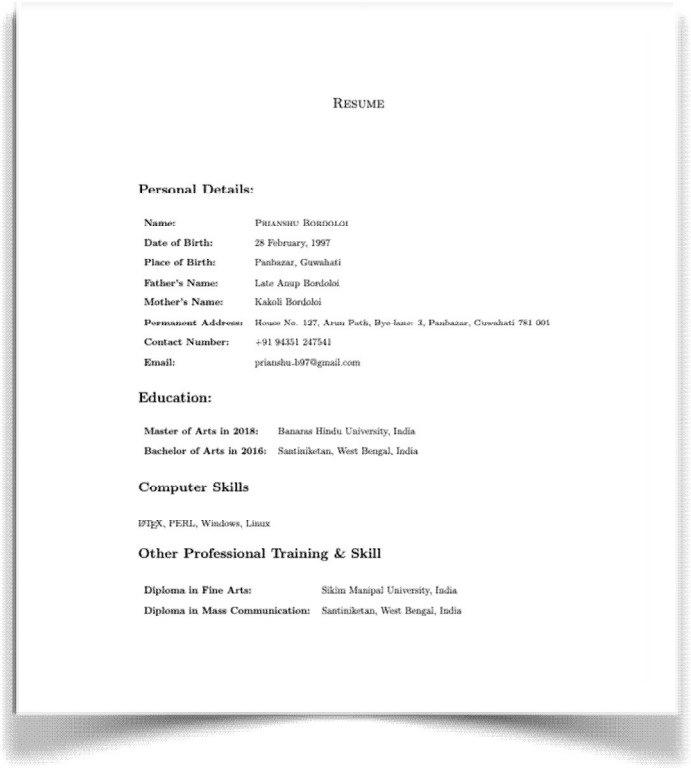
A resume or a CV can be prepared easily in LaTeX. Below we show a simple resume which is prepared using `article` class.

```
\documentclass{article}
\usepackage[a4paper, left=3cm, right=2cm,
top=2cm,bottom=2cm{geometry}
\begin{document}
\pagestyle{empty}
\begin{center}
{\Large\sc Resume}\\[2.5cm]
\end{center}
```

```
\section{Personal  Details:}
\vskip 0.5cm
\begin{tabular}{ll}
{\bf Name:} & {\sc Prianshu Bordoloi}\\[0.3cm]
{\bf Date of Birth:} & 28 February, 1997\\[0.3cm]
{\bf Place of Birth:} & Panbazar, Guwahati\\[0.3cm]
{\bf Father’s Name:} & Late Anup Bordoloi\\[0.3cm]
{\bf Mother’s Name:} & Kakoli Bordoloi\\[0.3cm]
{\bf Permanent Address:} & House No. 127, Arun
Path, Byelane:
3, Panbazar, Guwahati 781 001\\[0.3cm]
{\bf Contact Number:} & +91 94351 247541\\[0.3cm]
{\bf Email:} & prianshu\_b97@gmail.com\\[0.3cm]
\end{tabular}
\section{Education:}
\vskip 0.5cm
\begin{tabular}{ll}
{\bf Master of Arts in 2018:} & Banaras Hindu
University, India\\[
0.3cm]
{\bf Bachelor of Arts in 2016:} & Santiniketan,
West Bengal,
India\\[0.3cm]
\end{tabular}
\section{Computer Skills}
\vskip 0.5cm
\LaTeX, PERL, Windows, Linux
\section{Other Professional Training \& Skill}
\vskip 0.5cm
\begin{tabular}{ll}
{\bf Diploma in Fine Arts:} & Sikim Manipal
University, India\\[
0.3cm]
```

```
{\bf Diploma in Mass Communication:} &
Santiniketan, West
Bengal, India\\
\end{tabular}
\end{document}
```

Figure 7.1: Resume Example



RESUME

Personal Details:

Name:	PRIANSHU BORDOLOI
Date of Birth:	28 February, 1997
Place of Birth:	Panbazar, Guwahati
Father's Name:	Late Anup Bordoloi
Mother's Name:	Kakoli Bordoloi
Permanent Address:	House No. 127, Arun Path, Bye-lane: 3, Panbazar, Guwahati 781 001
Contact Number:	+91 94351 247541
Email:	prianshu.b97@gmail.com

Education:

Master of Arts in 2018:	Banaras Hindu University, India
Bachelor of Arts in 2016:	Santiniketan, West Bengal, India

Computer Skills

LaTeX, PERL, Windows, Linux

Other Professional Training & Skill

Diploma in Fine Arts:	Sikim Manipal University, India
Diploma in Mass Communication:	Santiniketan, West Bengal, India

Here we are using simple article class to produce a resume as shown above. As you see, we write the title **Resume** in the `center` environment as

```
\begin{center}
{\Large\sc Resume}\\[2.5cm]
\end{center}
```

We use the font size `\Large` and font style `\sc` for small cap. We can use `\section` command to fill the various fields like personal detail, education, computer skills, employment history, etc. The personal details are entered within the `tabular` environment. The line `\begin{tabular}{ll}` indicates that tabular environment contains two columns indicated by `{ll}`, `l` indicates that the column is left-aligned.

As you see in the output file, the argument `Name` appears in the left column and “Prianshu Bordoloi” appears in the right column. The contents of each column are separated by the symbol `&`. This way, we can prepare a simple resume using the known commands.

Instead of using `article` class for writing a CV or a resume, if we want a different style, we can use some standard document class that are available for writing a CV. For instance, we can use `moderncv` class to produce a modern CV. This involves various themes using which we can change the style of the CV for better looking. Below we show an example to show how `moderncv` class can be used for a CV or a resume. This example can be used as a template for practice.

```
\documentclass[11pt,a4paper,sans]{moderncv}
\usepackage[scale=0.75]{geometry}

%% ModernCV themes
\moderncvstyle{casual}
\moderncvcolor{blue}
\nopagenumbers{}

%% Personal data
\firstname{Anil}
\familyname{Buragohain}
\title{Resumé}
\address{Department of Archaeology, Downing Street,
University of Cambridge}{Cambridge, CB2 3DZ}
%\address{street and
number}{postcode city}
\mobile{+44 (0)1223 333 520}
\phone{+44 (0)1223 333 520}
\fax{+44 (0)1223 339288}
\email{anilb@arch.cam.ac.uk}
\homepage{www.arch.cam.ac.uk}
\extrainfo{additional information}
\photo[64pt][0.4pt]{shadow.jpg}
```

```
%\quote{Some quote (optional)}

\begin{document}
\makecvtitle

\section{Education}
\cventry{1987—1991}{PhD in Archaeology}{Pennsylvania
State University}{
Pennsylvania.}{\textit{Suervisor:}}
{Dr. L. Hamilton}
\cvitem{Thesis Title:}{\emph{Application of Remote
Sensing and Historical
Archaeology in Ancient Rome} }
\cventry{1985—1987}{MS in Archaeology}{Pennsylvania
State University}{
Pennsylvania}{\textit{}}{}
\cvitem{Thesis Title:}{\emph{Archaeology in Ancient
Rome} }

%% A publications list
\nocite{*}
\bibliographystyle{plain}
\bibliography{publications}
\end{document}
```

As shown in the above example, we have inserted a photo named shadow.jpg by using the command `\photo`. The picture will appear at the top left position of the resume. Further, we have used the command `\bibliography{publications}` for writing the list of publications. It contains a bibtex file named `publications.bib`. This bibtex file containing the list of publications needs to be prepared separately. Here we prepare such a bibtex file as shown below and named it as `publications.bib`.

```
@book{Shlesinger95,
editor="M. F. Shlesinger and A. Buragohain and U.
```

```
Frish”,
title=”Ancient Archeology in Rome”,
publisher=”Springer”,
year=”Berlin (1995)”}
@book{Wiltschko95,
author=”R. Wiltschko and W. Wiltschko and A.
Buragohain”,
title=”Animal Archeology”,
publisher=”Springer”,
year=”Berlin (2001)”}
@article{Meinhardt72,
author=”H. Meinhardt and A. Buragohain”,
title=”A modern tool of archaeology for
investigating the {A}ncient
{R}ome”,
journal=”Kybernetik {\bf 12}”,
year=”30-39 (2009)”}
```

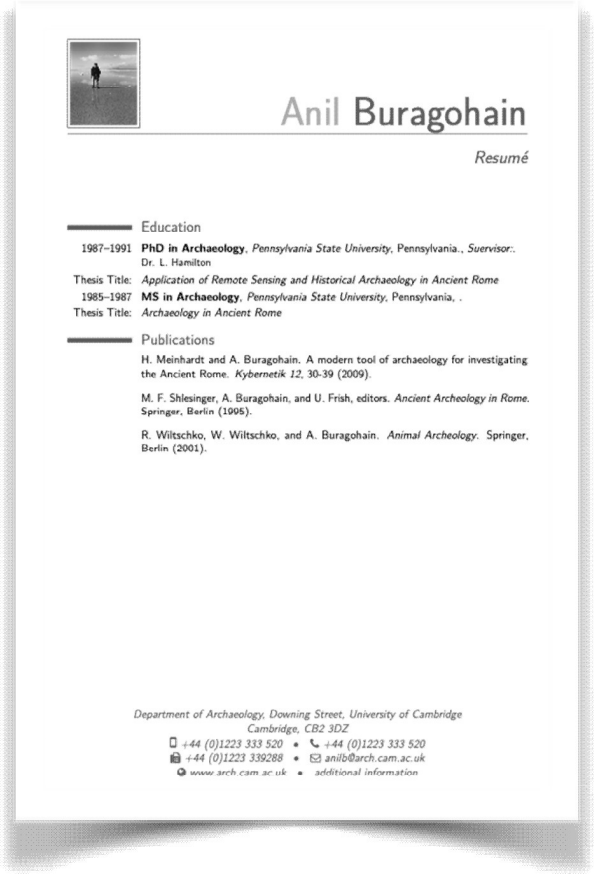
You can create such a bibtex file using any text editor. However it is important to remember that the bibtex file should be saved in the same folder containing the resume file.

Once both the files (resume.tex and publications.bib) are ready, we run the command

```
latex resume.tex
bibtex resume.bib
latex resume.tex
dvipdf resume.dvi resume.pdf
```

Sometimes the above sequence of commands is required to run twice in order to appear the reference list correctly. Here, we show how the resume.pdf file looks like:

Figure 7.2: Resume.pdf example



Anil Buragohain

Resumé

Education

1987–1991 **PhD in Archaeology**, *Pennsylvania State University*, Pennsylvania., *Suervisor:*. Dr. L. Hamilton

Thesis Title: *Application of Remote Sensing and Historical Archaeology in Ancient Rome*

1985–1987 **MS in Archaeology**, *Pennsylvania State University*, Pennsylvania, .

Thesis Title: *Archaeology in Ancient Rome*

Publications

H. Meinhardt and A. Buragohain. A modern tool of archaeology for investigating the Ancient Rome. *Kybernetik 12*, 30-39 (2009).

M. F. Shlesinger, A. Buragohain, and U. Frish, editors. *Ancient Archeology in Rome*. Springer, Berlin (1995).

R. Wiltschko, W. Wiltschko, and A. Buragohain. *Animal Archeology*. Springer, Berlin (2001).

Department of Archaeology, Downing Street, University of Cambridge
Cambridge, CB2 3DZ
+44 (0)1223 333 520 • +44 (0)1223 333 520
+44 (0)1223 339288 • anilb@arch.cam.ac.uk
www.arch.cam.ac.uk • additional information

In the first line of the document, we declare the document class as `\documentclass[11pt,a4paper,sans]{moderncv}`

Here we can use other options for font size ('`10pt`', '`11pt`' and '`12pt`'), paper size (`a4paper`, `letterpaper`, `a5paper`, `legalpaper`, `executivepaper` and `landscape`) and font family (`sans` and `roman`).

In the second line, we declare `\usepackage[scale=0.75]{geometry}`. This is used to adjust the page margins. The next two lines

```
\moderncvstyle{casual}
\moderncvcolor{blue}
```

declare the style and color. There are many styles and colours that can be used within `moderncv` class. For instance, instead of casual, we can use other options such as `classic`, `oldstyle` and `banking`.

For color, other options are `orange, green, red, purple, grey` and `black`. The next line `\nopagenumbers{}`is used to suppress automatic page numbering. If we want the page number, the page number can be generated by commenting this line in the file as as `%\nopagenumbers{}`.

In the `moderncv` class, all the personal data are written in the preamble, i.e., before declaring `\begin{document}` command. As we see, while entering personal data, there exists a number of fields. These are`\firstname{}, \familyname{}, \title{}, \address{}{}, \mobile{}, \phone{} \fax{}, \email{}, \homepage{}, \extrainfo{}, \photo[][]{filename}`. If we do not want to appear any one of these fields, we can remove it from the output file just by commenting the field using `%` sign. For instance if we write

```
% \photo[64pt][0.4pt]{shadow.jpg}
```

the picture will not appear in the output pdf file.

As you see, we use `\photo[64pt][0.4pt]{shadow.jpg}` where `'64pt'` indicates the height of the picture, `0.4pt` is the thickness of the frame around it and `shadow.jpg` is the name of the picture. If we replace `0.4pt` by `0pt`, no frame will appear in the picture. The appearance of other fields of the personal profile , namely, `\address{}{}, \mobile{}, \phone{} \fax{}, \email{}, \homepage{},`and`\extrainfo{},` depends on the options of the command `\moderncvstyle{}`. They appear in the bottom line of the first page in the case of `casual` style, as we have shown above. If we use `classic` style, then they will appear at the top right, as shown below. However, while using `classic` style, we need to define a line such as

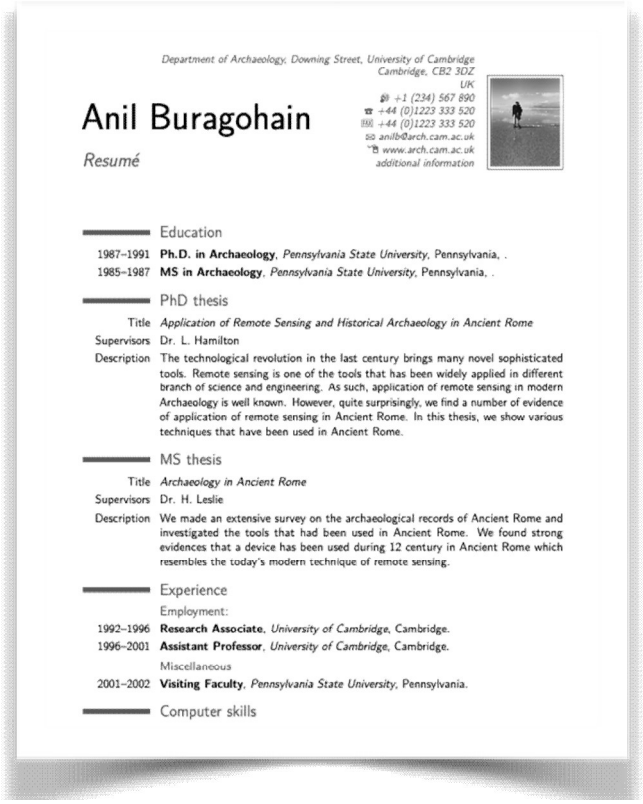
```
\setlength{\makecvtitlenamewidth}{10cm}
```

This simply declares a width of 10cm for the name that appear at the top left to avoid any line breaks.

Thus, in the above mentioned LaTeX file, if we change the command line `\moderncvstyle{casual}` to `\moderncvstyle {classic}` and add one more line as `\setlength{ \makecvtitl-`

enamewidth}{10cm}, then the style of the resume looks like what we have shown here.

Figure 7.3: Resume Example with classic style



Department of Archaeology, Downing Street, University of Cambridge
Cambridge, CB2 3DZ
UK
+1 (234) 567 890
+44 (0)1223 333 520
+44 (0)1223 333 520
anilb@arch.cam.ac.uk
www.arch.cam.ac.uk
additional information

Anil Buragohain

Resumé

Education

1987–1991 **Ph.D. in Archaeology**, *Pennsylvania State University*, Pennsylvania, .
1985–1987 **MS in Archaeology**, *Pennsylvania State University*, Pennsylvania, .

PhD thesis

Title *Application of Remote Sensing and Historical Archaeology in Ancient Rome*
Supervisors Dr. L. Hamilton
Description The technological revolution in the last century brings many novel sophisticated tools. Remote sensing is one of the tools that has been widely applied in different branch of science and engineering. As such, application of remote sensing in modern Archaeology is well known. However, quite surprisingly, we find a number of evidence of application of remote sensing in Ancient Rome. In this thesis, we show various techniques that have been used in Ancient Rome.

MS thesis

Title *Archaeology in Ancient Rome*
Supervisors Dr. H. Leslie
Description We made an extensive survey on the archaeological records of Ancient Rome and investigated the tools that had been used in Ancient Rome. We found strong evidences that a device has been used during 12 century in Ancient Rome which resembles the today's modern technique of remote sensing.

Experience

Employment:

1992–1996 **Research Associate**, *University of Cambridge*, Cambridge.
1996–2001 **Assistant Professor**, *University of Cambridge*, Cambridge.

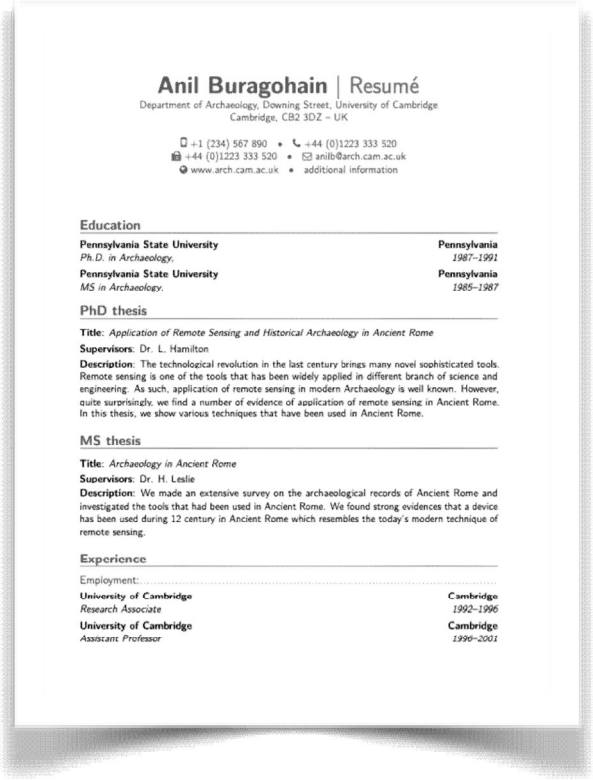
Miscellaneous

2001–2002 **Visiting Faculty**, *Pennsylvania State University*, Pennsylvania.

Computer skills

It is important to remember that the command line \setlength{\makecvtitlenamewidth}{10cm} is used only for classic style. If we use different style, we have to run the file by commenting this line, otherwise the LaTeX compiler will show an error message. The advantage of using moderncv class is that we can produce the output with different styles just by changing the style in the command line \moderncvstyle{}. Below we show how it differs when we change the casual style to banking.

Figure 7.4: Resume Example with banking style



Anil Buragohain | Resumé

Department of Archaeology, Downing Street, University of Cambridge
Cambridge, CB2 3DZ – UK

+1 (234) 567 890 • +44 (0)1223 333 520
+44 (0)1223 333 520 • anilb@arch.cam.ac.uk
www.arch.cam.ac.uk • additional information

Education

Pennsylvania State University — **Pennsylvania**
Ph.D. in Archaeology. — *1987–1991*
Pennsylvania State University — **Pennsylvania**
MS in Archaeology. — *1985–1987*

PhD thesis

Title: *Application of Remote Sensing and Historical Archaeology in Ancient Rome*
Supervisors: Dr. L. Hamilton
Description: The technological revolution in the last century brings many novel sophisticated tools. Remote sensing is one of the tools that has been widely applied in different branch of science and engineering. As such, application of remote sensing in modern Archaeology is well known. However, quite surprisingly, we find a number of evidence of application of remote sensing in Ancient Rome. In this thesis, we show various techniques that have been used in Ancient Rome.

MS thesis

Title: *Archaeology in Ancient Rome*
Supervisors: Dr. H. Leslie
Description: We made an extensive survey on the archaeological records of Ancient Rome and investigated the tools that had been used in Ancient Rome. We found strong evidences that a device has been used during 12 century in Ancient Rome which resembles the today's modern technique of remote sensing.

Experience

Employment:

University of Cambridge — **Cambridge**
Research Associate — *1992–1996*
University of Cambridge — **Cambridge**
Assistant Professor — *1996–2001*

As we see here, the picture at the top left does not appear here as the command `\photo` is not going to work when we use banking style in `moderncv` class.

Let us go back again to the LaTeX file. As we see there, the essential fields like Education, Experience etc. are entered through `\section` command. The data in each field are entered through `\cventry` and `\cvitem` commands. The `\cventry` command has four subfields, each one enclosed by curly brackets. If all the subfields are not needed, we can leave them empty but can not change the structure, otherwise the LaTeX compiler will show error messages.

Let's see how we have entered the field Education in the resume. Below we show the command lines that involve the command `\cventry`.

```
\section{Education}
\cventry{1987–1991}{Ph.D. in Archaeology}{Pennsylvania
State
University}{Pennsylvania}{}{}
```

```
\cventry{1985—1987}{MS in Archaeology}{Pennsylvania State
University}{Pennsylvania}{}{}
```

For writing the section PhD Thesis in the resume, we use he following commands consisting of `\cvitem` command.

```
\section{PhD thesis}
\cvitem{Title}{\emph{Application of Remote Sensing and Historical
Archaeology in Ancient Rome}}
\cvitem{Supervisors}{Dr. L. Hamilton}
\cvitem{Description}{The technological revolution in the last
century brings many novel sophisticated tools. Remote sensing
is one of the tools that has been widely applied in different
branch of science and engineering. As such, application of
remote sensing in modern Archaeology is well known. However,
quite surprisingly, we find a number of evidence of application
of remote sensing in Ancient Rome. In this thesis, we
show various techniques that have been used in Ancient
Rome.}
```

As we see, the \cvitem command has two subfields. This way we can create an attractive CV of our own.



CHECK YOUR PROGRESS

Q 1: What are the minimum requirements for producing a resume using article class?

.....

.....

Q 2: Write down the environments that are required to create a resume in LaTeX using article class.

.....

.....

Q 3: What is the structure of moderncv class?

.....

.....

Q 4: Write down the fields that are available in moderncv class for writing personal information.

.....

.....

Q 5: What is the command for changing style in moderncv class?

.....

.....

Q 6: Write down the available style in moderncv class.

.....

.....

7.5 WRITING QUESTION PAPERS

We see widely different styles of question papers depending on the type of examination, who is setting the questions and for what purpose the questions are asked. The style also varies according to the marks distribution. In general, in a standard university question paper, we see the first-page as the instruction page for the examination together with marks distribution. In LaTeX, we can create a question papers with different style using the standard document class called `exam.cls`. It sets the page layout so that there are one inch margins all around (no matter what is the size of the paper) and provides commands that make it easy to format questions, create flexible headers and footers, change the margins, and create grading tables. Below we show some examples showing the structure of the `.tex` files together with the resulting output

pdf files from which we can learn how to use the `exam` class for writing a question paper in LaTeX.

Before starting the exam class, it is important to know how can we create a question paper using `article` class. Here is one example:

```
\documentclass[11pt]{article}
\usepackage{amsmath,amssymb,amsfonts}

\setlength{\topmargin}{0.0in}
\setlength{\textheight}{9.25in}
\setlength{\oddsidemargin}{0.1in}
\setlength{\textwidth}{6.2in}

\begin{document}
\Large

\noindent{\bf Coordinate Systems: Test 1\hfill
\today}\\
\medskip\hrule
\vskip 0.1cm
\noindent{Time: 1 Hour} \hfill Total Marks: 20

\begin{enumerate}
\item A two-dimensional plane in a coordinate system
is defined
as $x=r\cos\theta$, $y=r\sin\theta$
\begin{enumerate}
\item What will be the position coordinate of a vector
${\mathbf r^2}$ in terms of $r$ and $\theta$.
\item What is the inverse coordinate transformation
equations?
\item Define the range of $r$ and $\theta$ when $-
\infty\leqslant x\leqslant\infty$ and $-
\infty\leqslant
```

```
x\leqslant\infty$.  \hfill[2+2+2=6]
\end{enumerate}

\item  In  a  spherical  polar  coordinate  system
\begin{enumerate}
   \item What are the coordinates involved?
   \item Write down the coordinate transformation
   equations.
   \item Derive the expression for volume element.
\hfill[2+2+4=8]
\end{enumerate}

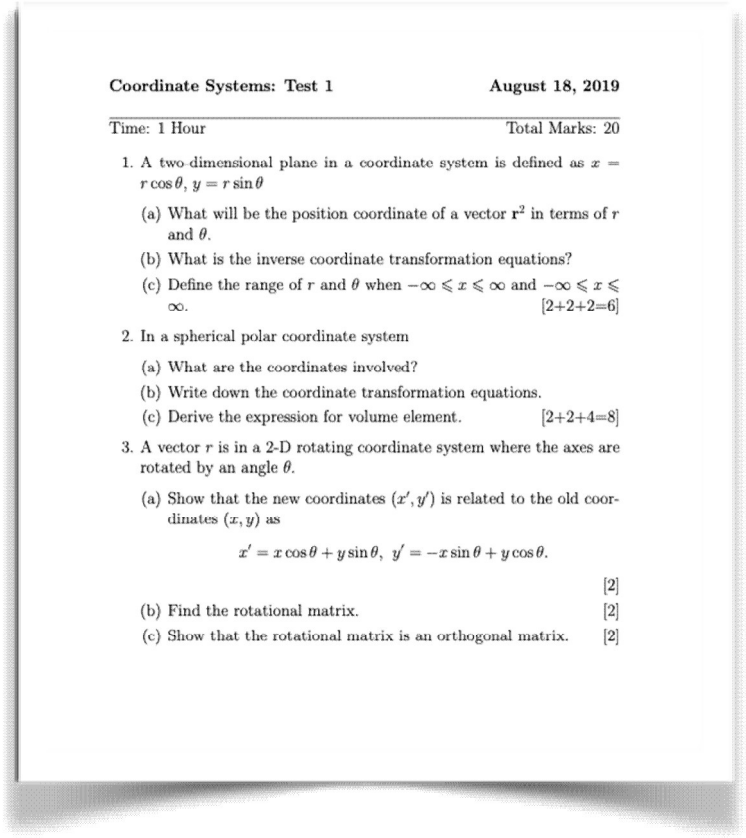
\item A vector $r$ is in a 2-D rotating coordinate
system
where the axes are rotated by an angle $\theta$.
\begin{enumerate}
\item Show that the new coordinates ($x',y'$) is
related to
the old coordinates ($x,y$) as
\[x'=x\cos\theta+y\sin\theta,
\hspace{0.2cm} y'=-x\sin\theta+y\cos\theta. \]
\hfill[2]

\item Find the rotational matrix.\hfill[2]
\item Show that the rotational matrix is an orthogonal
matrix.
\hfill[2]
\end{enumerate}

\end{enumerate}
\end{document}
```

The output file of this tex file is as shown below in figure 7.5.

Figure 7.5: Question Paper example using article class



Coordinate Systems: Test 1 **August 18, 2019**

Time: 1 Hour Total Marks: 20

1. A two-dimensional plane in a coordinate system is defined as $x = r\cos\theta$, $y = r\sin\theta$
   (a) What will be the position coordinate of a vector $\mathbf{r}^2$ in terms of r and θ.
   (b) What is the inverse coordinate transformation equations?
   (c) Define the range of r and θ when $-\infty \leqslant x \leqslant \infty$ and $-\infty \leqslant x \leqslant \infty$. [2+2+2=6]
2. In a spherical polar coordinate system
   (a) What are the coordinates involved?
   (b) Write down the coordinate transformation equations.
   (c) Derive the expression for volume element. [2+2+4=8]
3. A vector r is in a 2-D rotating coordinate system where the axes are rotated by an angle θ.
   (a) Show that the new coordinates (x', y') is related to the old coordinates (x, y) as
   $$x' = x\cos\theta + y\sin\theta,\ y' = -x\sin\theta + y\cos\theta.$$
   [2]
   (b) Find the rotational matrix. [2]
   (c) Show that the rotational matrix is an orthogonal matrix. [2]

As shown in figure 7.5, here we have used article class and all the questions are written within `\begin{enumerate} . . . \end{enumerate}` environment. The points (Marks) are given by using `\hfill` command. Below we show a simple example where the `exam` class is used to create a University question paper.

```
\documentclass[letterpaper,12pt,addpoints]{exam}
\usepackage[top=1in, bottom=1in, left=0.75in,
right=0.75in]{geometry}
\usepackage{amsmath,amssymb}

\newcommand{\university}{University of Hyderabad}
\newcommand{\faculty}{Department of Computer Science
\& Engineering}
\newcommand{\class}{CSE-26}
\newcommand{\examnum}{Class Test \#3}
\newcommand{\content}{C Programming \& Numerical
```

```
Methods}
\newcommand{\examdate}{08/06/2018}
\newcommand{\timelimit}{60 minutes}

\pagestyle{headandfoot}
\firstpageheader{}{}{}
\firstpagefooter{}{Page \thepage\ of \numpages}{}
\runningheader{\class}{\examnum}{\examdate}
\runningheadrule
\runningfooter{}{Page \thepage\ of \numpages}{}

\begin{document}

\title{\Large \textbf{\university\\ \faculty\\
\bigskip
\class — \examnum \\ \content}}
\author{Instructor: Prof. M. K. Dubey}
\date{\examdate}
\maketitle
\begin{flushleft}
\makebox[12cm]{\textbf{Name}:\ \hrulefill}
\medskip
\makebox[12cm]{\textbf{Student Number}:\ \hrulefill}
\end{flushleft}
\noindent \rule{\textwidth}{1pt}

\noindent This exam contains \numpages\ pages
(including this
cover page) and \numquestions\ questions. Total of
points is
\numpoints.\\
Good luck and Happy reading work!

\begin{center}
\textbf{Distribution of Marks}\\
\medskip
\gradetable[v][questions]
\end{center}
```

```
\clearpage
\begin{questions}
\question
An array is given as $A=\{1,5,4,9,5,6\}$.

\begin{parts}
\part[3] Write a C program to find the sum of the
array.
\part[3] Write a C program to find the sum of
minimum absolute
difference of the array.
\part[2] Check the correctness of your results.
\end{parts}

\clearpage
\question
A general form of a first-order differential
equation is given
as $\frac{dx}{dy}=f(x,y)$, with $y(0)=y_0$.

Compute the following:
\begin{parts}
\part[3] Write down the algorithm for solving the
equation
with Runge-Kutta 4th Order Method.
\part[3] Write a C programme for solving the given
equation
with Runge-Kutta 4th Order Method.
\part[2] What is the significance of tolerance
factor in your
computation?
\end{parts}

\clearpage
\question A quadratic equation is given as
$5x^2+3x-2=0$.
Compute the following:
```

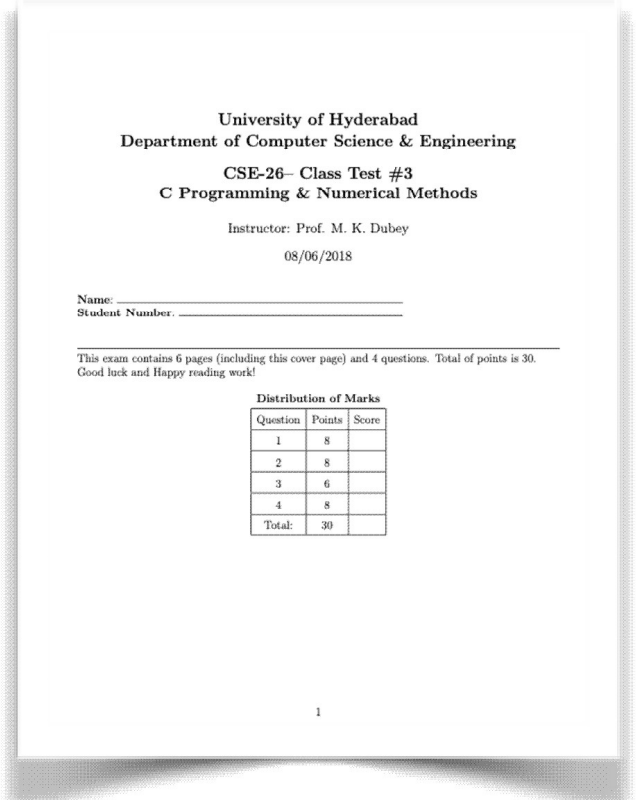
```
\begin{parts}
\part[2] What are the steps involved in solving
the given
equation numerically using bisection method?
\part[1] Under what conditions the bisection method
fails in
solving the equation?
\part[3] Write the algorithm for solving the given
equation
using bisection method.
\end{parts}

\clearpage
\question An integral is given as $I=\int_a^b f(x)
dx$. Answer
the following:
\begin{parts}
\part[4] What approximation is involved in solving
the given
integral using Simpson's $\frac{1}{3}$ rule? Write
down the
algorithm for solving the given integral with
$f(x)=\sin^2x$
using Simpson's $\frac{1}{3}$ rule.
\part[4] Write a C program for it.
\end{parts}
\end{questions}
\clearpage

This page is intentionally left blank to accommodate
work that wouldn't fit elsewhere and/or scratch work.
\end{document}
```

The first two pages of the output pdf file is shown below in figure 7.6.

Figure 7.6: (a) First page of question paper with exam class



University of Hyderabad
Department of Computer Science & Engineering

CSE-26– Class Test #3
C Programming & Numerical Methods

Instructor: Prof. M. K. Dubey

08/06/2018

Name: _____
Student Number. _____

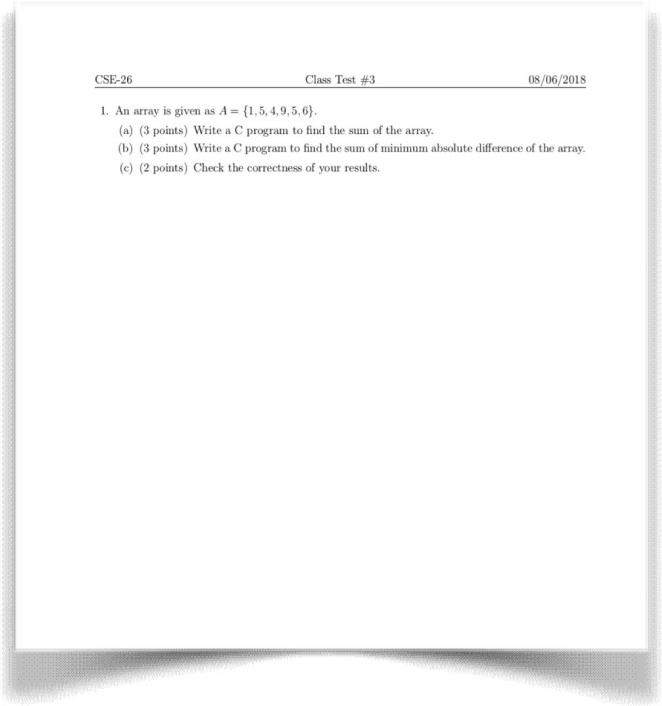
This exam contains 6 pages (including this cover page) and 4 questions. Total of points is 30.
Good luck and Happy reading work!

Distribution of Marks

Question	Points	Score
1	8	
2	8	
3	6	
4	8	
Total:	30	

1

Figure 7.6: (b) Second page of question paper with exam class



CSE-26 Class Test #3 08/06/2018

1. An array is given as $A = \{1, 5, 4, 9, 5, 6\}$.
   (a) (3 points) Write a C program to find the sum of the array.
   (b) (3 points) Write a C program to find the sum of minimum absolute difference of the array.
   (c) (2 points) Check the correctness of your results.

As shown here, the basic structure of an `exam` class can be classified into three parts: (i) the title page (ii) the header and footer and (iii) questions and marks.

The title-page consists of `\title` and `\author`, as in `article` or `report` class. As the title-page consists of a number of fields, here we use the `\newcomand` option for simplicity. A `\newcommand` provides a new short-hand name against its entry. For instance, here we use `\university` as a new command for the University of Hyderabad. We use the other new commands such as `\faculty, \class, \examnum, \content, \examdate,` and `\timelimit`. The table in the first page for indicating the distribution of marks is created by the command line`\gradetable[v][questions].`

For header and footer we use the command `\pagestyle {headandfoot}.` The command line`\firstpagefooter{}{Page \thepage\ of \numpages}{}`will display page 1 of 7 in the footer of the first-page and so on. Here the command `\thepage` and `\numpages` indicate the current page number and total number of pages, respectively. We also use the command line `\runningheader{\class} {\examnum}{\examdate}`to display the class, exam name and exam date, respectively, as the header of second page onwards.

All the questions are written within the `\begin{questions}. . .\end{questions}` environment. Within this environment, a question is created by the command \question. If a question involves parts such as Q.1 (a), (b), (c), then they can be created by writing each part within `\begin{parts}. . . \end{parts}` environment. For instance, our first question in the above example is

1. An array is given as A = {1,5,4,9,5,6}.
   (a) (3 points) Write a C program to find the sum of the array.
   (b) (3 points) Write a C program to find the sum of minimum absolute difference of the array.
   (c) (2 points) Check the correctness of your results.

We created it by using the commands within the `\begin {questions}. . .\end{questions}` environment as:

```
\question
An array is given as $A=\{1,5,4,9,5,6\}$.
\begin{parts}
\part[3] Write a C program to find the sum of the
array.
\part[3] Write a C program to find the sum of
minimum absolute
difference of the array.
\part[2] Check the correctness of your results.
\end{parts}
```



CHECK YOUR PROGRESS

Q 7: How do you create the questions while preparing a question paper using article class?

.....

.....

Q 8: What are the essential fields in exam class?

.....

.....

Q 9: How do you create a table that shows distribution of marks in the firstpage of a question paper?

.....

.....

Q 10: Write the command lines for creating a question having three parts, namely:

(a) What is X?

.....

(b) What is Y?

.....

(c) What is Z? Each carries 1 mark. (Use the exam class).

.....

7.6 INSERTING GRAPHICS

In order to import graphics, we need to load the `graphicx` package in the preamble of our document. We can do this by using the command below.

```
\usepackage{graphicx}
```

The supporting image format depends on the compiler. We can only include Encapsulated PostScript (EPS) format while compiling with `latex`. It is because the EPS file (extension is normally `.eps`) declares the size of the image and, hence, make it easy for LaTeX to arrange the text and the graphic in the best way. There are many decent graphics software that has the ability to save images in the EPS format. For example, commercial graphics software such as *Adobe illustrator, CorelDRAW* and *FreeHand* are commonly used and can read and write EPS figures. However, these packages are limited to Windows and Mac-OS platform. The package Inkscape can save in vector EPS format and it can run on multiple platforms.

Instead of compiling with `latex`, if we use `pdflatex`, there will be wider choice and we can insert varieties of graphics format such as JPG, PNG, PDF, etc. JPG and PNG are supported by any image processing program.

Such a graphics file can be inserted in the LaTeX file using the command `\includegraphics` as follows:

```
\includegraphics[option1,option2,option3]{elephant.eps}
```

For instance, if we have an image file for an elephant, we can simply insert it using the following commands, provided the image file is kept in the same directory.

```
\documentclass[11pt]{article}
\usepackage{graphicx}
\begin{document}

\begin{figure}[h]
\begin{center}
\includegraphics[width=0.8\textwidth,height=0.6\textheight,an
```

```
gle=90]{elephant.eps}
\caption{A roaming elephant in Africa}
\label{fig:elephant}
\end{center}
\end{figure}
\end{document}
```

Figure 7.7: Example of image with caption



Figure 1: A roaming elephant in Africa

As we have seen, we can use options such as `width, height,` and `angle,` to resize the image. Further, in order to positioning figure, LaTeX uses placement specifiers. In the above example, we have used the placement specifier `h` in the line `\begin{figure} [h]` which indicates that the figure should be displayed approximately at the same point it occurs in the source text. Instead of `[h]`, we can use other specifiers such as `[t]` for displaying at the top of the page, `[b]` for bot-tom of the page, and [!] for a good float position.

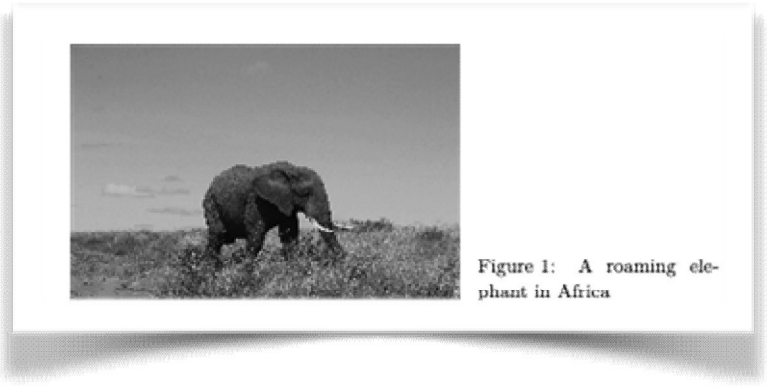
It is always good practice to add a caption to any figure that we incorporate in a document. In LaTeX, this can be easily done simply using the `\caption{text}` command within the float environment. In the above example, we have used `\caption{A roaming elephant in Africa` to describe the figure. LaTeX automatically take care of numbering the figure, we don't need to include this within the caption

text. The location of the caption is, traditionally, underneath the float. However, if we wish, we can place it before and the caption will appear above the float. Instead of above and below, sometimes it is desirable to have a caption appear on the side of a float. This can be done using `sidecap` package as shown below.

```
\documentclass[11pt]{article}
\usepackage{graphicx,sidecap}
\begin{document}
\begin{SCfigure}
\includegraphics[width=0.6
\caption{A roaming elephant
\label{default}
\end{SCfigure}
\end{document}
```

This will produce a side caption as shown in figure 7.8.

Figure 7.8: Example of image with side caption



Sometimes we prefer not to break the flow of the text and allow text to wrap around the figure. In LaTeX, there are several package to allow this. Below we show an example to show how such a task is possible using the package `wrapfig`.

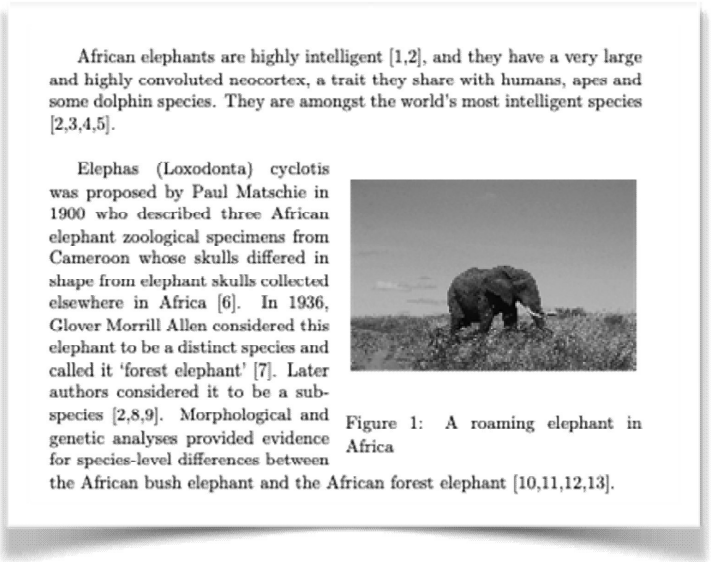
```
\documentclass[11pt]{article}
\usepackage{graphicx,wrapfig}
\begin{document}
African elephants are highly intelligent [1,2], and
they have a very large and highly convoluted neocortex,
a trait they share with humans, apes and some dolphin
```

```
species. They are amongst the world's most intelligent
species [2,3,4,5].
\begin{wrapfigure}{r}{0.5\textwidth}
   \begin{center}
      \includegraphics[width=0.48\textwidth]{elephant.jpg}
   \end{center}
   \caption{A roaming elephant in Africa}
\end{wrapfigure}

Elephas (Loxodonta) cyclotis was proposed by Paul
Matschie in 1900 who described three African elephant
zoological specimens from Cameroon whose skulls differed
in shape from elephant skulls collected elsewhere in
Africa [6]. In 1936, Glover Morrill Allen considered
this elephant to be a distinct species and called it
'forest elephant' [7]. Later authors considered it to
be a subspecies [2,8,9]. Morphological and genetic
analyses provided evidence for species-level differences
between the African bush elephant and the African
forest elephant [10,11,12,13].
\end{document}
```

This will produce an output as shown in figure 7.9.

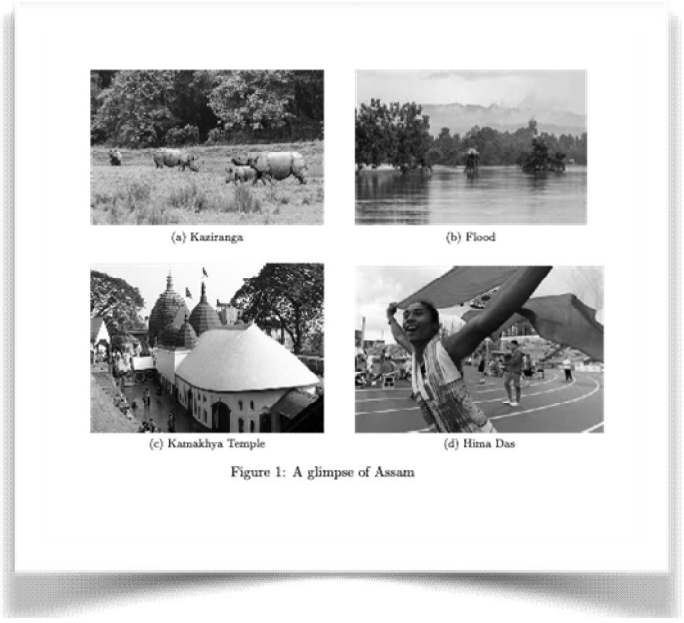
Figure 7.9: Example of image with caption without breaking flow of text



In order to insert subfigures within figure in LaTeX, there is a package, called **subfig**. An example of how to use this subfig package is given below.

```
\documentclass[11pt,a4paper]{article}
\usepackage{graphicx,subfig}
\begin{document}
\begin{figure}
\subfloat[Kaziranga]{\label{fig:kaziranga}\includegraphics[wi
dth=0.5\textwidth,height=0.2\textheight]{Kaziranga}}
\hspace{0.8cm}
\subfloat[Flood]{\label{fig:flood}\includegraphics[width=0.5\
textwidth,height=0.2\textheight]{Flood}}\\[0.5cm]
\subfloat[Kamakhya Temple]{\
label{fig:kamakhya}\includegraphics[width=0.5\textwidth
,height=0.22\textheight]{Kamakhya}}\hspace{0.8cm}
\subfloat[Hima
Das]{\label{fig:him}\includegraphics[scale=0.7]{Hima}}
\caption{A glimpse of Assam}
\end{figure}
\end{document}
```

Figure 7.10: Using subfigures inside a figure





CHECK YOUR PROGRESS

Q 11: An image file of a house is given as boathouse.jpg . How do you resize the picture by an amount of 0.6 of the text width, 0.2 of the text height, and rotate it by an angle of -90 degree?

.....

.....

Q 12: How do you display an image caption on the side of the image?

.....

.....

Q 13: What is the use of wrapfig package in inserting a graphics?

.....

.....

Q 14: How do you create a number of subfigures? Write down the command line.

.....

.....

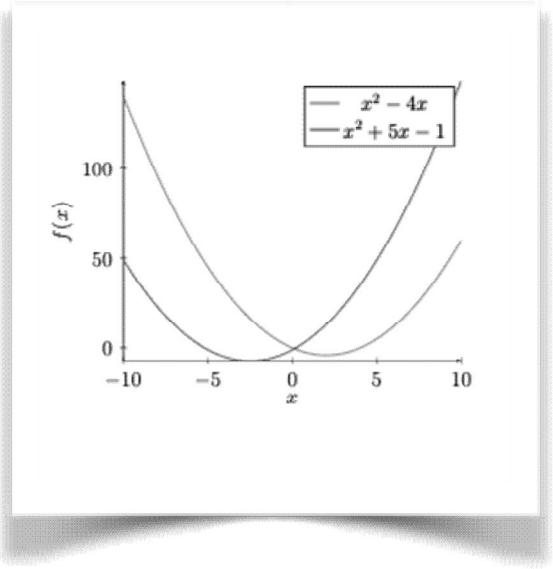
7.7 PLOT FUNCTIONS

Drawing graphics in LaTeX can be done in various ways. One of the methods is the use of TikZ package that we have already described in Unit 5 with a number of examples. We shall not discuss it here again. Instead, we shall see here how `tikz picture` environment can be used in drawing, for instance, various graphs or mathematical functions.

In order to plot functions, we need to use the package `pgfplots` which is a powerful visualisation tool and it is based on tikz package . As such, the plot must be inside the `tikzpicture` environment. Following examples will show how easier it becomes to plot a function just by using the command `\addplot`. For example, let us see how to plot the functions $f(x)= x^2-4x$ and $f(x)=x^2 + 5x - 1$. Here is the tex file:

```
\documentclass[12pt]{article}
\usepackage{pgfplots}
\begin{document}
\begin{tikzpicture}
\begin{axis}[axis lines=left, xlabel=$x$,
ylabel={$f(x)$},]
\addplot [domain=-10:10,samples=100,color=red,]{x^2 -
4*x};
\addlegendentry{$x^2 - 4x$}
\addplot [domain=-10:10,samples=100,color=blue, ]{x^2
+ 5*x - 1};
\addlegendentry{$x^2 + 5x - 1$}
\end{axis}
\end{tikzpicture}
\end{document}
```

Figure 7.11: Plot for functions f(x)= x^2-4x and f(x)=x^2 + 5x - 1.

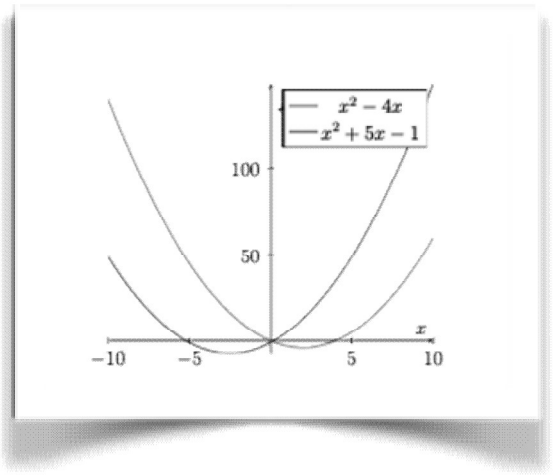


As shown in the tex file, we use `\usepackage{pgfplots}` and the `\begin{tikzpicture} . . . \end{tikzpicture}` environment. In order to plot the functions, we use the command `\addplot[plotoptions]{ function}`. In the plot options, we can define the domain-size (representing the range of x-axis), number of

samples (data points to be plotted within the domain), and color of the graph.

As we see, there is one more environment that we have used here: `\begin{axis} … \end{axis}` to control the position of the axes, labelling of the axes and for giving a legend for the plots. We have defined `axis lines=left, xlabel=$x$, ylabel={$f(x)$}`. The command `axis lines=left` sets the axes only on the left and bottom sides of the plot. If we change `axis lines=left` to `axis lines=center`, then the plots will looks like the one as shown in figure 7.12. We can hide the axes by writing `hide axes` in place of `axis lines` command. Further, in order to put labels on the horizontal and vertical axis, we use the `xlabel` and `ylabel` commands.

Figure 7.12: Plot for functions with centered axis

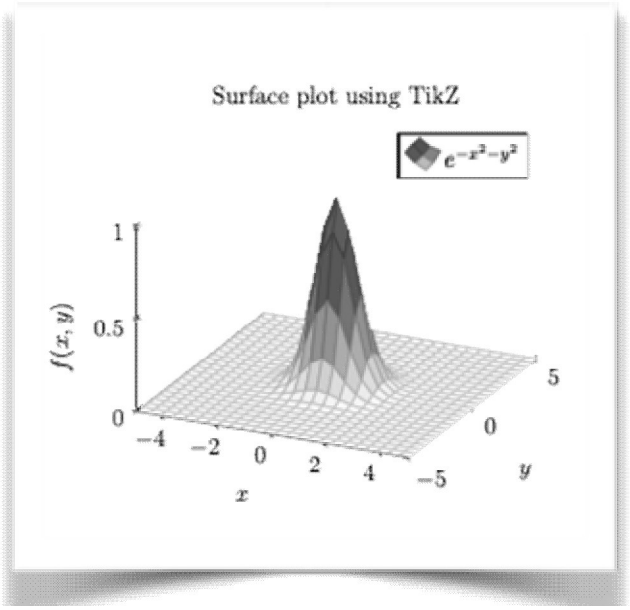


Here we shall discuss some more tricks involved in drawing graphs using surface plots. In `tikzpicture` environment, we can use `\addplot3[surf]` for creating surface plot of a given function. Here we show how to create the surface plot of a function $f(x, y) = e^{-x^2-y^2}$.

```
\documentclass[12pt]{article}
\usepackage{pgfplots}
\begin{document}
\begin{tikzpicture}
\begin{axis}[title=Surface plot using TikZ, axis
```

```
lines=left,
xlabel=$x$, ylabel=$y$, zlabel={$f(x,y)$}, colormap/
cool,]\addplot3[surf]{exp(-x^2-y^2)};
\addlegendentry{$e^{-x^2-y^2}$}
\end{axis}
\end{tikzpicture}
\end{document}
```

Figure 7.13: Example showing Surface plot

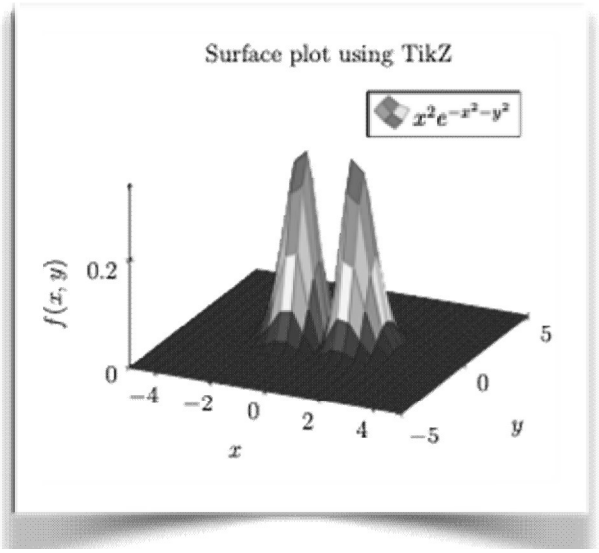


As we see here in figure 7.13, we can draw surface plot of any functions just by using `tikzfigure` environment. We have used a `title` and `colormap/cool` as the color scheme. Here is one more example:

```
\documentclass[12pt]{article}
\usepackage{pgfplots}
\begin{document}
\begin{tikzpicture}
\begin{axis}[title=Surface plot using TikZ, axis
lines=left,
xlabel=$x$, ylabel=$y$, zlabel={$f(x,y)$}, colormap/
hot,]
\addplot3[surf]{exp(-x^2-y^2)*x^2};
```

```
\addlegendentry{$e^{-x^2-y^2}$}
\end{axis}
\end{tikzpicture}
\end{document}
```

Figure 7.14: Another example showing Surface plot

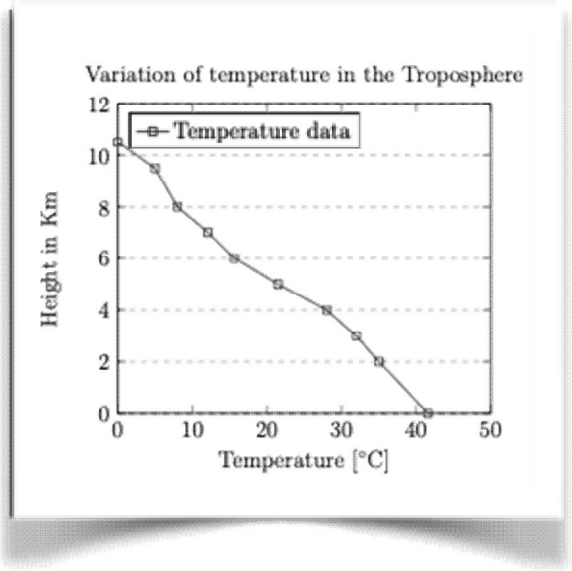


Here in figure 7.14 we plot the function $f(x,y) = x^2e^{-x^2-y^2}$ and have used the `colormap/hot` as the color scheme. `TikZ` can also be used to plot data points as shown in figure 7.15.

```
\documentclass[12pt]{article}
\usepackage{pgfplots}
\begin{document}
\begin{tikzpicture}
\begin{axis}[
title={Variation of temperature in the
Troposphere},
xlabel={Temperature [$^\circ$C]},
ylabel={Height in Km},
xmin=0, xmax=50,
ymin=0, ymax=12,
xtick={0,10,20,30,40,50},
ytick={0,2,4,6,8,10,12},
legend pos=north west,
```

```
ymajorgrids=true,
grid  style=dashed,
]\addplot[color=blue,mark=square,]coordinates{(0,10.5)(5,9.5)
(8,8)(12.1,7)(15.6,6)(21.5,5)(28,4)(32,3)  (35,2)
(41.6,0)};
\legend{Temperature  data}
\end{axis}
\end{tikzpicture}
\end{document}
```

Figure 7.15: Example showing plotting of data points





CHECK YOUR PROGRESS

Q 15: What is `pgfplots`?

.....

.....

Q 16: How do we change the axes position while plotting a function using tikzfigure environment?

.....

.....

Q 17: What are the environments that are required in order to plot graphs using pgfplots?

.....

.....

Q 18: Create a LaTeX file for drawing the graph $f(x)=x^3-6x^2 + 5x - 9$.

.....

.....

Q 19: A set of data points for the rise in water temperature with time for a constant supply of heat are given as (time, temperature) =(0,26), (2,26), (4,30), (6,36), (8,43), (10,99). Plot the data using pgfplots.

.....

.....

7.8 WRITING COMPLICATED EQUATIONS

We have came across various utilities of `amsmath` and `amssymb` packages in creating mathematical environment in Unit 2. We shall not repeat the same thing again, instead we shall show in this unit, how to write a complicated mathematical expression simply by using `amsmath` and `amssymb` packages. Below are some examples.

(i) $$\sum_{\substack{1\le i\le p\\ 1\le j\le q\\ 1\le k\le r}} a_{ij}b_{jk}c_{kl}$$

The expression of summation involves a number of limits. In order to write it in LaTeX, we can use `\substack` command as follow:

```
\begin{equation}
\sum_{\substack{1\le i\le p\\ 1\le j\le q\\
1\le k\le r} }a_{ij}b_{jk}c_{ki}
\end{equation}
```

(ii) $$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n cos\frac{n\pi x}{L} + b_n sin\frac{n\pi x}{L}\right)$$

This can be produced by writing in the following way:

```
f(x)=a_0+\sum_{n=1}^\infty\left(a_n\cos\frac{n\pi
x}{L}+b_n\sin\frac{n\pi x}{L} \right)
```

As shown here, we use `\left(. . . \right)` to produce the first brackets in a larger size.

(iii) $$(1+x)^n = 1 + \frac{nx}{1!} + \frac{n(n-1)x^2}{2!} + \cdots$$

This can be produced by writing:

```
(1+x)^n=1+\frac{nx}{1!}+\frac{n(n-1)x^2}{2!}+\ldots
```

(iv) $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, -\infty < x < \infty$

The above equation can be produced by writing:

```
e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots, -\infty<x<\infty
```

(v) $\lim_{n\to\infty} \int_0^1 x^n + y^n dx$

The above equation can be produced by writing:

```
\[ \lim_{n\to\infty} \int_0^1 x^n + y^n \ dx \]
```

(vi) $\sum_{i=1}^{\infty} \frac{1}{n^s} = \prod_p \frac{1}{1-p^{-s}}$

The above equation can be produced by writing:

```
\[ \sum_{i=1}^{\infty} \frac{1}{n^s}
  = \prod_p \frac{1}{1 - p^{-s}} \]
```

(vii) $\frac{n!}{k!(n-k)!} = \binom{n}{k}$

The above equation can be produced by writing:

```
\frac{n!}{k!(n-k)!} = \binom{n}{k}
```

(viii) $(x + a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k}$

The above equation can be produced by writing:

```
(x+a)^n = \sum_{k=0}^n\binom{n}{k}x^ka^{n-k}
```

(ix) $\int_{-1}^{+1} \frac{f(x)}{\sqrt{1-x^2}} dx \approx \frac{\pi}{n} \sum_{i=1}^{n} f\left(cos\left(\frac{2i-1}{2n}\right)\right)$

The above equation can be produced by writing:

```
\[ \int_{-1}^{+1}\frac{f(x)}{\sqrt{1-
x^{2}}}\,\mathrm{d}x\approx\frac{\pi}{n}
\sum_{i=1}^{n}f\left(\cos\left(\frac{2i-
}{2n}\right)\right)
\]
```



CHECK YOUR PROGRESS

Q 20: Show the expression

$sinx = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots$ in LaTeX.

.....

.....

Q 21: Show $\Gamma(n) = \int_0^\infty x^{n-1}e^{-x}dx$ in LaTeX.

.....

.....

Q 22: Write the command line for writing $\frac{d^2y}{dx^2} + k_0\frac{dx}{dt} + \omega_0^2 x = 0$

.....

.....



7.9 LET US SUM UP

- LaTeX is an effective tool as it has many wide applications for different purposes.
- Using LaTeX, we can create our own resume or CV according to our requirements. Using moderncv class, we can create them in a more attractive way.
- LaTeX can be used for setting question papers of different styles that are required for various purposes. In order to create question papers, we can uses exam class as it has many more options to change the layout.
- There are various graphics handling packages. Some of them, namely, subfig, wrapfig, sidecap are discussed here using which we can create a widely different look of the inserted figures.
- TikZ graphic package has a number of options for drawing graphics. Here we have discussed some of them which will be quite useful for many purposes, particularly, for drawing graphical plots of mathematical functions, for surface plots, and for plotting recorded data points.
- Any complicated mathematical expressions or equations can easily be produced using the amsmath and amssymb packages, as we have shown here some of them.



7.10 FURTHER READINGS

1) Kopka, H., & Daly, P. W. (1999). *A Guide to La T E X: document preparation for beginners and advanced users*.

2) Jackson, A. L., & Geckeis, C. K. (2003). *How to prepare your curriculum vitae*. McGraw-Hill.

3) Grätzer, G. (2013). *Math into LATEX*. Springer Science & Business Media.



7.11 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: The minimum requirements for producing a resume using `article` class are (i) the geometry package for page margin , (ii) `\section` command for arranging the content like personal details, education, computer skills, employment history, etc., and (iii) `the tabular environment` for arranging the sub fields.

Ans to Q No 2: The environments that are required to create a resume in LaTeX using `article` class are `centering` environment (`\begin{center} . . .\end{center}`) and tabular environment (`\begin{tabular}. . .\end{tabular}`).

Ans to Q No 3: The `moderncv` class contain a list of command for entering the subfields of personal information. It also contains `\section`, `\cventry`, and `\cvitem` commands.

Ans to Q No 4: The fields that are available in `moderncv` class for writing personal information are `\firstname{}`, `\familyname{}`, `\title{}`, `\address{}{}`, `\mobile{}`, `\phone{}` `\fax{}`, `\email{}`, `\homepage{}`, `\extrainfo{}`, `\photo[][]{filename}`.

Ans to Q No 5: The commands for changing style in `moderncv` class are `\moderncvstyle{}` and `\moderncvcolor{}`.

Ans to Q No 6: The available options for `\moderncvstyle{}` are `casual, classic, oldstyle and banking.` For `\moderncvcolor{},` the options are `blue, orange, green, red, purple, grey` and `black.`

Ans to Q No 7: Using `article` class we can create the questions by itemise them using `\begin{enumerate} . . .\end{enumerate}` environment.

Ans to Q No 8: The essential fields in `exam` class are `\title, \author, \questions, \part,` etc.

Ans to Q No 9: The table in the first page for indicating the distribution of marks I can be created by using the command line`\gradetable[v][questions].`

Ans to Q No 10:

```
\begin{questions}
\question
\begin{parts}
\part[1] What is X?
\part[1] What is Y?
\part[1] What is Z?
\end{parts}
\end{questions}
```

Ans to Q No 11: The given image can be resized using the options given in the command line of `\includegraphics` as follows: `\includegraphics[width=0.6\textwidth,height=0.2\textheight,angle=-90]{elephant.eps}`

Ans to Q No 12: The image caption can be displayed on the side of the image by using `sidecap` package and then writing the `\includegraphics` command within the `SCfigure` environment.

Ans to Q No 13: The package `wrapfig` is used with `graphicx` package to insert a picture by breaking the flow of the text and allowing text to wrap around the figure.

Ans to Q No 14: Subfigures are generated by using the `subfig` package along with the `graphicx` package. Below is an example where

two pictures named `tiger.jpg` and `monkey.jpg` are inserted and, in the output file, it will show “A picture of a Bengal Tiger” and “A monkey” as the sub caption names.

```
\documentclass[11pt,a4paper]{article}
\usepackage{graphicx,subfig}
\begin{document}
\begin{figure}
\subfloat[A picture of a Bengal Tiger]\
includegraphics[width=0.5\textwidth,height=0.2
\textheight
]{tiger.jpg}
\subfloat[A monkey]\
includegraphics[width=0.5\textwidth,height=0.2
\textheigh
t]{monkey.jpg}
\end{figure}
\end{document}
```

Ans to Q No 15: A `pgfplots` is a package used to draw graphs or plot mathematical functions.

Ans to Q No 16: Within the `\begin{axis} … \end{axis}` environment, if we declare `axis lines=options,` then we can change the position of the axes by changing the options. The available options are `left, right,` and `center`.

Ans to Q No 17: In `pgfplots` package the required environments for drawing graphs are: `\begin{tikzpicture}. . .\end{tikzpicture}`, and `\begin{axis}. . .\end{axis}`.

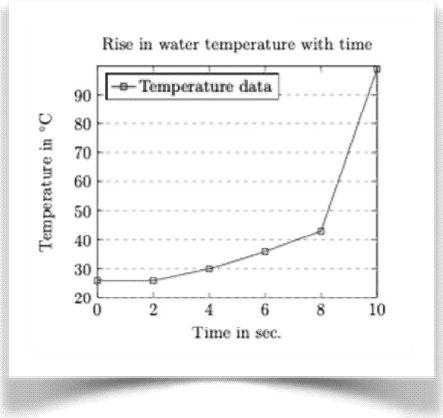
Ans to Q No 18: A LaTeX file for drawing the graph f(x)=x3-6x2+5x - 9 contains the following command lines:

```
\documentclass[12pt]{article}
\usepackage{pgfplots}
\begin{document}
\begin{tikzpicture}
\begin{axis}[axis lines=center, xlabel=$x$,
```

```
ylabel={$f(x)$},]
\addplot [domain=-
10:10,samples=100,color=red,]{x^3-6*x^2+5*x-9};
\addlegendentry{$x^3 -6x^2+5x-9$}
\end{axis}
\end{tikzpicture}
\end{document}
```

Ans to Q No 19:

```
\documentclass[12pt]{article}
\usepackage{pgfplots}
\begin{document}
\begin{tikzpicture}
\begin{axis}[
title={Rise in water temperature with time},
xlabel={Time in sec.},
ylabel={Temperature
in $^\circ$C},
xmin=0, xmax=10,
ymin=20, ymax=100,
xtick={0,2,4,6,8,10},
ytick={20,30,40,50,60,
70,80,90},
legend pos=north
west,
ymajorgrids=true,
grid style=dashed,
]\addplot[color=blue,mark=square,]coordinates{(0,26)
(2,26) (4,30)
(6,36) (8,43) (10,99)};
\legend{Temperature data}
\end{axis}
\end{tikzpicture}
\end{document}
```



Ans to Q No 20: `\sin x=x-`

```
\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots
```

Ans to Q No 21: `\Gamma(n)=\int_{0}^\infty x^{n-1}e^{-x} dx`

Ans to Q No 22:

```
\frac{d^2y}{dx^2}+k_0\frac{dx}{dt}+\omega_0^2x=0
```



7.12 MODEL QUESTIONS

Q 1: Create a simple resume using `article` class. Enter the fields Personal details, Education, Computer skills, and Employment history.

Q 2: How do you create different subfields within `article` class while preparing a resume?

Q 3: Explain how will you create different styles using `moderncv` class?

Q 4: How a picture of the author can be inserted in `moderncv` class?

Q 5: What is the structure of `exam` class? Describe the various fields that are used in `exam` class while creating a question paper.

Q 6: Create a LaTeX file containing a paragraph and a figure. Insert the figure within the paragraph.

Q 7: How do we wrap the texts around the figure?

Q 8: Write the command lines for displaying two subfigures in a figure.

Q 9: How `pgfplots` package is used for drawing graphs of mathematical functions?

Q 10: What are the available options within `tikzpicture` environment to draw graphs?

Q 11: A set of data points for the rise in water temperature with time for a constant supply of heat are given as (time, temperature)=(0,26), (2,26), (4,30), (6,36), (8,43), (10,99). Plot the data using `pgfplots`.

Q 12: Using LaTeX, produce the expression for the Laplacian in spherical polar coordinates given by:

$$\nabla^2\Psi = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\Psi}{\partial r}\right) + \frac{1}{r^2 sin\theta}\frac{\partial}{\partial\theta}\left(sin\theta\frac{\partial\Psi}{\partial\theta}\right) + \frac{1}{r^2 sin^2\theta}\frac{\partial^2\Psi}{\partial\phi^2}$$

*** ***** ***

UNIT 8 : INSTALLATION OF SCILAB

UNIT STRUCTURE

8.1 Learning Objectives

8.2 Introduction

8.3 Scilab

8.3.1 Installation of Scilab in Windows

8.3.2 Installation of Scilab in Linux

8.4 Let Us Sum Up

8.5 Further Readings

8.6 Answers to Check Your Progress

8.7 Model Questions

8.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- install Scilab in Windows
- install Scilab in Linux

8.2 INTRODUCTION

In the previous units in the first block you have been introduced to different applications of Latex like writing resume, writing question paper, writing articles/research papers. Applications like presentation using beamer, inserting graphics, drawing graphics and putting equations also had been covered in detail in the previous unit.

In this unit you will be introduced to installation of Scilab in both Windows and Linux environment. In the next unit we will discuss Scilab in detail. We will learn about basic syntax, mathematical operators, predefined constants, handle .sci files and install additional packages.

8.3 SCILAB

Scilab is an open source powerful computing environment for scientific and engineering applications. Scilab is developed by the Scilab

team within ESI Group. It is a major alternative for Matlab. Scilab is available under the GPL License. This means that the user has the freedom to:

- use the software for any purpose,
- change the software to suit different needs,
- share the software with friends and
- share the changes made by the user.

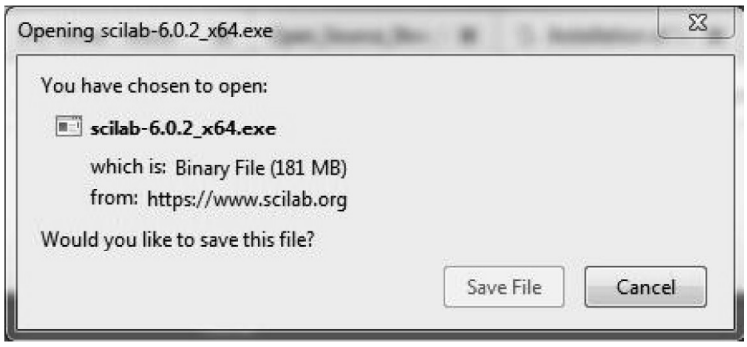
We will be now learn how to install Scilab in Windows and Linux environment.

8.3.1 Installation of Scilab in Windows

- Download Scilab from official Scilab website *https:// www.scilab.org/*, under download section and download the latest installation package. Download the Windows 64 bit package if you have a 64 bits machine or download the Windows 32 bit package in case of a 32 bits machine.

  Save the *scilab-x.x.x.exe* file in a location in your system.

Figure 8.1: Opening Scilab



- Run the *scilab-x.x.x.exe* file as administrator.

Figure 8.2: Run Scilab



Select the language for installation by using the dropdown list.

Figure 8.3: Select Language



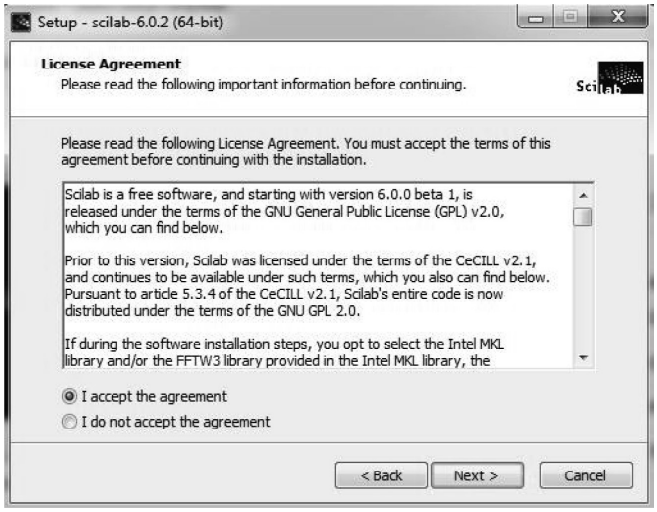
- Launch the Scilab setup wizard. Click “Next” button.

Figure 8.4: Scilab Setup



- Read the license agreement and select “I accept the agreement” then click on the “Next” button.

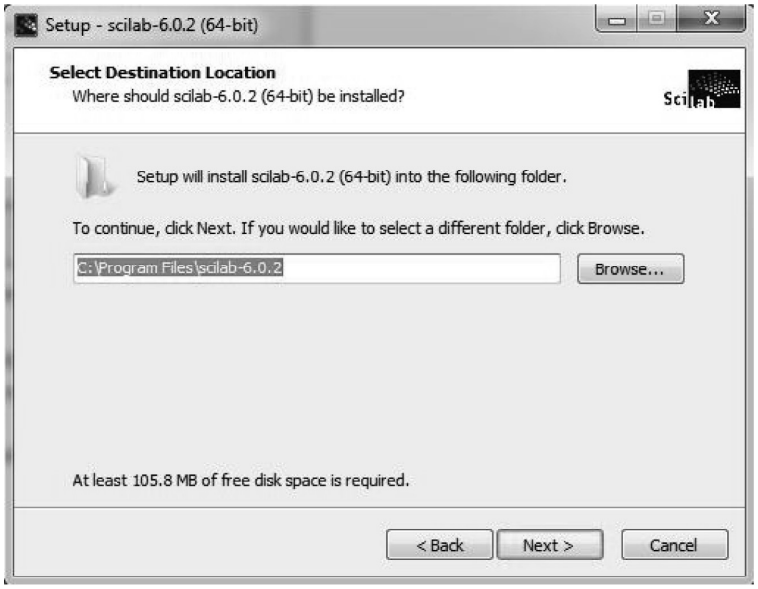
Figure 8.5: License Agreement



- Select a local hard drive location for the installation and make

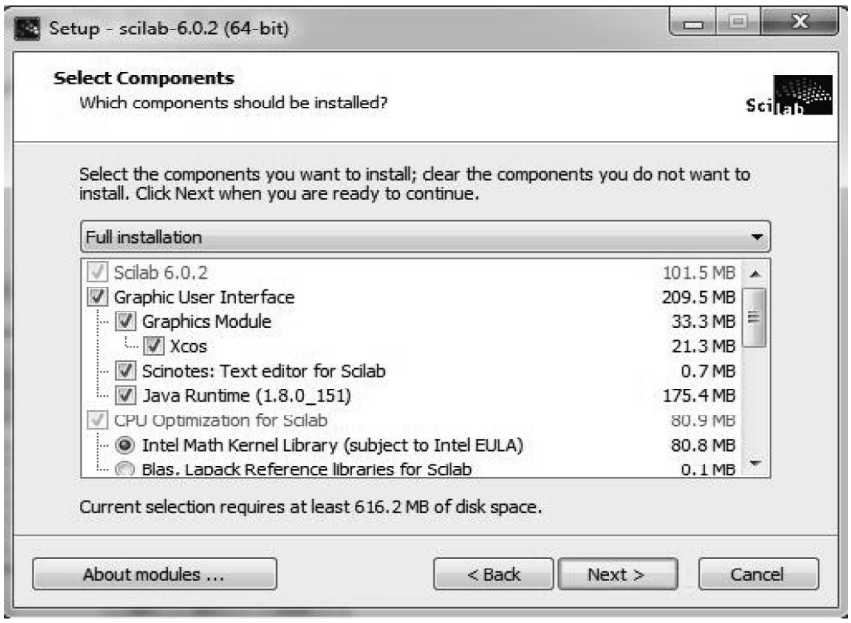
sure you have enough free disk space for Scilab installation. Click “Next” button.

Figure 8.6: Select Location



- Select the type of installation. “Full installation” will install the complete/full package of components/modules. Click “Next” button.

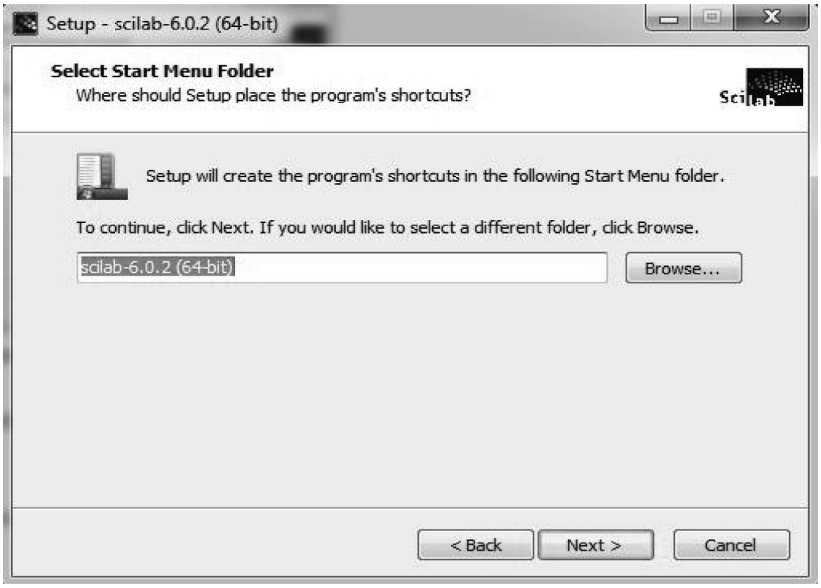
Figure 8.7: Type of Installation



- Select start menu folder. If you want a different location than

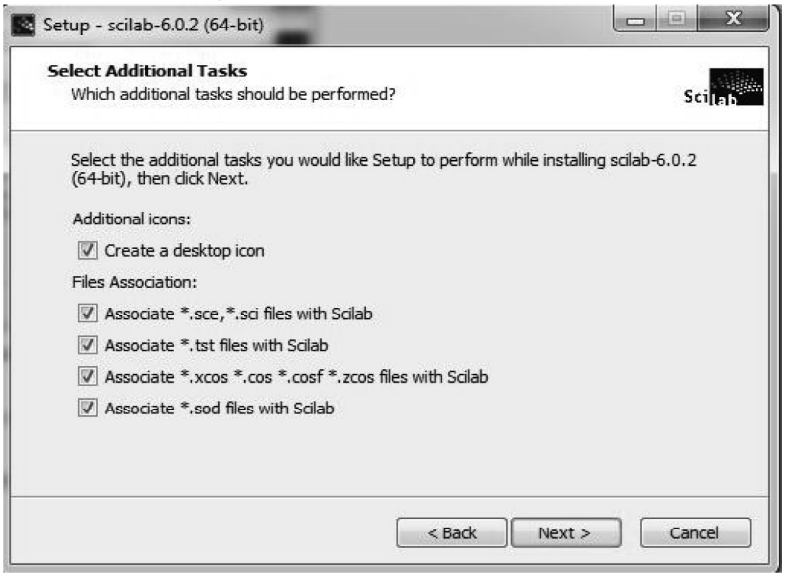
the default one, click the “Browse…” button. then click “Next” button.

Figure 8.8: Start Menu Folder



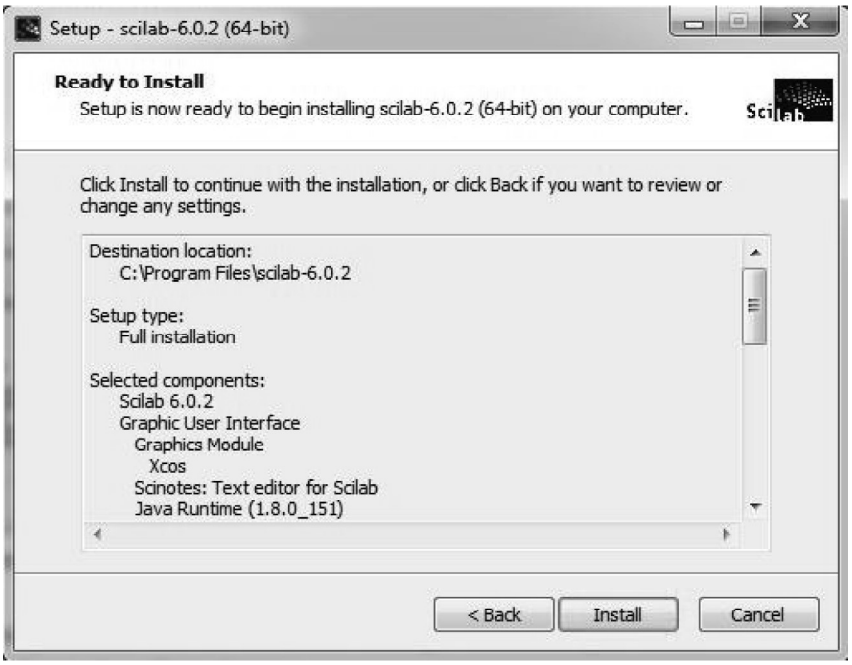
- Select additional tasks. If you want to create a Desktop icon make sure to check the box for “Create a desktop icon”. Check the three boxes for different file association. Then click “Next” button.

Figure 8.9: Additional Tasks



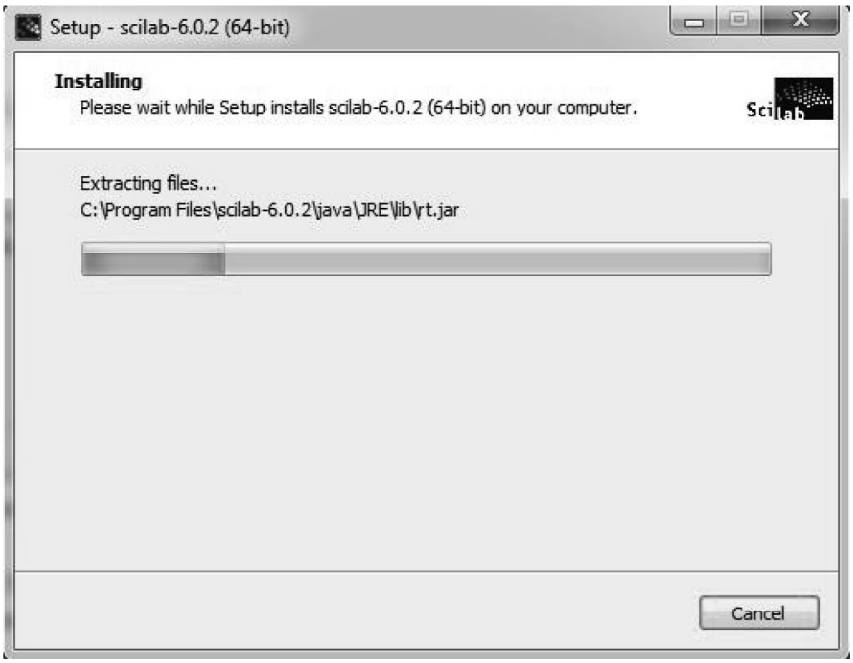
- Review the summary window with installation settings. If all the settings are as expected click the “Install” button.

Figure 8.10: Review summary



- Wait for the installation to complete until the progress bar fills up.

Figure 8.11: Installation progress bar



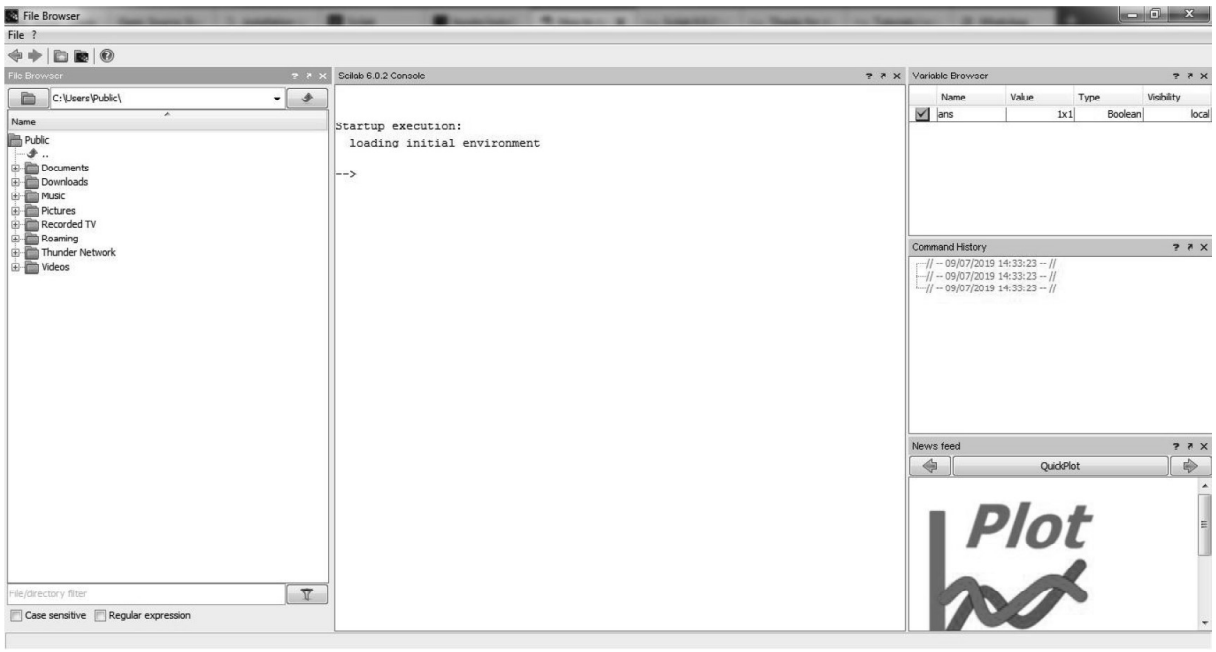
- Setup is finished. Then check the box “Launch Scilab” and click “Finish” button to launch Scilab.

Figure 8.12: Completion of Scilab Installation



- After Scilab is launched you get the start screen shown below.

Figure 8.13: Start Screen after Launch



8.3.2 Installation of Scilab in Linux

The steps for installation of Scilab in Linux are given below:

- Step 1: Get the binary file and put it in a local directory where you want to install Scilab (<scilab-path>).
- Step 2: Launch the following commands to get the binary file and install Scilab.
  - **For Linux 32 bits:**
    1) *[$SHELL] wget http://www.scilab.org/download/6.0.2/scilab-6.0.2.bin.linux-i686.tar.gz*
    2) *[$SHELL] cd <scilab-path>*
    3) *[$SHELL] tar xzvf scilab-6.0.2.bin.linux-i686.tar.gz*
  - **For Linux 64 bits:**
    1) *[$SHELL] wget http://www.scilab.org/download/6.0.2/scilab-6.0.2.bin.linux-x86_64.tar.gz*
    2) *[$SHELL] cd <scilab-path>*
    3) *[$SHELL] tar xzvf scilab-6.0.2.bin.linux-x86_64.tar.gz*
- Step 3: To launch Scilab.
  - *[$SHELL] cd <scilab-path>/scilab-5.5.2*
  - *[$SHELL] ./bin/scilab*



CHECK YOUR PROGRESS

Q 1: From where can we download Scilab Python for windows installation?

.....

.....

Q 2: What is the command to launch Scilab in Linux?

.....

.....



8.4 LET US SUM UP

- Scilab can be downloaded from the official Scilab website https://www.scilab.org/.
- For both Windows and Linux operating systems there are 64 and 32 bit packages for 64 and 32 bit machines.
- In Linux we can launch Scilab using the command "./bin/scilab".



8.5 FURTHER READINGS

1) Verma, A. K., (2018). SCILAB: A beginner's Approach.
2) Nagar, S., (2018). Introduction to Scilab: For Engineers and Scientists.



8.6 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: https://www.scilab.org/

Ans to Q No 2: ./bin/scilab



8.7 MODEL QUESTIONS

Q 1: What is the official Scilab website?

Q 2: How can Scilab be installed in Linux?

Q 3: Give the steps to install Scilab in Windows.

*** ***** ***

UNIT 9 : INTRODUCTION TO SCILAB

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Scilab

9.4 Basic Syntax

9.5 Mathematical Operators

9.6 Predefined Constants

9.7 Handling .sci Files

9.8 Installation of Additional Packages

- 9.8.1 Accessing ATOMS GUI
- 9.8.2 Installing Packages
- 9.8.3 Removing Packages

9.9 Let Us Sum Up

9.10 Further Readings

9.11 Answers to Check Your Progress

9.12 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define Scilab
- write basic syntax
- define mathematical operators
- define predefined constants
- handle .sci files
- install additional packages

9.2 INTRODUCTION

In the previous unit you had been introduced to installation of Scilab in both Windows and Linux environment. In this unit you will be introduced to basic syntax of Scilab. We will learn about the mathematical operators

and predefined constants in Scilab. We will also learn about handling .sci files and installing additional packages.

In the next unit you will be introduced to matrix operations and functions in Scilab. We will learn about built in functions, complex numbers, polynomials, vectors, matrix operations and handling these data structures using built in functions.

9.3 SCILAB

Scilab is a major alternative for Matlab. Similar to Matlab, Scilab uses matrix as the main data type. All the operations are performed with respect to the matrix. Scilab has many mathematical operators which we will be learning at the later part of the unit.

In the next section you will learn about the basic syntax and how to use the Scilab console.

9.4 BASIC SYNTAX

After we launch Scilab, Scilab environment consist of the following docked windows – File Browser, Console, Variable Browser, Command History and News feed.

In the console, after the prompt "—>", just typing a command and pressing the Enter key gives the corresponding result of the command. For example, if we type 2+3 we get the addition result in the default variable "**ans**" as 5.

```
—>  2+3
ans   =
      5.
```

Similarly, we can perform other mathematical operations too in the console.

```
—>  (2+10)*5
ans   =
      60.
```

Now, if we want to define a variable in Scilab, then we do the definition as given below.

```
-->  a=5
a     =
      5.
```

Here, in the above example, we have defined a variable "*a*" and assigned a value of 5 to it.



CHECK YOUR PROGRES

Q 1: What is Scilab an alternative to?

.....

.....

Q 2: What are the windows docked in a Scilab environment?

.....

.....

9.5 MATHEMATICAL OPERATORS

In Scilab, the mathematical operators available for mathematical operations are addition (+), subtraction (-), multiplication (*), division (/) and exponentiation (^).

For example:

addition (+)

```
-->     2+3
ans      =
         5.
```

subtraction (-)

```
-->     5-3
ans      =
         2.
```

multiplication (*)

```
-->     4*5
ans      =
        20.
```

division (/)

```
-->     6/4
ans      =
        1.5
```

exponentiation (^)

```
-->     2^3
ans      =
         8.
```

9.6 PREDEFINED CONSTANTS

In Scilab, there are some special variables whose values are predefined and they are known as predefined constants. These constants have values which cannot be changed. The character "%" is used to access these variables followed by the name of the variable.

Table 9.1: predefined Constants

Constant	Description/Properties	Value
%pi	Greek letter π, sin(%pi/2) = 1	3.1415927
%e	Euler's constant , log(%e) = 1	2.7182818
%eps	Greek letter ε	$2.2 \cdot 10^{-16}$
%i	Imaginary unit	square root of -1
%inf	Infinity	1/0
%nan	not a number	$\infty - \infty$
%t	True (boolean)	1
%f	False (boolean)	1
%s	Polynomial expression	s=poly(0,"s")
%z	Polynomial expression	z=poly(0,"z")



CHECK YOUR PROGRESS

Q 3: What are the available mathematical operators in Scilab?

.....

.....

Q 4: Define two predefined constants in Scilab.

.....

.....

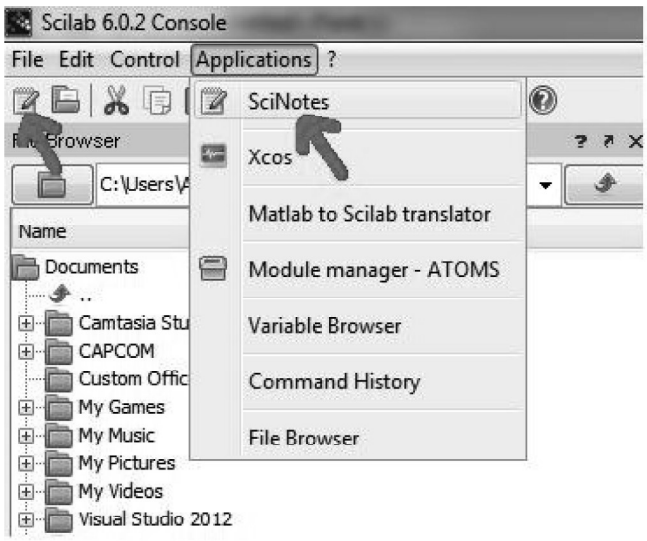
9.7 HANDLING .SCI FILES

While writing commands in the console it is not possible to save the commands and it is not easy to edit multiple lines of commands. A .sci file is a script used by Scilab, containing these commands or instructions. A .sci file basically is a function file with a function statement at the start. It can also have multiple function definitions containing multiple

Scilab commands performing operations on the function arguments.

To write multiple commands, the editor is the best tool available in Scilab. To open an editor in Scilab, just click the first icon in the toolbar. We can also open an editor from the Menubar, under “Applications” and select “SciNotes”.

Figure 9.1: Opening Scilab editor



The editor opens with a default file name “Untitled 1”. To save the file, we select from the Menubar the “File” option and then we select “Save as” option. We can now give a suitable name for the file and save the file with “.sci” extension.

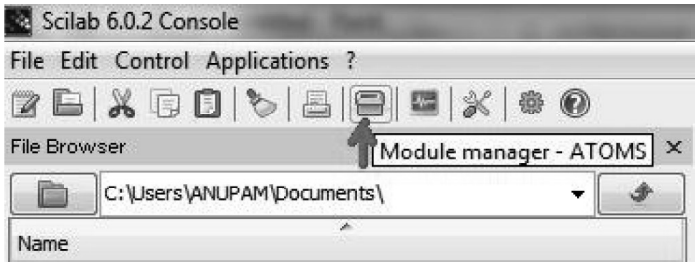
9.8 INSTALLATION OF ADDITIONAL PACKAGES

In Scilab we can add many additional packages or modules called as “Toolboxes” which have been created by the Scilab community. ATOMS (AuTomatic mOdules Management for Scilab) is the repository for these packaged extension modules. Scilab includes a GUI to facilitate the use of ATOMS.

9.8.1 Accessing ATOMS GUI

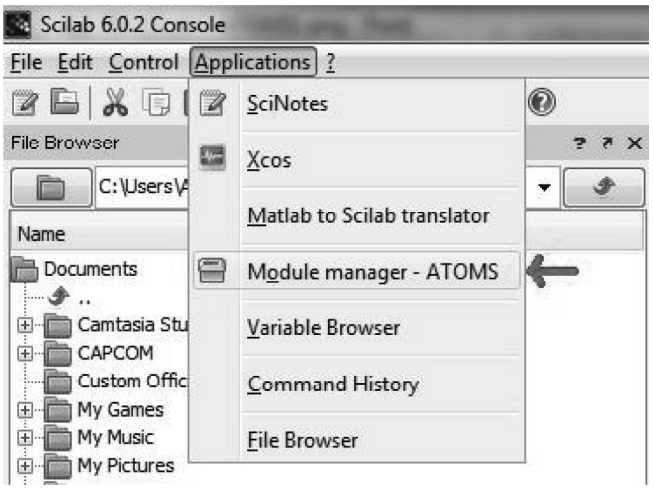
The ATOMS GUI is available through the Scilab using the icon shown below in figure 9.2.

Figure 9.2: ATOMS GUI



This GUI is also available through the Scilab menu "Applications".

Figure 9.3: Module manager ATOMS

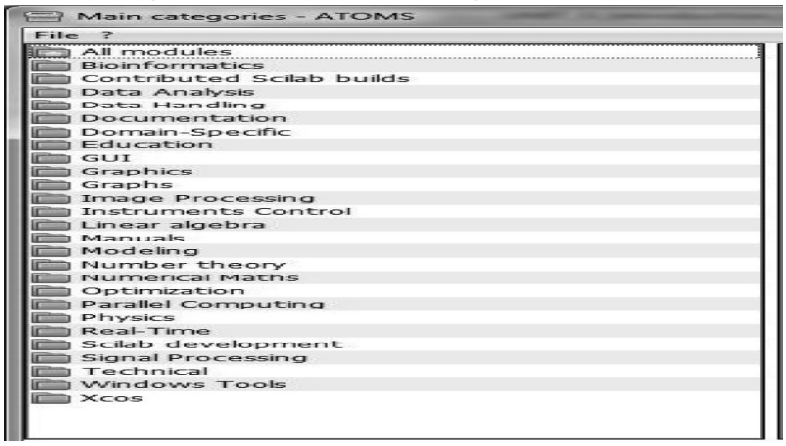


The GUI can also be accessed by entering the command "*atomsGui()*" in the console.

—> *atomsGui();*

The GUI gives the list of available packages for installation.

Figure 9.4: List of packages in ATOMS



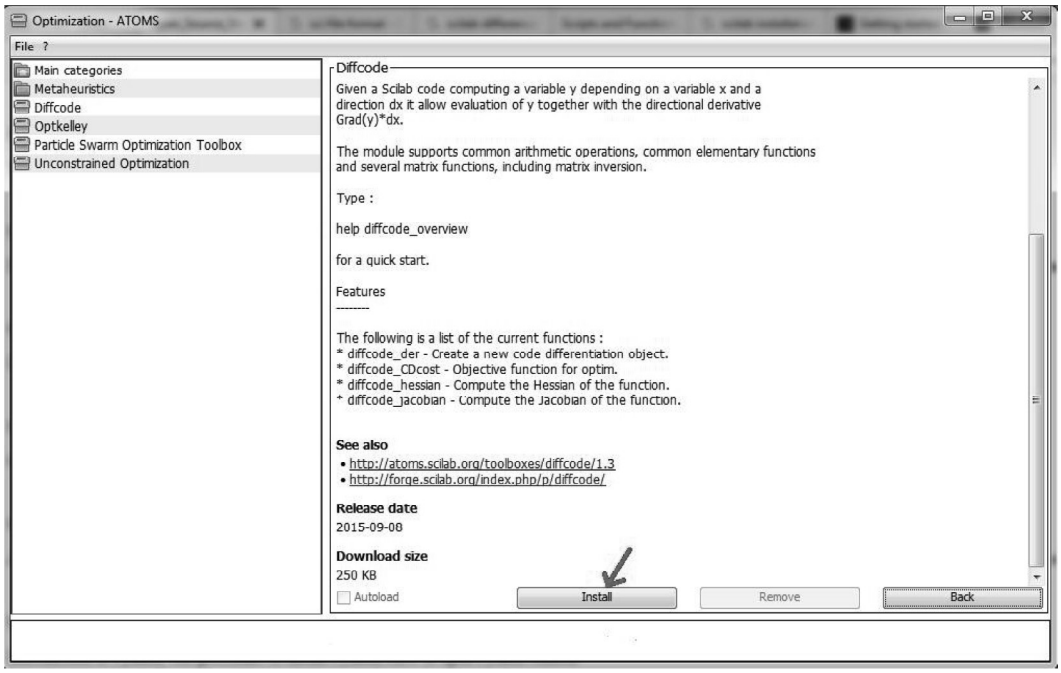
The list of available packages can also be displayed in console by entering the command "*atomsList();*" in the console.

—> *atomsList();*

9.8.2 Installing Packages

The packages can be installed using the GUI as shown below in figure 9.5. In this figure, we are installing package "Diffcode" available under Optimization packages. This is a Scilab code for computing a variable y depending on a variable x and a direction dx that allows evaluation of y together with the directional derivative Grad(y)*dx. For installing packages, we need to select the package to be installed and click "Install".

Figure 9.5: Installing Packages "diffcode"



The same package can be installed by entering the command "*atomsInstall('<package_name>')*" in the console.

```
—>  atomsInstall  ('Diffcode')
```

9.8.3 Removing Packages

For removing a package in GUI we need to select the

UNIT 10 : MATRIX AND BUILT-IN FUNCTIONS

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

10.3 Matrix

10.4 Built-in Functions

- 10.4.1 Length()
- 10.4.2 Size()
- 10.4.3 Zeros()
- 10.4.4 Ones()
- 10.4.5 Eye()
- 10.4.6 Rand()
- 10.4.7 Inv()
- 10.4.8 Spec()

10.5 Complex Numbers

10.6 Polynomials

10.7 Let Us Sum Up

10.8 Further Readings

10.9 Answers to Check Your Progress

10.10 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- represent matrix and vectors
- use the different build in functions available in Scilab
- represent complex numbers in Scilab
- represent polynomials in Scilab

10.2 INTRODUCTION

In the previous unit you have been introduced to basic syntax of Scilab. We also know how to use the Scilab console. In this unit you will

learn about matrix operations and functions in Scilab. You have also learn in detail about different built in functions that can be used on matrixes. You will also learn how to represent complex numbers and polynomials in Scilab. In the next unit, you will learn about the different conditional statements and loops that can be used in Scilab. You will learn about conditional statements like if else, ifelseif, select and looping statements like for loop, while loop etc.

10.3 MATRIX

You are already familiar with the Scilab console and know how to perform basic mathematical operations. Scilab is very powerful at computations with variables such as vectors and matrices. It has a lot of built-in functions that allows the user to perform complex computations and manipulations on vectors (1-D matrices) and matrices.

Let us look at how the matrixes are defined in Scilab. In the example below, we have defined a matrix variable “p” which contains three values.

```
-> a = [1 2 3];
```

Matrixes can also be defined by using comma’s between the values. For example,

```
-> b = [4, 5, 6];
```

Two dimensional matrixes can also be defined in Scilab. For example,

```
-> c = [ 1 2 3; 4 5 6; 7 8 9];
```

If we now output the values of the variable “c” it will be represented in Scilab as the following.

```
-> c
 c  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

We can perform different mathematical functions on two or more matrixes. For example, we perform addition of two matrixes “c” and “d”

and put the sum in a third matrix “add” below.

```
-> c = [ 1 2 3; 4 5 6; 7 8 9];
-> d = [ 1 2 3; 1 2 3; 1 2 3];
-> add= c + d
 add   =
    2.    4.    6.
    5.    7.    9.
    8.    10.   12.
```

We can also find the transpose of any vector or matrix by using apostrophe. For example, the transpose of matrix “add” is stored in matrix “add_t” below.

```
-> add_t = add'
 add_t   =
    2.    5.    8.
    4.    7.    10.
    6.    9.    12.
```

10.4 BUILT-IN FUNCTIONS

Scilab has a large range of build-in functions that can be used on matrixes. Let us look at some of the most commonly used such build-in functions.

10.4.1 Length()

We can find the length of any vector in Scilab by using the built-in function **length**(). For example, the vector “a” defined below has three entries, and hence the length of the vector “a” is returned as 3.

```
-> a = [1 2 3];
-> length(a)
 ans   =
    3.
```

10.4.2 Size()

We can find the size of any matrix in Scilab by using the built-in function **size()**. For example, the matrix "x" defined below has two columns and three rows, and hence the size of the marix "x" is returned as (3,2). The first result of size() represents the number of rows and the second result represents the number of columns.

```
—> x = [ 1 2; 3 4; 5 6]
 x  =
   1.   2.
   3.   4.
   5.   6.
—>  size(x)
 ans  =
   3.   2.
```

We can create some special matrixes in Scilab by using certain build-in functions. Let us look at some of these functions.

10.4.3 Zeros()

We can create a special matrix consisting of only zeros by using the function **zeros()**. For example, we can create a matrix of 3 rows and 4 columns consisting only of zero values by simply using the built in function zeros(3,4). For functions like these, the first entry always represents the number of rows and the second entry represents the number of columns.

```
—>  zeros(3,4)
  ans    =
     0.     0.     0.     0.
     0.     0.     0.     0.
     0.     0.     0.     0.
```

10.4.4 Ones()

We can create another special matrix consisting of only

ones by using the function **ones()**. Similar to the zeros() function, if we want to create a matrix of 2 rows and 3 columns consisting of entries with values only “1” then we can do this by simply using the built in function ones(2,3). For this function also, the first entry always represents the number of rows and the second entry represents the number of columns.

```
—>  ones(2,3)
  ans    =
     1.     1.     1.
     1.     1.     1.
```

10.4.5 Eye()

We can create identity matrixes by using the function **eye()**. Similar to the earlier two functions, if we want to create an identity matrix by simply specifying the number of rows and columns using the built in function **eye()**. For example, if we want to create an identity function for a 3*3 square matrix then we would simply type eye(3,3) in the Scilab console.

```
—>  eye(3,3)
  ans    =
     1.     0.     0.
     0.     1.     0.
     0.     0.     1.
```

10.4.6 Rand()

We have now learned that we can create matrixes with values of only zeros or ones or simply identity matrixes. Apart from these, we can also create matrixes with random values. For creating random value matrixes we simply need to use the build-in function **rand()**. For example, if we want to create a random matrix of 2 rows n 5 column then we can simply write the function rand(2,5).

```
—>  rand(2,5)
```

```
 ans    =
0.2113249  0.0002211  0.6653811 0.8497452 0.8782165
0.7560439  0.3303271  0.6283918 0.685731  0.068374
```

10.4.7 Inv()

The **inv()** function is used to find the inverse of a square matrix. A warning message is printed if the matrix is badly scaled or nearly singular. For example, we have found the inverse of a matrix below.

```
->  A=rand(3,3)
 A    =
     0.4818509     0.2806498     0.211903
     0.2639556     0.1280058     0.1121355
     0.4148104     0.7783129     0.6856896
—>  inv(A)*A
 ans    =
       1.             0.         0.
       -1.305D-15     1.         0.
       0.             2.220D-16  1.
```

10.4.8 Spec()

The **spec()** function is used to find the Eigen values of a matrix. We can find the Eigen values of a matrix like A in the example below.

```
—>  A=[2,1,1;1,3,2;3,1,4]
 A    =
    2.     1.     1.
    1.     3.     2.
    3.     1.     4.
—>  spec(A)
 ans    =
    6.095824
    1.452088  +  0.4336988i
```



CHECK YOUR PROGRESS

Q 1: How are matrixes defined in Scilab?

.....

.....

Q 2: Give the purpose of using rand() function.

.....

.....

Q 3: How can we find the transpose of a matrix?

```
1.452088  -  0.4336988i
```

.....

.....

10.5 COMPLEX NUMBERS

Complex numbers are numbers which contain a pair of real numbers with a imaginary part. Complex numbers can be represented as follows:

c = a + b*i

where, c is the complex number

a is the real number

b is the real number

i is the imaginary part

If the real number (a) is zero, the complex number is purely imaginary. Also, if the real number (b) is zero, the complex number becomes a real number. In Scilab, we define the complex numbers by using the special constant **%i**. For example, a complex number c which has real part 4 and imaginary part 5i can be represented as follows.

```
-> c = 4 + 5*%i
c  =
   4. + 5.i
```

10.6 POLYNOMIALS

Polynomials can be defined as an mathematical expression of

determined length which contains operations between variables and constants (coefficients). Generally, polynomials can be expressed as follows:

$$\mathbf{p(x) = a_0x^0 + a_1x^1 + \ldots + a_nx^n}$$

where, a_n are real numbers ($a_n \in R$) that represent the coefficients of the polynomial

x is a symbolic variable of the polynomial

n is a natural number ($a_n \in N$) that represents the exponents of the polynomial.

In Scilab, polynomials can be defined using the built-in function **poly().** The syntax of the Scilab poly() function is:

```
p = poly(data, 'var', 'options')
```

where,

data is a vector or real number that represents the coefficients or the roots of the polynomial,

'var' is a string that represents the symbolic variable name of the polynomial,

'options' are string that represents the type of the polynomial definition; the possible values are predefined as: 'roots' (short 'r'), default value, for the definition of the polynomial based on its roots or 'coeff' (short 'c'), for the definition of the polynomial based on its coefficients

p is the variable defined as a polynomial.

Now, if we want to represent a polynomial which has roots: $x_1 = 2$ and $x_2 = -2$, then it can be represented by the following example.

```
-> p=poly([2, -2],'x','r')
 p    =
            2
   -4  +x
```

The result of the above polynomial definition is: $-4 + x^2$

Let us try to define the polynomial: $4 + 5x - 6x^2 + 7x^3$. Below, we have represented the polynomial using the coefficients:

$a_0 = 4, a_1 = 5, a_2 = -6$ and $a_3 = 7$.

```
-> p=poly([4  5  -6  7],'x','c')
```

```
 p  =
                        2       3
      4  +5x  -6x    +7x
```

The roots of a polynomial can be found by using the function roots(). For example, let us try to find the roots of the polynomial p, where p is represented as $p = -4 + x^2$. For this, we can simply type the following in the Scilab console.

```
—>  poly1  =  poly([-4  0  1],'x','c')
  poly1    =
             2
    -4  +x
—>  roots(poly1)
  ans    =
     2.
```



CHECK YOUR PROGRESS

Q 4: How are complex numbers represented in Scilab?

.....

.....

Q 5: Give the syntax of the poly() function.

.....

.....



10.7 LET US SUM UP

- Scilab is very powerful at computations with variables such as vectors and matrices.
- Matrixes can also be defined by using commas between the values.
- We can afind the transpose of any vector or matrix by using apostrophe.
- The length of any vector in Scilab is found by using the built-in

function length().

- We can create a special matrix consisting of only zeros by using the function zeros().
- Identity matrixes n be created by using the function eye().
- For creating random value matrixes we simply need to use the build-in function rand().
- The inv() function is used to find the inverse of a square matrix.
- The spec() function is used to find the Eign values of a matrix.
- In Scilab we define the complex numbers by using the special constant **%i**.
- Polynomials can be defined as an mathematical expression of determined length which contains operations between variables and constants (coefficients).



10.8 FURTHER READINGS

1) Verma, A. K., (2018). *SCILAB: A beginner's Approach*.

2) Nagar, S., (2018). *Introduction to Scilab: For Engineers and Scientists*.



10.9 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: A matrix variable "p" which contains three values can be defined as follows.

—> a = [1 2 3];

Ans to Q No 2: The rand() function is used for creating random value matrixes.

Ans to Q No 3: We can also find the transpose of any vector or matrix by using apostrophe symbol after the matrix.

Ans to Q No 4: In Scilab, complex numbers are represented by using the special constant **%i**. For example, a complex number c which

has real part 3 and imaginary part 6i can be represented as c = 3 + 6*%i.

Ans to Q No 5: The syntax of the Scilab poly() function is: p = poly(data, 'var', 'options')

where,

data is a vector or real number that represents the coefficients or the roots of the polynomial,

'var' is a string that represents the symbolic variable name of the polynomial,

'options' are string that represents the type of the polynomial definition; the possible values are predefined as: 'roots' (short 'r'), default value, for the definition of the polynomial based on its roots or 'coeff' (short 'c'), for the definition of the polynomial based on its coefficients and

p is the variable defined as a polynomial.



10.10 MODEL QUESTIONS

Q 1: Describe how vectors and matrixes are represented in Scilab.

Q 2: Describe in brief any five built-in functions related to matrixes.

Q 3: Give an example using inv() function.

Q 4: Give an example using spec() function.

Q 5: How are rand(), ones() and zeros() functions useful in real life applications?

Q 6: How are complex numbers represented in Scilab?

Q 7: Give an example for finding roots of a polynomial.

Q 8: Show addition of any two polynomials in Scilab.

Q 9: How are polynomials represented in Scilab?

Q 10: Represent the polynomial, $p = 4x^4 - 3x^3 + 2x^2 - x + 15$ in Scilab.

*** ***** ***

UNIT 11 : CONDITIONAL STATEMENTS IN SCILAB

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Conditional Statements

- 11.3.1 if statement
- 11.3.2 if else statement
- 11.3.3 if elseif statement
- 11.3.4 select statement
- 11.3.5 break statement
- 11.3.6 continue statement

11.4 Loops

- 11.4.1 for loop
- 11.4.2 while loop

11.5 Let Us Sum Up

11.6 Further Readings

11.7 Answers to Check Your Progress

11.8 Model Questions

11.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- use if, if-else and if-elseif statements in scilab programs
- use break and continue in scilab programs
- use select statements in scilab programs
- use while loops in scilab programs
- use for loops in scilab programs

11.2 INTRODUCTION

In the previous units, you had been introduced to basic syntax of Scilab. You also learnt about matrix operations and functions in Scilab.

You had learned about built in functions, complex numbers, polynomials, vectors, matrix operations.

In this unit, you will learn about the different conditional statements and loops that can be used in Scilab. You will learn about conditional statements like if else, ifelseif, select etc. Apart from these, you will also learn in detail how to use loop statements like for loop and while loop in Scilab. In the next unit you will be introduced to graphics in Scilab, handling 2D and 3D graphics, generating .jpg files, plotting functions, data plotting, creating GUI in Scilab and plotting 2D graphs.

11.3 CONDITIONAL STATEMENTS

You are already familiar with the Scilab console and know how to perform basic mathematical operations and matrix operations as well. Generally programs and instructions in all languages like C, Java are executed sequentially. This is true for Scilab also. However, this happens when no conditions are put around the instructions or statements. Sometimes we may want to put certain conditions or restrictions on the flow of execution. This can be done with the help of decision making or conditional statements.

Let us look at the different conditional statements that can be used in Scilab. Some of the conditional statements are listed below.

- IF ELSE
- IF ELSEIF
- SELECT CASE
- break
- continue

We will look at each of these conditional statements in detail now.

11.3.1 IF conditional statement

The IF conditional statement is used in Scilab when we need to put certain conditions around instructions. It is one of the simplest conditional statements in Scilab. The general structure

for IF statements is given below.

General structure:

```
if condition_1 then
      instructions_1
end
```

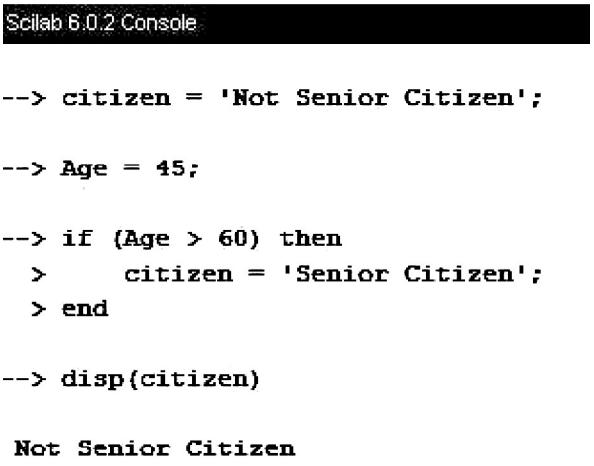
In the above general structure given for IF, the statement is started by using the keyword **if** followed by the condition. Once the condition is stated it is followed by the keyword **then.** The instructions which are to be followed when the condition is true are to be specified between the keyword **if** and keyword **end**. The **end** keyword specifies the ending of the conditional statements. An example of **if** statement bold shown in example 11.1.

Example 11.1: Example showing use of If statement

```
citizen = 'Not Senior Citizen';
Age = 45;
if (Age > 60) then
      citizen = 'Senior Citizen';
end
disp(citizen)
```

In the above example, we have first defined a variable "citizen" with value "`Not Senior Citizen`" and a variable "age" with value "45". In the following statements the variable age is compared with the value 60 and if the condition is true the next line will be executed where the value of variable citizen will be modified to "`Senior Citizen`". In this example, since the value in variable **"age"** is not greater than **"60"** so the **if** statement will not be executed and the value in the variable "citizen" will remain "`Not Senior Citizen`". The output is shown in the figure 11.1.

Figure 11.1: Output of example 11.1



11.3.2 IF ELSE conditional statements

IF ELSE is one of the most commonly used conditional statements in Scilab. The general structure for IF ELSE statements is given below.

General structure:

```
if  condition_1  then
 instructions_1
else
 default  instructions
end
```

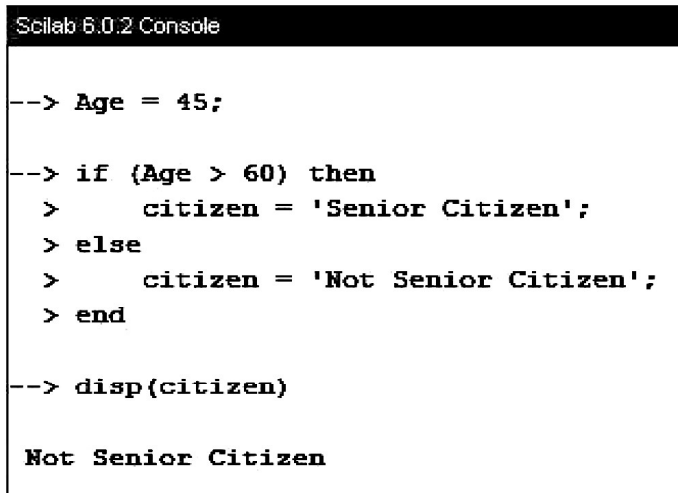
In the above general structure given for IF ELSE, the statement is started by using the keyword **if** command followed by the condition. Once the condition is stated it is followed by the keyword **then.** This is followed by the keyword **else**. The series of instructions that are to be followed when this condition is true are to be specified between the keyword **then** and keyword **else**. The instructions which are to be followed when the condition is not true are to be specified between the keyword **else** and keyword **end**. The **end** keyword specifies the ending of the conditional statements. An example for if else statement is given in example 11.1.

Example 11.2: Example showing use of If Else statement

```
Age = 45;
if (Age > 60) then
    citizen = 'Senior Citizen';
else
    citizen = 'Not Senior Citizen';
end
disp(citizen)
```

In the above example, we have first defined a variable “age” with value “45”. In the following statements the variable age is compared with the value 60 and if the condition is true the next line will be executed. Otherwise, the statements after the else keyword will be executed. In this example the value that will be stored in the variable “citizen” will be “`Not Senior Citizen`”. The output is shown in the figure 11.2.

Figure 11.2: Output of example 11.2



11.3.3 IF ELSEIF conditional statements

The general structure for IF ELSEIF statements is given below.

General structure:

```
if condition_1 then
    instructions_1
elseif condition_2 then
```

```
     instructions_2
…….
elseif  condition_N  then
     instructions_N
else
     default  instructions
end
```

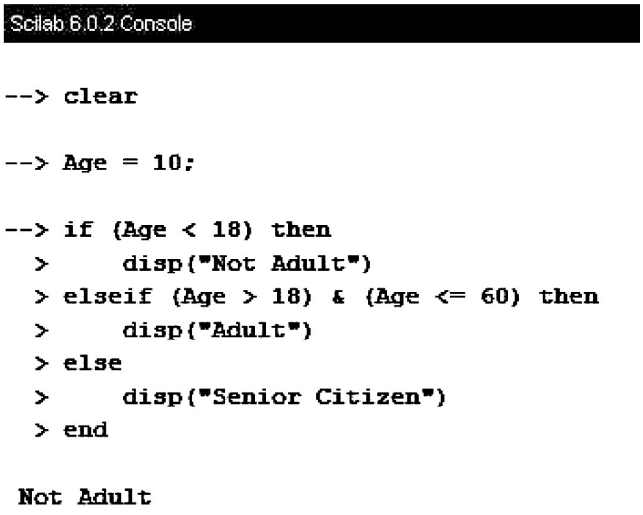
In the above general structure given for IF ELSEIF, the statement is started by using the keyword **if** command followed by the condition. Multiple conditions can be stated using **elseif** statements. The instructions which are to be executed when the particular condition is true are specified between the keyword **then** and the next **elseif** or **else** keyword. The **end** keyword specifies the ending of the elseif conditional statements. An example for elseif statement is given in example 11.3.

Example 11.3: Example showing use of If Elseif statement

```
Age  =  10;
if  (Age  <  18)  then
     disp("Not  Adult")
elseif  (Age  >  18)  &  (Age  <=  60)  then
     disp("Adult")
else
     disp("Senior  Citizen")
end
```

In the above example, we have first defined a variable "age" with value "10". In the following statements the variable age is first compared with the value 18. Since this condition is true so the variable will not be compared anymore. The output is shown in the figure 11.3.

Figure 11.3: Output of example 11.3



11.3.4 SELECT CASE conditional statements

The general structure for SELECT CASE statements is given below.

General Structure

```
select  controlVariable
case  value_1  then  instructions_1
case  value_2  then  instructions_2
…
case  value_N  then  instruction_N
else  defaultInstructions
end
```

The current value of the control variable (controlVariable) is evaluated and compared with each of the cases starting with the first one (value_1). When a match case is found, the corresponding instruction is executed.

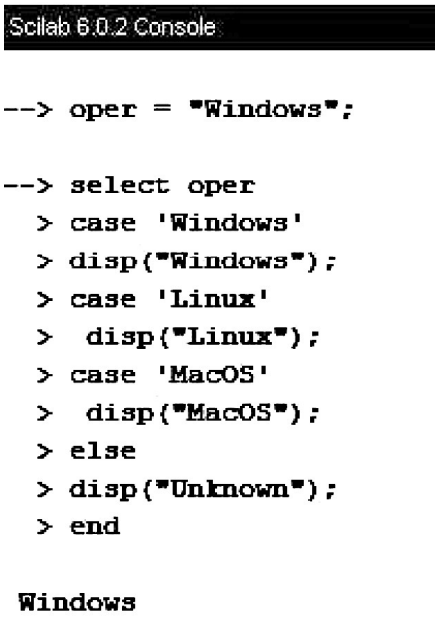
Example 11.4: Example showing use of Select case statement

```
oper  =  "Windows";
select  oper
case  'Windows'
disp("Windows");
case  'Linux'
```

```
  disp("Linux");
case  'MacOS'
disp("MacOS");
else
disp("Unknown");
end
```

In the above example, we have first defined a variable "oper" with value "Windows". In the following statements the variable "oper" is compared with different values using different case statements. Since the value in the first case condition matches with the variable "oper", so the set of instructions for that case are executed. The output is shown in the figure 11.4.

Figure 11.4: Output of select case example 11.4



11.3.5 Break Statement

The general structure for **break** is given below.

General Structure

break

Let us look at an example to understand the use of break statement. Inside a **for** or **while** loop, the command bold break forces the end of the loop. This is shown in example 11.5.

Example 11.5: Example showing use of break statement

```
k=0;
while  1==1,
k=k+1;
if  k  >  100  then
break
end;
end
```

In the above example, we have first defined a variable “k” with value “0”. In the following statements the variable “k” is incremented and then compared if it is greater than 100. The loop stops only when the value of k gets bigger than 100 and executes the break statement.

11.3.6 Continue Statement

The general structure for **continue** is given below.

General Structure

Let us look at an example to understand the use of continue statement. In example 11.6, inside a for or while loop, the command break forces the end of the loop. Inside a for or while loop, the command **continue** passes control to the next iteration of the loop in which it appears, skipping any remaining statements between this instruction and the loop’s end instruction. This is shown in example 11.6.

Example 11.6: Example showing use of continue statement

```
for  k=1:10
if  k>2  &  k<=8  then
continue
disp('hello')
end
k
end
```

In the above example, when the values of the variable “k” has values greater than 2 and lesser than 8 then the if statement will be true and since there is a continue statement inside the if condition so, the control will then ignore the execution sequence and control will jump to the next iteration of the for loop. So, the values of k will be output for 1, 2, 9 and 10 only.

Figure 11.5: Output of example 11.6

```
Scilab 6.0.2 Console

-1-> for k=1:10
  > if k>2 & k<=8 then
  > continue
  > disp('hello')
  > end
  > k
  > end
 k  =

   1.

 k  =

   2.

 k  =

   9.

 k  =

   10.
```



CHECK YOUR PROGRESS

Q 1: What are conditional statements?

.....

.....

Q 2: Give the syntax of if else statements.

.....

.....

11.4 LOOP STRUCTURES

Conditional statements are used to execute specific instructions when certain conditions are true. Similarly, if we want to execute a particular instruction for a given number of times then we can use loops or looping statements. The looping statements evaluate an instruction repeatedly until a condition is satisfied. Conditional loop is to execute a set of instructions repeatedly until a condition is satisfied.

Let us look at the different loop statements that can be used in Scilab. Some of them are listed below.

- For Loop
- While Loop

11.4.1 For Loop

The general structure of **for loop** is given below.

General Structure

```
for  expression
     instructions
end
```

The expression contains a set of Scilab operations which perform:

- variable initialisation
- variable incrementation each time the instruction is executed and
- variable comparison (condition)

The instruction is the piece of code that is evaluated each time the control variable is incremented as long as the condition within the expression is satisfied.

Example 11.7: Example showing use of for loop structure

```
for  x=1:1:5
disp(x);
end
```

In the above example, in the first line, the variable x is first initialized to one and then the variable is to incremented by one till it reaches the value of five. Till the condition is true i.e. till the variable reaches the value of 5 the instructions within the loop are executed. So value of 1 2 3 4 5 will be shown as output. Once the value of the variable x is incremented to 6, the condition for the loop to execute is false and hence the iteration stops.

11.4.2 While Loop

` The general structure for **while loop** is given below.

General Structure

```
while  condition
     instructions
end
```

In while loop, the condition is usually a comparison of a variable to a constant. The instructions are a set of Scilab operations that are executed as long as the condition is true.

Example 11.8: Example showing use of while loop structure

```
x  =  1;
while  x<=5  do
disp(x);
x  =  x  +  1;
end
```

In the above example, in the first line, the variable x is first initialized to one. In the second line the condition is given to check whether the value of the variable x is less than or equal to 5. If the condition is true, the set of instructions that are given inside the while loop are executed sequentially. First the value of variable x is shown as output and then the value of the variable is incremented by one till the condition remains true. Once the value of the variable x is incremented to 6, the condition for the while loop is false and hence the iteration stops.



CHECK YOUR PROGRESS

Q 3: What are loop statements?

.....

.....

Q 4: Give the syntax of for loop.

.....

.....



11.5 LET US SUM UP

- Conditional statements are statements with the help of which we can put certain conditions or restrictions on the flow of execution.
- In select case, the current value of the control variable (controlVariable) is evaluated and compared with each of the cases starting with the first one (value_1). When a match case is found, the corresponding instruction is executed.
- The looping statements are statements that evaluate an instruction repeatedly until a condition is satisfied.
- The instruction is the piece of code that is evaluated each time the control variable is incremented as long as the condition within the expression is satisfied.
- In while loop, the condition is usually a comparison of a variable to a constant. The instructions are a set of Scilab operations that are executed as long as the condition is true.



11.6 FURTHER READINGS

1) Verma, A. K., (2018). *SCILAB: A beginner's Approach*.

2) Nagar, S., (2018). Introduction to Scilab: For Engineers and Scientists.



11.7 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Conditional statements are statements with the help of which we can put certain conditions or restrictions on the flow of execution.

Ans to Q No 2: The syntax for if else is given below:

```
if condition_1 then
        instructions_1
else
        default instructions
end
```

Ans to Q No 3: The looping statements are statements that evaluate an instruction repeatedly until a condition is satisfied.

Ans to Q No 4: The syntax of for loop is given below:

```
for expression
        instructions
end
```

The expression contains a set of Scilab operations which perform:

- variable initialisation
- variable incrementation each time the instruction is executed and
- variable comparison (condition)



11.8 MODEL QUESTIONS

Q 1: Describe any two conditional statements with syntax and example.

Q 2: Give an example using if-elseif statements.

Q 3: Give an example using if-else statements.

Q 4: Give an example using select statements.

Q 5: What is the difference between continue and break? Explain using

example.

Q 6: Define loop statements. Why do we need loop statements?

Q 7: Give an example using while loop statements.

Q 8: Give an example using for loop statements.

Q 9: Give an example using continue statements.

Q 10: Explain the need for conditional statements.

*** ***** ***

UNIT 12 : GRAPHICS IN SCILAB

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Graphics handling

12.3.1 2D Graphics

12.3.2 3D Graphics

12.4 Generating .jpg files

12.5 Function Plotting

12.6 Data Plotting

12.7 GUI in Scillab

12.8 Plotting 2D Graphs

12.9 Let Us Sum Up

12.10 Further Readings

12.11 Answers to Check Your Progress

12.12 Model Questions

12.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- handle 2D and 3D graphics
- generate .jpg files
- plot functions
- plot data
- create GUI in Scilab
- plot 2D graphs

12.2 INTRODUCTION

In the previous unit you had been introduced to conditional statements and loops in Scilab. In this unit you will be introduced to Graphics in Scilab. We will learn about handling 2D and 3D graphics, generating .jpg files, plotting functions, data plotting, creating GUI in Scilab

and plotting 2D graphs.

In the next unit you will be introduced to applications of Scilab. You will learn about numerical linear algebra by solving linear equations, eigen values, etc. We will also look into solving ordinary differential equations, numerical analysis using iterative methods and comparison of Scilab with C/C++/Matlab.

12.3 GRAPHICS HANDLING

In Scilab, Graphics is used to handle and produce many types of graphs. Scilab native functions are used to plot these graphs. Scilab offers many ways to create and customize various types of 2D and 3D plots.

12.3.1 2D Graphics

2D Graphics are handled using 2D plot functions available in Scilab. Some of these functions are shown in table 12.1

Table 12.1: Some 2D Graphics Functions

Function	Description
LineSpec	to quickly customize the lines appearance in a plot
Matplot	2D plot of a matrix using colors
Matplot1	2D plot of a matrix using colors
Matplot properties	Description of the Matplot entities properties
Sfgrayplot	smooth 2D plot of a surface defined by a function using colors
Sgrayplot	smooth 2D plot of a surface using colors
champ	2D vector field plot
champ1	2D vector field plot with colored arrows
champ properties	description of the 2D vector field entity properties
comet	2D comet animated plot.
contour2d	level curves of a surface on a 2D plot

contour2di contour2dm	compute level curves of a surface on a 2D plot compute level curves of a surface defined with a mesh
contour	filled level curves of a surface on a 2D plot
errbar	add vertical error bars on a 2D plot
fchamp	direction field of a 2D first order ODE
fec	pseudo-color plot of a function defined on a mesh
fec properties	description of the fec entities properties
fgrayplot	2D plot of a surface defined by a function using colors
fplot2d	2D plot of a curve defined by a function
grayplot	2D plot of a surface using colors
grayplot properties	description of the grayplot entities properties
graypolarplot	Polar 2D plot of a surface using colors
histplot	plot a histogram
paramfplot2d	animated plot of a 2D parameterized curve.
plot	2D plot
plot2d	2D plot
plot2d2	2D plot (step function)
plot2d3	2D plot (vertical bars)
plot2d4	2D plot (arrows style)
polarplot	Plot polar coordinates
scatter	2D scatter plot

12.3.2 3D Graphics

3D Graphics are handled using 3D plot functions available in Scilab. Some of these functions are shown in Table 12.2.

Table 12.2: 3D Graphics Functions

Function	**Description**
comet3d	3D comet animated plot.
contour	level curves on a 3D surface
eval3d	values of a function on a grid => **this function is obsolete**

eval3dp fac3d	compute facets of a 3D parametric surface 3D plot of a surface. **This function is obsolete.**
fplot3d	3D plot of a surface defined by a function
fplot3d1	3D gray or color level plot of a surface defined by a function
genfac3d	Compute facets of a 3D surface
geom3d	projection from 3D on 2D after a 3D plot
hist3d	3D representation of a histogram
mesh	3D mesh plot
nf3d	Rectangular facets to plot3d parameters
param3d1	plots a set of curves in a 3D cartesian frame. 1D mesh plot
param3d	properties description of the 3D curves entities properties
plot3d	3D plot of a surface
plot3d1	3D gray or color level plot of a surface
plot3d2	plot surface defined by rectangular facets
plot3d3	mesh plot surface defined by rectangular facets
scatter3	3D scatter plot
secto3d	3D surfaces conversion
surf	3D surface plot
surface properties	description of the 3D entities properties

12.4 GENERATING .jpg FILES

In Scilab, we can generate .jpg files from the graphic window by using the native function *xs2jpg. xs2jpg* exports the display of a graphic window into a JPG file.

```
xs2jpg(win_num, file_name [, compression_quality])
xs2jpg(fig, file_name [, compression_quality])
```

where, win_num → an integer, ID of the figure to export

fig → handle of the figure to export

file_name → a string, name of the exported file

compression quality → a real scalar, a number between 0 and 1

The default compression quality is set to 0.95.

Example 12.1: Using XS2 jpg function

```
scf(0);    // set the current graphic figure (window)
plot2d(); // 2D plot
//JPG export
xs2jpg(0,'foo.jpg'); // export as filename foo.jpg
xs2jpg(gcf(),'foo.jpg');
// gcf() returns handle of current graphic window
xs2jpg(0,'foo_0.jpg', 0);
// poor quality since compression quality=0
xs2jpg(gcf(),'foo_1.jpg', 1);
// best quality since compression quality=1
```



CHECK YOUR PROGRESS

Q 1: Name two 2D plot and two 3D plot functions.

.....

.....

Q 2: What is the function used to generate .jpg files?

.....

.....

12.5 FUNCTION PLOTTING

In Scilab, function *fplot2d* can be used to plot a 2D plot of a curve defined by a function.

```
fplot2d (xr, f, [style, strf, leg, rect, nax])
fplot2d (xr, f, <opt_args>)
```

Example 12.2: Say for an instance to define a function

f(x) = x |x|/(1+x2)

in the console we would write

```
deff('[y] = f(x)',['y = x*abs(x)/(1+x^2)']);
```

This will define a Scilab function with the name f.

Now to plot the function f in the interval [0,1] we use

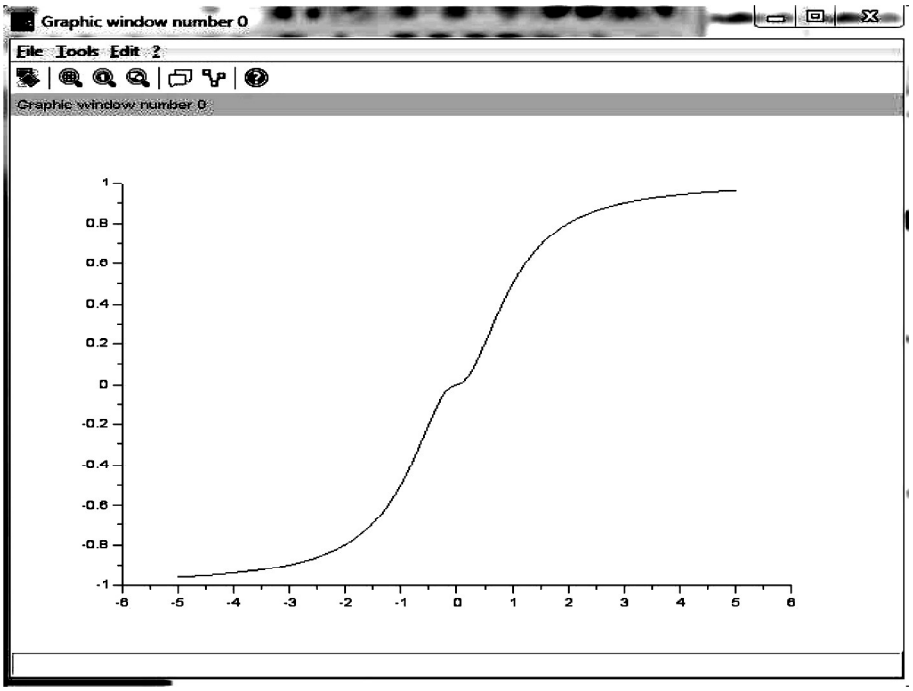
```
x = (-5:0.1:5)';
```

```
fplot2d(x,  f);
```

The first command produces a column vector of x from 0 to 1 in steps 0.1. The second produces the plot of the function f.

Output:

Figure 12.1: Plot of function f in example 12.2



Similarly, function *fplot3d* can be used to plot a 3D plot of a surface defined by a function.

```
fplot3d (xr, yr, f, [theta, alpha, leg, flag, ebox])
fplot3d (xr, yr, f, <opt_args>)
```

Example 12.3: Say for an instance to define a function

f(x,y) = x4 – y4

in the console we would write

```
deff('z = f(x,y)','z = x^4 - y^4');
```

This will define a Scilab function with the name f.

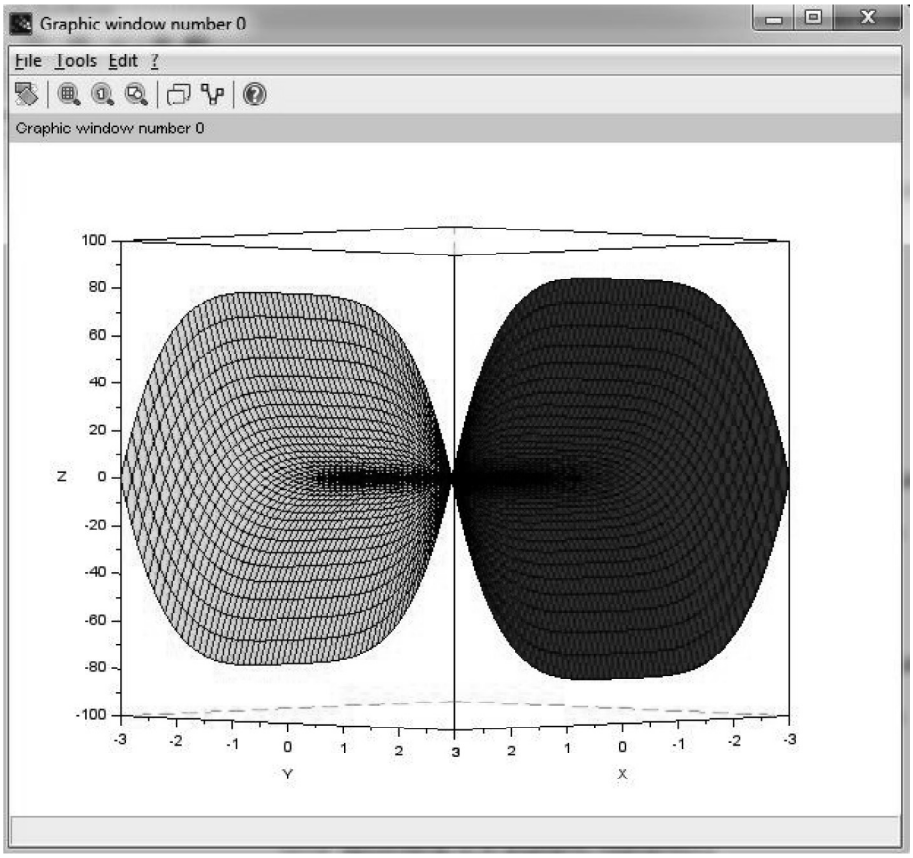
Now to plot the function f in the interval [-3,3] we use

```
x = (-3:0.1:3)';
y=x;
fplot3d(x, y, f);
```

The first command produces a column vector of x from -3 to 3 in steps 0.1. The second command produces a column vector of y same as x i.e. -3 to 3 in steps 0.1. The third produces the plot of the function f.

Output:

Figure 12.2: Plot of function f in example 12.2



12.6 DATA PLOTTING

In Scilab, function *plot2d* can be used to plot a 2D plot of a given data.

```
plot2d ([logflag,][x, ], y[, style[, strf[, leg[, rect[, nax]]]]])
plot2d ([logflag,][x, ], y, <opt_args>)
```

Example 12.4: Say for an instance to plot the data given below,

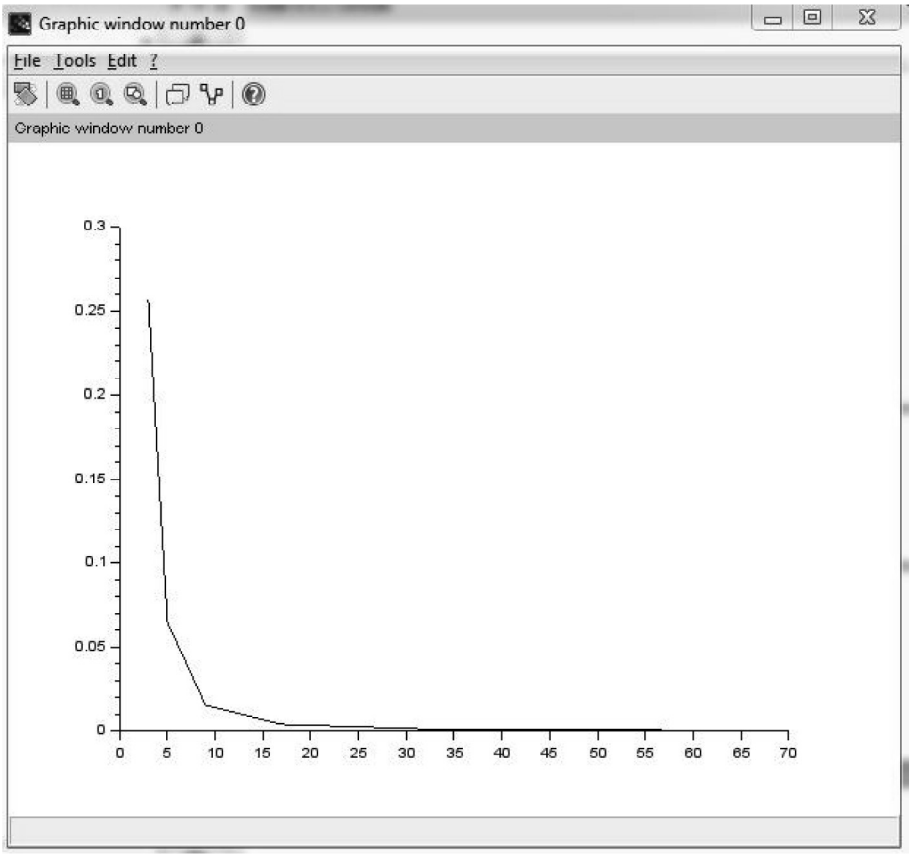
```
n = [ 3 5 9 17 33 65 ]';
s = [ 2.57e-1 6.46e-2 1.51e-2 3.96e-3 9.78e-4 2.45e-4]';
```

```
plot2d (n, s);
```

The command plot2d(n, s) produces the plot of the given data.

Output:

Figure 12.3: Plot of given data in example 12.4



Similarly, function *plot3d* can be used to plot a 3D plot of a given data.

```
plot3d (x, y, z, [theta, alpha, leg, flag, ebox])
plot3d (x, y, z, <opt_args>)
plot3d (xf, yf, zf, [theta, alpha, leg, flag, ebox])
plot3d (xf, yf, zf, <opt_args>)
plot3d (xf, yf, list(zf, colors),[theta, alpha, leg,
flag, ebox])
plot3d (xf, yf, list(zf, colors), <opt_args>)
plot3d (z)
```

Example 12.5: Say for an instance to plot the data given below:

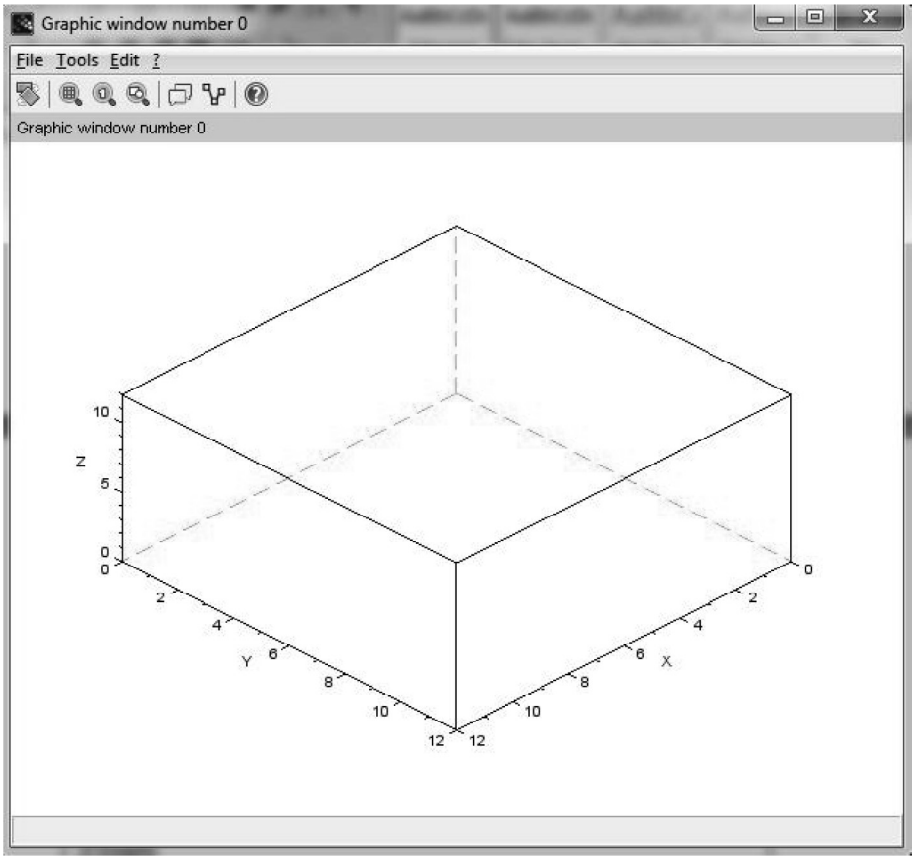
```
x=0:0.1:12;
```

```
y=0:0.1:12;
z=0:0.1:12;
plot3d (x, y, z);
```

The command plot3d(x, y, z) produces the plot of the given data.

Output:

Figure 12.4: Plot of function f in example 12.5





CHECK YOUR PROGRESS

Q 3: Define two functions used to plot a given function in Scilab.

.....

.....

Q 4: Define two functions used to plot given data in Scilab.

.....

.....

12.7 GUI IN SCILAB

A GUI is a human-computer interface that uses graphical objects like windows, menus and icons to interact with users through the use of mouse in keyboard.

The function that creates a GUI object in Scilab is "*uicontrol*".

```
h = uicontrol (PropertyName, PropertyValue,...)
h = uicontrol (parent, PropertyName, Property Value,...)
h = uicontrol (uich)
```

The function *h = uicontrol(PropertyName, PropertyValue,...)* creates an uicontrol and assigns the specified properties and values to it. It assigns the default values to any properties you do not specify. The default uicontrol style is a "Pushbutton". The default parent is the current figure.

The function *h = uicontrol(parent, PropertyName, PropertyValue,...)* creates a uicontrol in the object specified by the handle, parent. If you also specify a different value for the Parent property, the value of the Parent property takes precedence. parent is the handle of a figure.

The function *h = uicontrol(uich)* gives focus to the uicontrol specified by uich.

The style of an uicontrol must be set at creation using the "Style" property and cannot be changed once the uicontrol is created. The available styles are listed below.

- Checkbox: a button with two states
- Edit: an editable string zone.
- Frame: a container for other uicontrols or axes.
- Image: a component where a specified image is displayed.
- Layer: a container for frame style uicontrols.
- Listbox: a control representing a list of items that can be scrolled.
- Popupmenu: a button which makes a menu appear when clicked.
- Pushbutton: a rectangular button generally used to run a callback.
- Radiobutton: a button with two states
- Slider: a scale control, that is a slider used to set values between

in range with the mouse.

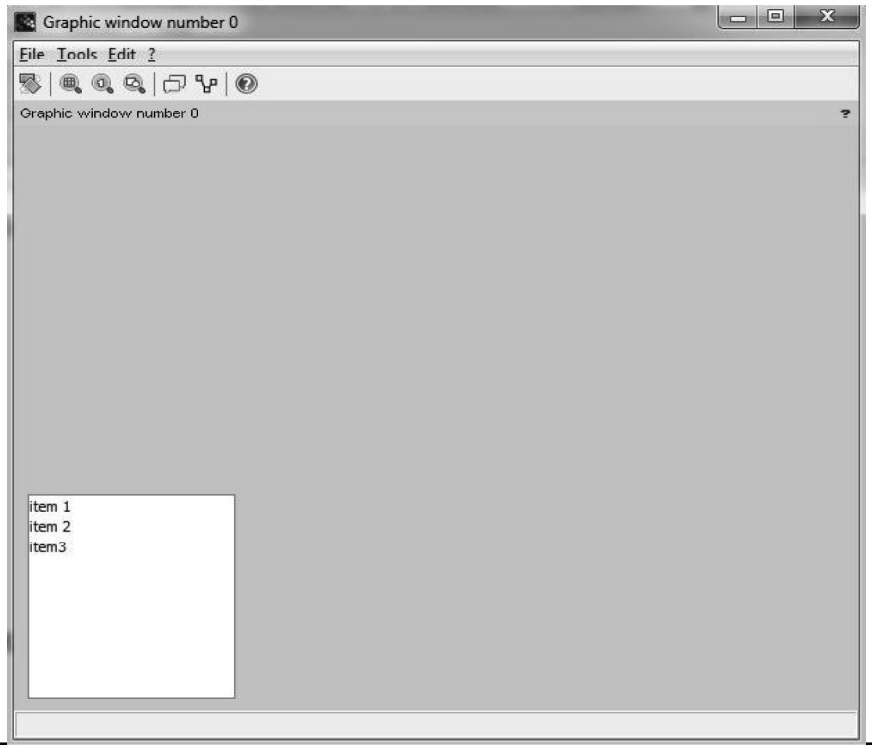
- Spinner: a component which enables the user to select/edit a value between bounds with a fixed step.
- Tab: a container for frame style uicontrols enabling to switch between them by clicking on a tab with a given title and/or icon.
- Table: an editable table.
- Text: a text control

Example 12.6: Using uicontrol() Function

```
f=figure();                    // create a figure
h=uicontrol(f,'style','listbox','position', [10 10
150 160]);      // create a listbox
set(h, 'string', "item 1|item 2|item3");   // fill
the list
set(h, 'value', [1 3]);    // select item 1 and 3 in
the list
close(f);                      // close the figure
```

Output:

Figure 12.5: Graphic window in example 12.6



12.8 PLOTTING 2D GRAPHS

In Scilab, we can create and customize various types of 2D plots. Scilab can create x-y plots, contour plots, pie chart, histograms, bar chart, horizontal bar chart and many other types of plots.

Table 12.3: Plotting 2D Graphs Functions

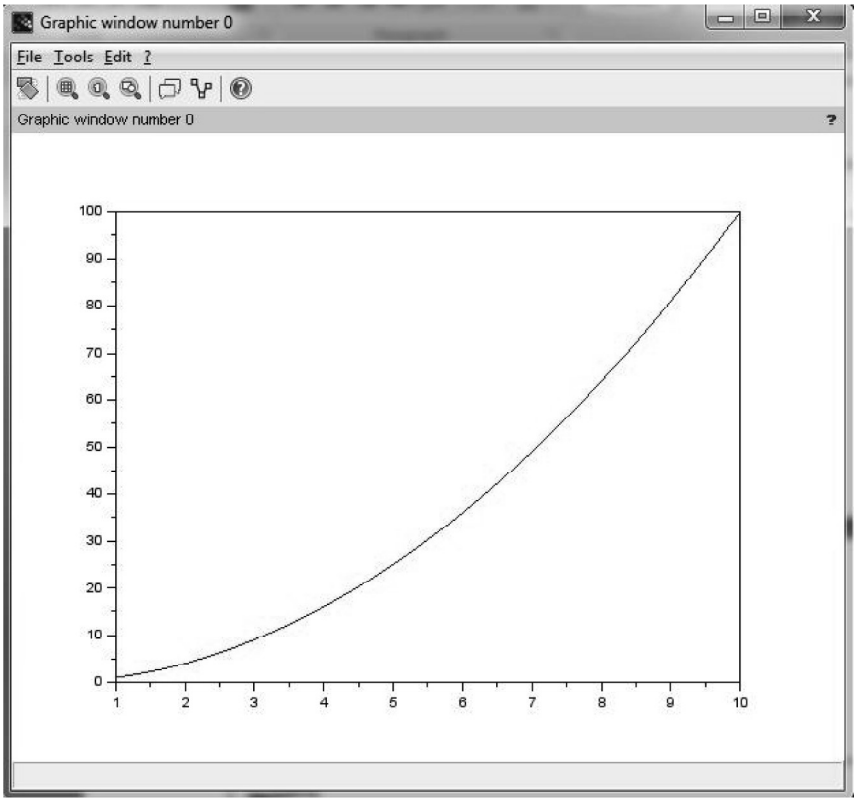
Functions	Descriptions
plot	2D plot
contour	contour plot
pie	pie chart
histplot	histogram
bar	bar chart
barh	horizontal bar chart
polarplot	plot polar coordinates
Matplot	2D plot of matrix using colors
Sgrayplot	smooth 2D plot of surface using colors
grayplot	2D plot of a surface using colors

Example 12.7: Example to create a simple x-y plot.

```
function f=myquadratic(x)
// function squaring the input argument x
f = x^2;
endfunction
xdata = linspace ( 1 , 10 , 50 );
// linespace function to produce 50 values
ydata = myquadratic ( xdata );
// pass myquadratic function and get the function
value
plot ( xdata , ydata );
```

Output:

Figure 12.6: Simple x-y plot from example 12.7



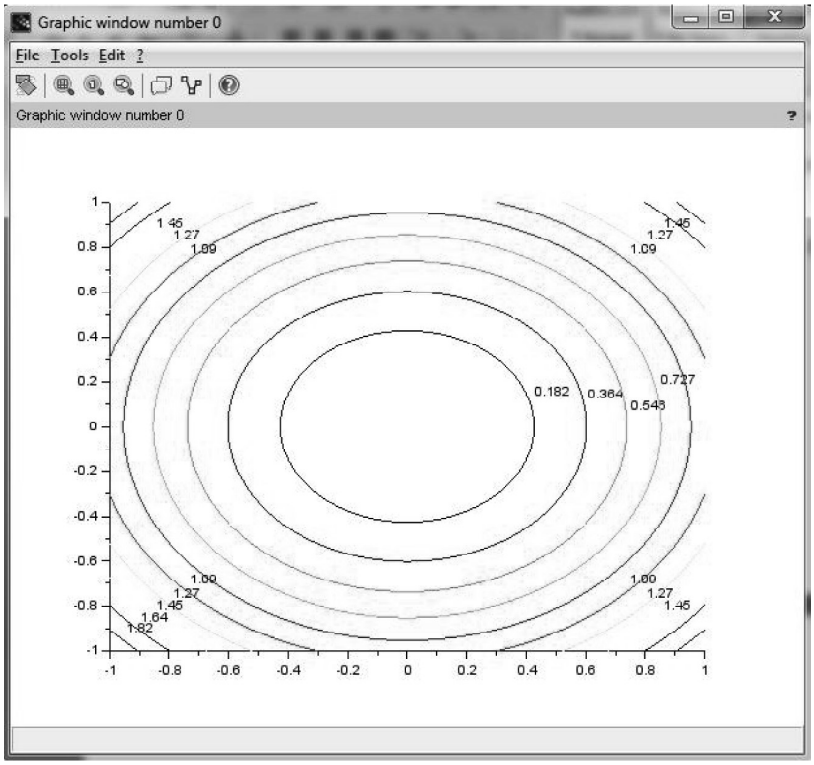
Example 12.8: Example for a contour plot.

```
function f=myquadratic2arg(x1, x2)
// function takes two input arguments x1 and x2
f = x1^2 + x2^2;    // returns f(x1, x2) = x12 +
x22
endfunction

xdata = linspace ( -1 , 1 , 100 );
// linespace function to produce 100 values
ydata = linspace ( -1 , 1 , 100 );
// linespace function to produce 50 values
contour ( xdata , ydata , myquadratic2arg , 10);
```

Output:

Figure 12.7: Contour x-y plot from example 12.8





CHECK YOUR PROGRESS

Q 5: Name two "Style" property of an uicontrol.

.....

.....

Q 6: Name two functions of 2D plotting.

.....

.....



12.9 LET US SUM UP

- Scilab native functions are used to plot graphs.
- Scilab offers many ways to create and customize various types of 2D and 3D plots.

- Function xs2jpg exports the display of a graphic window into a JPG file.
- Function fplot2d can be used to plot a 2D plot of a curve defined by a function.
- Function fplot3d can be used to plot a 3D plot of a surface defined by a function.
- Function plot2d can be used to plot a 2D plot of a given data.
- Function plot3d can be used to plot a 3D plot of a given data.
- Function that creates a GUI object in Scilab is “uicontrol”.
- Scilab can create x-y plots, contour plots, pie chart, histograms, bar chart, horizontal bar chart and many other types of plots.



12.10 FURTHER READINGS

1) Verma, A. K., (2018). *SCILAB: A beginner’s Approach*.

2) Nagar, S., (2018). *Introduction to Scilab: For Engineers and Scientists*.



12.11 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: LineSpec, Matplot and comet3d, surf

Ans to Q No 2: xs2jpg

Ans to Q No 3: fplot2d, fplot3d

Ans to Q No 4: plot2d, plot3d

Ans to Q No 5: Checkbox, Frame

Ans to Q No 6: pie, histplot

12.12MODEL QUESTIONS

Q 1: How to plot graphs in Scilab?

Q 2: How to create 2D and 3D plots in Scilab?

Q 3: Name five functions used to create 2D plots in Scilab.

Q 4: Name five functions used to create 3D plots in Scilab.

Q 5: How to create a .jpg file in Scilab?

Q 6: How to plot a 2D plot of a curve defined by a function in Scilab?

Q 7: How to plot a 3D plot of a surface defined by a function in Scilab?

Q 8: How to plot a 2D plot of a given data in Scilab?

Q 9: How to plot a 3D plot of a given data in Scilab?

Q 10: Which function is used to create GUI in Scilab.

Q 11: Name five “Style” property of an uicontrol.

Q 12: How to create 2D graphs in Scilab?

Q 13: Name five functions used to create 2D graphs in Scilab.

*** ***** ***

UNIT 13 : APPLICATIONS OF SCILAB

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 Applications

- 13.3.1 Numerical Linear Algebra
- 13.3.2 Ordinary Differential Equations

13.4 Numerical Analysis

- 13.4.1 Iterative Methods

13.5 Comparison with C/C++/Matlab

13.6 Let Us Sum Up

13.7 Further Readings

13.8 Answers to Check Your Progress

13.9 Model Questions

13.1 LEARNING OBJECTIVES

After going through this unit, you will be able to–

- use Scilab applications
- solve linear equations
- solve eigen values
- solve differential equations
- perform numerical analysis using iterative methods
- differentiate between Scilab and C/C++/Matlab

13.2 INTRODUCTION

In the previous unit you had been introduced to Graphics in Scilab. We have learned about handling 2D and 3D graphics and generating .jpg files. We have also learned how to plot functions, data, 2D graphs and how to create GUI in Scilab.

In this unit you will be introduced to applications of Scilab. We will learn about numerical linear algebra by solving linear equations, eigen

values, etc., solving ordinary differential equations, numerical analysis using iterative methods and comparison of Scilab with C/C++/Matlab. In the next unit you will be introduced to Python. We will learn how to install Python and also learn how to open a Python console.

13.3 APPLICATIONS

Scilab being a computing environment for scientific and engineering applications provides various functions for computation of any type of scientific and engineering applications. We will be going through some of these applications and will also be using some of these functions to do some of these computations.

13.3.1 Numerical Linear Algebra

Scilab supports computation of numerical linear algebra. Numerical linear algebra consists of problems like Eigenvalue and Singular value, Factorization, Kernel, Linear Equations, Markov Matrices, Matrix Analysis, Matrix Pencil, State-Space Matrices, Subspaces and projection.

We will be looking at how to solve linear equations and eigen values.

Linear equations:

Say there are three equations with three unknowns:

$$2x_1 + 3x_2 - 5x_3 - 13 = 0$$
$$x_1 - 3x_2 + 8x_3 + 13 = 0$$
$$2x_1 - 2x_2 + 4x_3 + 6 = 0$$

This can be written as the matrix equation A.x + c = 0, if

$$A = \begin{bmatrix} 2 & 23 & -5 \\ 1 & -3 & 8 \\ 2 & -2 & 4 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{and} \quad c = \begin{bmatrix} -13 \\ 13 \\ 6 \end{bmatrix}$$

If a unique solution exists it represents the point of intersection of the three planes in the coordinate system (x_1, x_2, x_3).

Now using Scilab function ***linsolve*** we try to find the solution.

```
—>A = [2, 3, -5; 1, -3, 8; 2, -2, 4]
—>C = [-13; 13; 6]
—>linsolve(A, C)
```

Output:

```
ans =
1.
2.
        -1.
```

The function ***linsolve*** gives the unique solution of the 3 equations.

Eigen value:

Say there is a matrix A.

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Now using Scilab function ***spec*** we try to find the eigen values.

```
—>A = [2, 2, 0; 1, 2, 1; 0, 1, 2]
—>spec(A)
```

Output:

```
ans=
      0.5857864
      2.
      3.4142136
```

The function *spec* gives the eigen values of the matrix A.

13.3.2 Ordinary Differential Equations

Scilab uses a function ***ode*** to evaluate an ordinary differential equation or a set of coupled first order differential equations.

```
y=ode(y0,x0,x,f)
```

where,

y_0=initial value of y

x_0=initial value of x

x=value of x at which you want to calculate y.

Example 13.1: Solve the given differential equation.

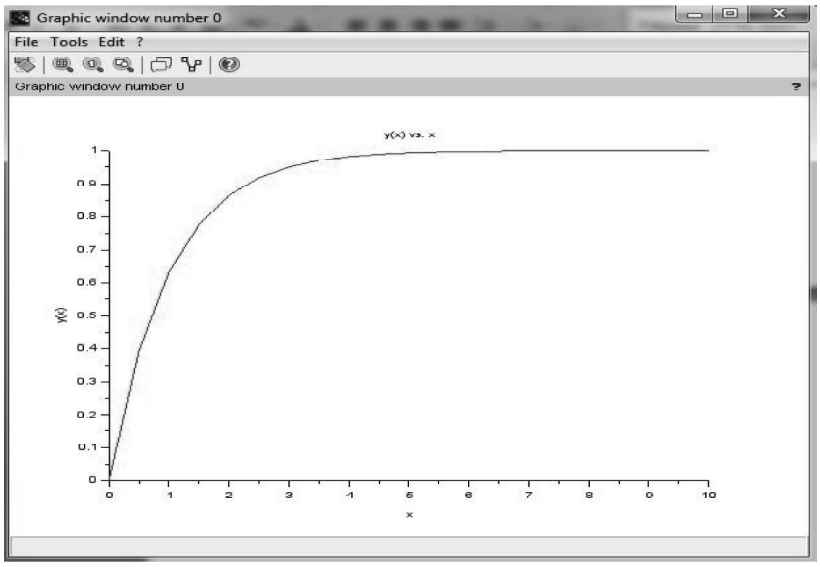
dy/dx = e^{-x} with y=0 for x=0

Solution:

```
funcprot(0)
clf;
function  dx=f(x,  y)
dx=exp(-x);
end  function
y0=0;
x0=0;
x=[0:0.5:10];
sol=ode(y0,x0,x,f);    //  function  ode
plot2d(x,sol,5)       // plot the solution
xlabel('x');
ylabel('y(x)');
xtitle('y(x)  vs.  x');
```

Output:

Figure 13.1: Plot for example 13.1



13.4 NUMERICAL ANALYSIS

Scilab supports numerical analysis. In numerical linear algebra, the Jacobi method and Gauss Seidel method are iterative methods used to solve a linear system of equations.

13.4.1 Iterative Methods

In computational mathematics, an iterative method is a procedure that starts from an initial guess to generate a sequence of solutions that converge to the desired solution.

Let us see the computation of solving a linear system of equations using Jacobi and Gauss Seidel method using Scilab.

- **Jacobi Method:**

```
A = [8 , -3, 2; 4, 11, -1; 6 , 3,12];  /   /
equation matrix
B = [20; 33; 36]       // solution matrix
for i =0:19
       X(i+1 ,1)=i;
end
for i =2:4
       X(1,i)=0;
end
for r =1:19
       for i =1:3
             k =0;
             for j =1:3
                   if(i~=j)
                         k=k-A(i,j)*X(r,j+1);
                   end
             end
             X(r+1,i +1) =(k+B(i ,1) )/A(i,i);
       end
end
```

```
printf ( ' r \t x ( r ) \t \t y ( r ) \t
z ( r ) ' );
printf ( ' \n ————————————————————' )
disp (X)
printf ( ' \n \n After 18 iterations exact
solution is : \n x=%i \t y=%i \t z=%i ' ,X(19
,2) ,X (19 ,3) ,X (19 ,4) )
```

Output:

Figure 13.2: Output of Jacobi method

Scilab 6.0.2 Console

```
--> exec('C:\Users\ANUPAM\jacobi.sci', -1)
r        x ( r )         y ( r )         z ( r )
 ---------------------------------------------------
   0.    0.          0.          0.
   1.    2.5         3.          3.
   2.    2.875       2.3636364   1.
   3.    3.1363636   2.0454545   0.9715909
   4.    3.0241477   1.9478306   0.9204545
   5.    3.0003228   1.9839876   1.0009685
   6.    2.9937532   1.9999707   1.0038417
   7.    2.9990286   2.0026208   1.0031307
   8.    3.0002001   2.0006379   0.9998305
   9.    3.0002816   1.9999118   0.9997405
   10.   3.0000318   1.999874    0.9998813
   11.   2.9999824   1.9999776   1.0000156
   12.   2.9999877   2.0000078   1.0000144
   13.   2.9999993   2.0000058   1.0000042
   14.   3.0000011   2.0000006   0.9999989
   15.   3.0000005   1.9999995   0.9999993
   16.   3.          1.9999997   0.9999999
   17.   2.9999999   2.          1.0000001
   18.   3.          2.          1.
   19.   3.          2.          1.


 After 18 iterations exact solution is :
 x=2     y=2     z=1
-->
```

- **Gauss Seidel Method**

```
A = [8, -3, 2; 4, 11, -1; 6, 3, 12];
//equation matrix
B = [20; 33; 36]     // solution matrix
```

```
for  i  =0:10
       X(i+1  ,1)=i;
end
for  i  =2:4
       X(1,i)=0;
end
for  r  =1:10
       for  i  =1:3
            k1  =0;
            for  j=1:i  -1
                 k1=k1 -A(i,j)*X(r+1,j +1) ;
            end
            k2  =0;
            for  j=i  +1:3
                 k2=k2  -A(i,j)*X(r,j  +1)  ;
            end
            X(r+1,i +1) =( k1+k2+B(i ,1) )/
            A(i,i);
       end
end
printf ( ' r \t x ( r ) \t \t y ( r ) \t z
( r ) ' );
printf ( ' \n ————————————————————' )
disp (X)
printf ( ' \n \nAfter 9 iterations exact
solution is : \n x =%i \ty=%i \t z=%i '
,X(10 ,2) ,X (10 ,3) ,X (10 ,4) )
```

Output:

Figure 13.3: Output of Gauss Seidel method

```
Scilab 6.0.2 Console

--> exec('C:\Users\ANUPAM\gauss_seidel.sci', -1)
 r       x ( r )          y ( r )          z ( r )
 ------------------------------------------------
   0.    0.               0.               0.
   1.    2.5              2.0909091        1.2272727
   2.    2.9772727        2.0289256        1.0041322
   3.    3.009814         1.9968069        0.9958912
   4.    2.9998298        1.9996884        1.000163
   5.    2.9998424        2.0000721        1.0000608
   6.    3.0000119        2.0000012        0.9999938
   7.    3.000002         1.9999987        0.9999993
   8.    2.9999997        2.0000001        1.0000001
   9.    3.               2.               1.
   10.   3.               2.               1.
   11.   2.9999824        1.9999776        1.0000156
   12.   2.9999877        2.0000078        1.0000144
   13.   2.9999993        2.0000058        1.0000042
   14.   3.0000011        2.0000006        0.9999989
   15.   3.0000005        1.9999995        0.9999993
   16.   3.               1.9999997        0.9999999
   17.   2.9999999        2.               1.0000001
   18.   3.               2.               1.
   19.   3.               2.               1.


After 9 iterations exact solution is :
 x=2     y=2       z=1
-->
```



CHECK YOUR PROGRESS

Q 1: Name the functions used to solve linear equations and eigen values.

.....

.....

Q 2: Name the function used to evaluate an ordinary differential equation.

.....

.....

13.5 COMPARISON WITH C/C++/MATLAB

Let us now compare Scilab with C/C++ and Matlab and learn how the different programming languages differ from each other. The comparison between Scilab and C/C++ is given in the table 13.1 below.

Table 13.1: Comparison between Scilab and C/C++

Scilab	C/C++
One-based indexing delimited by parentheses and commas. For example, A(4,5)	Zero-based indexing delimited by square brackets. For example, A[3][4].
a++, a—, a +=b, a -=b, a *=b, a \=b are not supported	a++, a—, a +=b, a -=b, a *=b, a \=b are supported
if, *for* and *while* statements ends with an *end*	End of *if*, *for* and *while* statements are ended by using parenthesis
Complex number notation a + bi is supported	Complex number notation is not supported
String data not supported	String data supported

Now, let us look at some of the differences between Scilab and Matlab. The comparison between Scilab and Matlab is given in the table 13.2 below.

Table 13.2: Comparison between Scilab and Matlab

Scilab	Matlab
One or more functions can be defined in a single file.	Only one function can be defined in a single file.
Functions are not automatically loaded	Functions are automatically loaded
Functions can be defined on-line or inside functions	Functions cannot be defined on-line
exec(“filename”) is used to execute a script file	Just need to type the name of the file
Scilab comments begins with //	Matlab comments begins with %
Predefined variables are prefixed with % in Scilab	Predefined variables are not prefixed
Boolean variables are %T and %F	Boolean variables are 0 and 1



13.6 LET US SUM UP

- Scilab supports computation of numerical linear algebra.
- The function **linsolve** is used to solve linear equations.
- The function **spec** gives the eigen values of the matrix A.
- The function **ode** is used to evaluate an ordinary differential equation or a set of coupled first order differential equations.
- In computational mathematics, an iterative method is a procedure that starts from an initial guess solution to generate a sequence of solutions that converge to the desired solution.



13.7 FURTHER READINGS

1) Verma, A. K., (2018). *SCILAB: A beginner's Approach*.
2) Nagar, S., (2018). *Introduction to Scilab: For Engineers and Scientists*.



13.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: *linsolve* and *spec*

Ans to Q No 2: *ode*



13.9 MODEL QUESTIONS

Q 1: What are some problems in numerical linear algebra?

Q 2: What function is used to solve linear equations in Scilab?

Q 3: What function is used to solve eigen values in Scilab?

Q 4: What function is used to evaluate ordinary differential equation in Scilab?

Q 5: What are iterative methods?

Q 6: Name two iterative methods.

Q 7: Mention five differences of Scilab and C/C++.

Q 8: Mention five differences of Scilab and Matlab.

*** ***** ***

UNIT 14 : INTRODUCTION TO PYTHON

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Python

14.4 Installation of Python

14.4.1 Python installation in Windows

14.4.2 Python installation in Linux

14.5 Opening Python Console

14.5.1 Windows

14.5.2 Linux

14.6 Let Us Sum Up

14.7 Further Reading

14.8 Answers to Check Your Progress

14.9 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define Python
- describe the steps for installing Python
- perform installation of Python
- open Python console in both Windows and Linux

14.2 INTRODUCTION

In the previous unit you had been introduced to applications of Scilab like numerical linear algebra (solving linear equations, eigen values etc.), solving ordinary differential equations and numerical analysis using iterative methods.

In this unit you will be introduced to Python. We will learn how to install Python and also learn how to open a Python console. In the next unit you will be introduced to some basic Python commands. We will also

learn the basics of python programming in the next unit.

14.3 PYTHON

Python is a high level object oriented programming language. It is an interpreted programming language since we can execute instructions directly and freely without needing to previously compile a program before execution. It is also an interactive language since we can use a Python prompt and interact directly with the interpreter to write programs.

14.4 INSTALLATION OF PYTHON

The procedure to install python in Windows and Linux environment are described in detail in the subsections below.

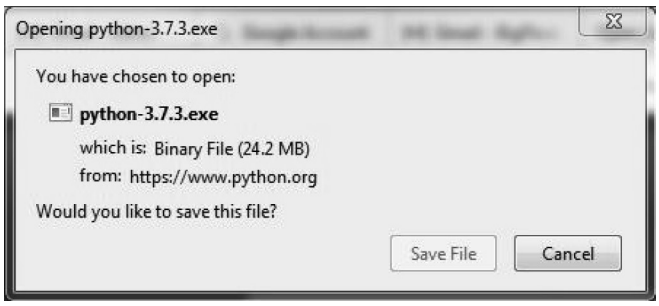
14.4.1 Python installation in Windows

For installation of Python in Windows, we need to follow the instructions given below:

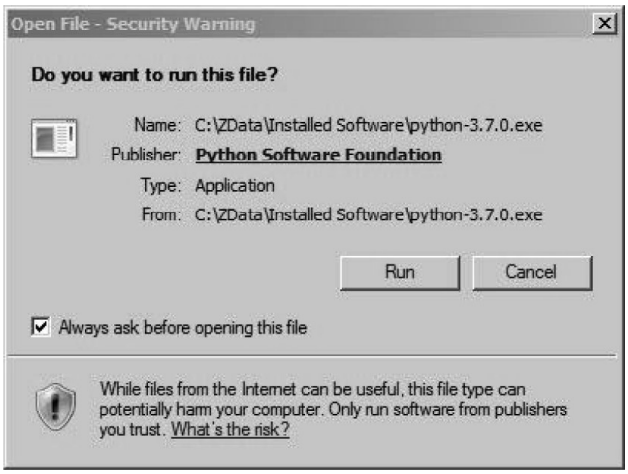
Step 1: We first need to download the latest Python version from the website *https://www.python.org/downloads/.*



Once we visit the website, we then save the downloaded file in a permanent location.



Step 2: We need to double-click the icon labeling the file python-x.x.x.exe. An **Open File - Security Warning** pop-up window will appear like shown below.

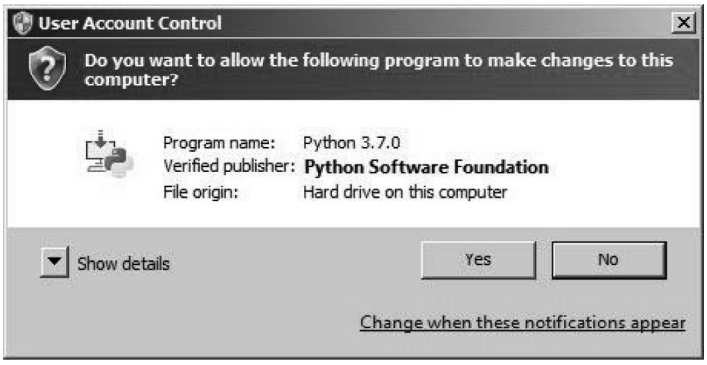


Step 3: Click Run. A Python 3.7.0 (32-bit) Setup pop-up window will appear.



Ensure that the "Install launcher for all users (recommended)" and the "Add Python 3.7 to PATH" checkboxes at the bottom are checked.

Step 4: Highlight the "Install Now" message, and then click it. A User Account Control pop-up window will appear, posing the question "Do you want to allow the following program to make changes to this computer?"



Step 5: We then need to click the **Yes** button. A new Python 3.7.0 (32-bit) Setup pop-up window will appear with a Setup Progress message and a progress bar. Soon, a new Python 3.7.0 (32-bit) Setup pop-up window will appear with a Setup was successfully message.



Step 6: Click the Close button.

Step 7: We first navigate to the directory where Python was installed. We then double click the icon/file python.exe. The following command prompt pop-up window will appear. We can now type our commands in this window.



14.4.2 Python installation in Linux

For installation of Python in Linux, we need to follow the instructions given below:

Step 1: First, we need to check if Python is already installed in the system. To see if Python is already installed, open a command prompt and run.

$ python —version or *$ python3 –version*

Step 2: If Python 2.7 or later is not installed, install Python with your distribution's package manager. The command and package name varies:

a. On Debian derivatives such as Ubuntu, use apt.

$ sudo apt-get install python3

b. On Red Hat and derivatives, use yum.

$ sudo yum install python3

c. On SUSE and derivatives, use zypper.

$ sudo zypper install python3

Step 3: Open a command prompt or shell and run the following command to verify that Python installed correctly.

$ python3 —version

14.5 OPENING PYTHON CONSOLE

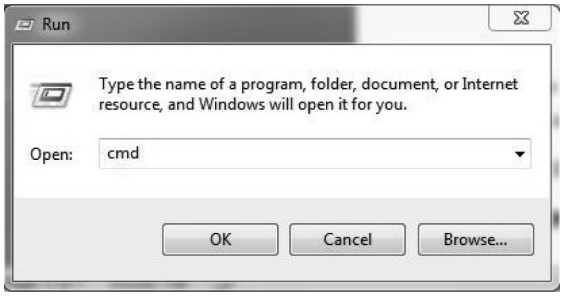
In this section, we will look at how to open the Python console in both Windows and Linux.

14.5.1 Windows

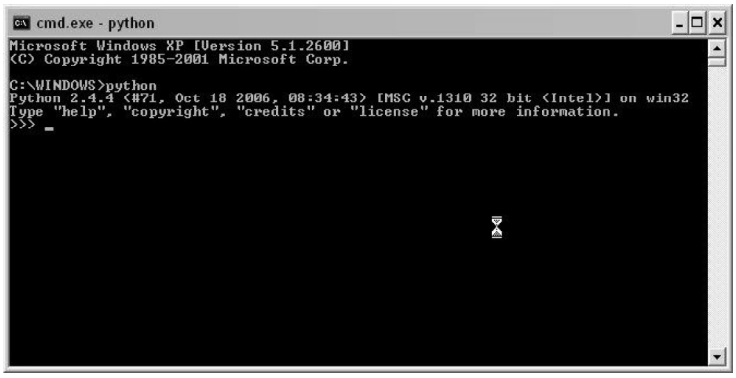
For opening the console in Windows, we need to follow the instructions given below:

Step 1: We first open the Command line. This is done by clicking the **Start Menu** and then typing **run** in the search bar. Typing run will open the Run window where we need to type "**cmd**".

Start menu -> Run and type *cmd*



Step 2: Once we get to the command prompt window, we type *python*.

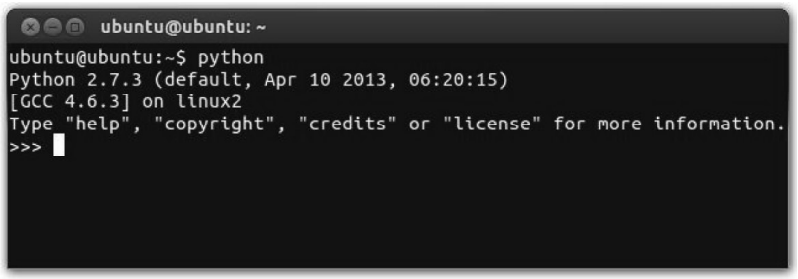


14.5.2 Linux

For opening the console in Linux, we need to follow the instructions given below:

Step 1: Open a command terminal.

Step 2: Type *python*.





CHECK YOUR PROGRESS

Q 1: State whether true or false.

a. Python is a low level language.

b. Python is an interpreted programming language.

c. Python is an interactive language.

Q 2: From where can we download Python for windows installation?

.....

.....

Q 3: What command is used in Linux to know if Python is installed or not?

.....

.....



14.6 LET US SUM UP

- Python is a high level language.
- Python is an object oriented programming language.
- Python is an interpreted programming language.
- Python is an interactive programming language.
- Latest Python version for Windows can be downloaded from *https://www.python.org/downloads/.*
- To see if Python is already installed in Linux environment, we open a command prompt and run *"python –version"* or *"python3 –version".*
- On Debian derivatives such as Ubuntu, we use apt and run command *"sudo apt-get install python3".*
- On Red Hat and derivatives, we use yum and run command "*sudo yum install python3".*
- On SUSE and derivatives, we use zipper and run command "*sudo zypper install python3".*
- For both Windows and Linux environment we use command "*python*" to start Python.



14.7 FURTHER READING

1) Barry, P. (2016). *Head First Python: A Brain-Friendly Guide. " O'Reilly Media*, Inc.".

2) Matthes, E. (2015). *Python crash course: a hands-on, project-based introduction to programming*. No Starch Press.

3) Zelle, J. M. (2004). *Python programming: an introduction to computer science*. Franklin, Beedle & Associates, Inc..



14.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: a. False

b. True

c. True

Ans to Q No 2: *https://www.python.org/downloads/*

Ans to Q No 3: *$ python —version* or *$ python3 —version*



14.9 MODEL QUESTIONS

Q 1: What is Python?

Q 2: Why Python is an interpreted programming language?

Q 3: How is Python is an interactive programming language?

Q 4: From where can we download the latest Python version for Windows Environment?

Q 5: How can we install Python in Linux environment?

Q 6: How can we find if Python is installed in Linux environment?

Q 7: How do we find the Python version installed in Linux environment?

*** ***** ***

UNIT 15 : BASIC PYTHON COMMANDS

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Basic Python Commands

- 15.3.1 Python in Interpreter Mode
- 15.3.2 Loading Python Commands from the Library
- 15.3.3 Defining own Functions in Python

15.4 Python Programming

- 15.4.1 Identifiers
- 15.4.2 Keywords
- 15.4.3 Standard Data Types
- 15.4.4 Operators
- 15.4.5 Decision Making
- 15.4.6 Loops
- 15.4.7 Functions

15.5 Writing Python Programs

15.6 Let Us Sum Up

15.7 Further Reading

15.8 Answers to Check Your Progress

15.9 Model Questions

15.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- run Python commands in interpreter mode
- load Python Commands from the Library
- describe basic Python commands
- describe Python programming basics like data types, operators
- write simple Python programs
- write functions in Python

15.2 INTRODUCTION

In the previous unit you have been introduced to Python installation process in both Windows and Linux environment. We have learned about the different steps involved in installing Python. We have also learned about how to open a Python console in both the environments.

In this unit you will be introduced to basic Python commands and learn python programming. We will discuss all the programming basics of Python- data types, operators, decision making statements and loops in detail in this unit. We will also learn how to use functions using Python programming language. This is the last unit of this course.

15.3 BASIC PYTHON COMMANDS

There are various ways of executing Python commands. They are discussed in the sections below:

15.3.1 Python in Interpreter mode

The easiest way to run Python commands is as an interpreter. In the interpreter mode, we just enter a command and then we get an answer from the Python environment. We know that Python is running in interpreter mode if we get the following message in Windows environment.

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019,
21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

For Linux environment, we know that Python is running in interpreter mode if we get the following message

```
Python 3.7.3 (default, Mar 25 2019, 21:26:53)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

Here, in the above messages the three symbols **>>>** indicates that the prompt is waiting for input.

The >>> prompt can be followed by any text. The Python interpreter can also be used as a calculator.

```
>>>2*1024
2048

>>>1+2+3+4
10

>>>2**10
1024
```

In the above code, multiplication is represented by *, addition by + and exponents by **.

Below is an example to print a table of perfect squares.

```
>>>for n in [1, 2, 3, 4, 5, 6]
…     print(n**2)
…
1
4
9
25
36
```

In the interpreter mode we can quit a Python session by typing “CTRL+D” (i.e. hold down the CTRL key while pressing the D key.) We can also quit by using the menu. If the interpreter gets stuck in an infinite loop, you can also quit the current execution by CTRL+C.

15.3.2 Loading Python commands from the library

Python has a very extensive library of commands, documented in the *Python Library Reference Manual*. These commands are organized into *modules*. One of the available

modules that is especially useful for us is the math module. We can use it as shown below

```
>>> from math import sqrt , exp
>>> exp(-1)
0.3678794411714423 3
>>> sqrt(2)
1.4142135623730951
```

We first import the **sqrt** and **exp** functions from the math module, then use them to compute e^{-1}= 1/e and $\sqrt{2}$.

Once we have loaded a function from a module, it is available for the rest of that session. When we start a new session, we have to reload the function if we need it.

Note that we can import all the functions in the math module by using a *:

```
>>> from math import *
```

15.3.3 Defining own functions in Python

In Python we can define and use our own functions. If we need to do a calculation only once, then we use the interpreter. But when we need to perform a certain type of calculation many times, then we can define a function. For example,

```
>>> def square(x):
...  return x*x
...
```

defines the squaring function square(x) =x^2, which is a popular example used in elementary math courses. We can use the function to calculate the square of any given input:

```
>>> def square(x):
...  return x*x
...
>>> square (3)
9
```

```
>>> square (2.5)
6.25
```

A function name is not completely arbitrary, since we are not allowed to use a reserved word as a function name.



CHECK YOUR PROGRESS

Q 1: What is an interpreter mode in Python?

.....

.....

Q 2: How to import all the functions in the math module?

.....

.....

15.4 PYTHON PROGRAMMING

Python program files are saved with **.py** extensions. Say we have a Python program file with name *my_first.py*, the program can be run using the command “python” in interpreter mode.

```
python my_first.py
```

The program is executed and it continues until the program is finished executing.

15.4.1 Identifiers

An identifier is a name given to a variable, function, class, module or other objects. An identifier starts with an underscore (_) or an alphabet followed by more alphabets, underscores or digits.

15.4.2 Keywords

There are certain keywords in Python which are reserved and cannot be used as a constant or an identifier names. These keywords contain lowercase letters only. The keywords are:

and, exec, not, assert, finally, or, break, for, pass, class, from,

print, continue, global, raise, def, if, return, del, import, try, elif, in, while, else, is, with, except, lambda, yield.

15.4.3 Standard Data Types

There are five standard data types in Python. They are described below:

- **Numbers –** Python supports four different numerical types : *int*, *long*, *float* and *complex* Example, *var1 = 1*.
- **Strings –** Strings are contiguous set of characters represented in quotation marks. Example, str = 'Hello World!'.
- **Lists –** Lists contains items separated by commas and enclosed with square brackets ([ ]). These items can be of different data types. Example, list = ['abcd', 123, 3.14, 'tom'].
- **Tuples –** Tuples are similar to lists containing items separated by commas but enclosed in parenthesis (( )) and whose items cannot be updated. Example, tuple = ('abcd', 123, 3.14, 'tom').
- **Dictionary –** Dictionaries are kind of hash tables where a dictionary key can be almost any Python type and values can be any arbitrary Python object which are enclosed by curly braces ({ }) and values are assigned and accessed using square braces ([ ]). Example, dict = { 'name' : 'tom', 'code' : 123, 'value' : 3.14}.

15.4.4 Operators

Python has many different types of operators. These functions of these operators are described in detail below.

- **Arithmetic Operators**

Operator	Operations
+	Addition
-	Subtraction
*	Multiplication

/	Division
%	Modulus
**	Exponent
//	Floor Division

- **Comparison Operators**

Operators	Operations
==	Equal
!=	Not equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

- **Assignment Operators**

Operators	Operations
=	Assign values right operand to left operand
+=	Add right operand to left operand and assign result to left operand
-=	Subtract right operand from left operand and assign result to left operand
*=	Multiply right operand with left operand and assign result to left operand
/=	Divide left operand with right operand and assign result to left operand
%=	Take modulus using two operands and assign result to left operand
**=	Exponential (power) calculation on operands and assign result to left operand
//=	Floor division on operands and assign result to left operand

- **Bitwise Operators**

Operators	Operations
&	Binary AND
|	Binary OR

^	Binary XOR
~	Binary Ones Complement
<<	Binary left shift
>>	Binary right shift

- **Logical Operators**

Operators	Operations
and	Logical AND
or	Logical OR
Not	Logical NOT

- **Membership Operators**

Operators	Operations
=	Assign values right operan
in	True if it finds a variable in the specified sequence and false otherwise.
not in	True if it does not finds a variable in the specified sequence and false otherwise.

- **Identity Operators –** is, is not.

Operators	Operations
is	True if the variables on either side of the operator point to the same object and false otherwise.
is not	False if the variables on either side of the operator point to the same object and true otherwise.

15.4.5 Decision Making

Decision making are actions taken considering some conditions. Some of the common decision making statements are – *if* statement, *if else* statement and *nested if* statements. We already know how to use these statements in programming languages like C, now let us look how these can be used in Python.

***if* statement**

```
if ( value == 100)
```

```
     print ("value is equal to 100.")
```

***if else* statement**

```
if ( value == 100)
     print ("value is equal to 100.")
else
     print ("value is not equal to 100.")
```

***nested if* statement**

```
if ( value > 1)
     if ( value < 100 )
          print ("value is greater than 1 and
          less 100.")
```

15.4.6 Loops

A loop statement allows us to execute a statement or group of statements. We have already come across different types of loops like *while* loop, *for* loop and *nested* loops in other programming languages like C, C++ etc. Let us look at some examples on how to use these different loops in Python.

***while* loop**

```
     i = 1
     while ( i <= 10)
          print ( i )
          i++
```

***for* loop**

```
     mylist = [ "Hello", "World"]
     for i in mylist:
          print ( i )
```

***nested* loop**

```
     mylist = [ 1, 2, 3 ]
     for i in mylist:
```

```
for j in mylist:
    print ( i*j )
```

Loop control statements are used to change execution of a loop's normal sequence.

Control Statement	Description
break statement	Terminates a loop statement and executes the statement after the loop
continue statement	Skips the remainder of loop statements and retest its condition
pass statement	Used when a loop statement is required but not to be executed

15.4.7 Functions

A function is a block of reusable code that is used to perform a single related action. The syntax for defining a function is given below.

```
def function_name( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

For example, let us look at a python function below:

```
def print_str( str ):
    "This prints a passed string into
    this function"
    print str
    return
```

A function is called by just calling it with necessary parameters. For example,

```
print_str("Calling function print_str!")\
```

15.5 WRITING PYTHON PROGRAMS

In this section, we will write Python programs using the various

programming concepts that we have learned in the earlier sections. Let us now write our first Python program for adding two numbers.

Program 15.1: Sum of two numbers

```
a = int(input("enter first number: "))
b = int(input("enter second number: "))
sum = a + b
print ("sum:", sum)
```

The above Python program 15.1 adds two integer numbers. Variables *a* and *b* are two integers to be added and its values are to be input by the user. The addition of the two numbers is stored in the variable *sum* and printed out using **print** function which is a in-built function.

Figure 15.1: Output of program 15.1

```
C:\Users\ANUPAM\Desktop>python add.py
enter first number: 2
enter second number: 23
sum: 25
```

Program 15.2: Check if a number is even or odd

```
def is_even( num ):
    if (num % 2) == 0:
        print("{0} is Even".format(num))
    else:
        print("{0} is Odd".format(num))
    return
num = int(input("Enter a number: "))
is_even( num );
```

The above Python program checks if a number is an even or an odd number. Variable *num* is an integer variable and its value is inputted by the user. The value of *num* is passed to function *is_even* which using *if else* statement checks if *num* is even or odd and prints out the result.

Figure 15.2: Output of program 15.2

```
C:\Users\ANUPAM\Desktop>python even.py
Enter a number: 5
5 is Odd
```

Program 15.3: Print nth Fibonacci number

```
def  Fibonacci(n):
     if  n<0:
          print("Incorrect  input")
     elif  n==1:
          return  0
     elif  n==2:
          return  1
     else:
          return  Fibonacci(n-1)+Fibonacci(n-2)
num  =  int(input("Enter  value  of  n:  "))
print(Fibonacci(num))
```

The above Python program finds the n-th Fibonacci number. The value of *n* is input by the user. Using the function *Fibonacci(n)* recursively the n-th Fibonacci number is found out.

Figure 15.3: Output of program 15.3

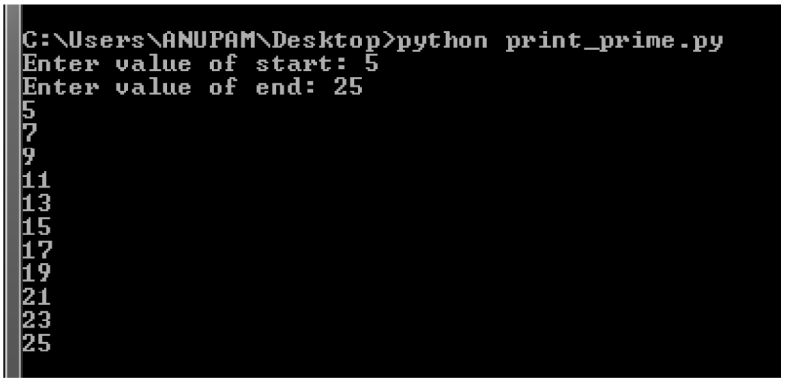


Program 15.4: Prime numbers in an interval

```
start  =  int(input("Enter  value  of  start:  "))
end  =  int(input("Enter  value  of  end:  "))
for  val  in  range(start,  end  +  1):
     if  val  >  1:
          for  n  in  range(2,  val):
               if  (val  %  n)  ==  0:
                    break
               else:
                    print(val)
                    break
```

The above program finds the prime numbers in an interval. The *start* and the *end* values are inputted by the user. Checking if the number is divisible by any number between 2 and the number, we find the prime numbers which are divided only by it.

Figure 15.4: Output of program 15.4



Program 15.5: Reversing a List

```
def  Reverse(mylist):
     return  [ele  for  ele  in  reversed(mylist)]


mylist  =  [10,  11,  12,  13,  14,  15]
print(Reverse(mylist))
```

The above program reverses the contents of the list *mylist*. In this program inbuilt function ***reversed()*** is used to reverse the list.

Figure 15.5: Output of program 15.5



Program 15.6: Check if a substring is present in a given string

```
def  check(string,  sub_str):
     if  (string.find(sub_str)  ==  -1):
          print("NO")
     else:
          print("YES")


string  =  str(input("Enter  value  of  string:  "))
sub_str  =  str(input("Enter  value  of  sub  string:
"))
check(string,  sub_str)
```

The above program checks if substring *sub_str* is present in string *string*. In this program inbuilt function ***find()*** is used to check if the substring is present in the string.

Figure 15.6: Output of program 15.6

```
C:\Users\ANUPAM\Desktop>python if_substring.py
Enter value of start: banana
Enter value of start: na
YES

C:\Users\ANUPAM\Desktop>python if_substring.py
Enter value of start: banana
Enter value of start: ra
NO
```

Program 15.7: Merge two dictionaries

```
def Merge(dict1, dict2):
    return(dict2.update(dict1))

dict1 = {'a': 10, 'b': 8}
dict2 = {'d': 6, 'c': 4}
print(Merge(dict1, dict2))
print(dict2)
```

The above program merges two dictionaries. The dictionaries *dict1* and *dict2* are merged using the inbuilt function ***update()***.

Figure 15.7: Output of program 15.7

```
C:\Users\ANUPAM\Desktop>python merge_dict.py
None
{'d': 6, 'c': 4, 'a': 10, 'b': 8}
```



CHECK YOUR PROGRESS

Q 3: What is the extension of a python file?

.....

.....

Q 4: Name three keywords in Python.

.....

.....



15.6 LET US SUM UP

- Python has an interpreter mode.
- In the interpreter, we just enter a Python command and it gives the answer.

- Python interpreter can be used as a calculator.
- Interpreter mode can be quit by CTRL-D.
- If the interpreter gets stuck in an infinite loop, you can quit the current execution by CTRL-C.
- In Python commands are organized into modules.
- Python functions can be loaded from modules.
- In Python we can define and use our own functions.
- When we need to perform a certain type of calculation many times, then we can define a function.
- A function name is not completely arbitrary.
- Python program files are saved with .py extensions.
- An identifier is a name given to a variable, function, class, module or other objects.
- Python have reserved keywords which cannot be used as a constant or an identifier names.
- Python has five standard data types - numbers, strings, lists, tuples and dictionaries.
- Python has seven types of operators - arithmetic, comparison, assignment, bitwise, logical, membership and identity operators.
- A loop statement allows us to execute a statement or group of statements.
- Python has three types of loops - while, for and nested loops.
- Python has three control statements - break, continue and pass statement.
- A function is a block of reusable code that is used to perform a single related action.
- A function is called by just calling it with necessary parameters.



15.7 FURTHER READING

1) Barry, P. (2016). Head First Python: A Brain-Friendly Guide. “ O’Reilly Media, Inc.”.

2) Matthes, E. (2015). Python crash course: a hands-on, project-based introduction to programming. No Starch Press.

3) Zelle, J. M. (2004). Python programming: an introduction to computer science. Franklin, Beedle & Associates, Inc..



15.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Interpreter mode in Python is where we just enter a command and Python gives the answer.

Ans to Q No 2: By running the command "*from math import **" in interpreter mode.

Ans to Q No 3: The extension is *.py.*

Ans to Q No 4: Three keywords in Python are: and, exec, not.



15.9 MODEL QUESTIONS

Q 1: What is a Python interpreter mode?

Q 2: How can we quit a Python interpreter mode?

Q 3: How to quit a Python interpreter mode when it gets stuck in infinite loop?

Q 4: From where can we load Python commands?

Q 5: Give an example of how to load a Python command.

Q 6: How can we define our own function in Python?

Q 7: What is an identifier?

Q 8: What are standard data types in Python?

Q 9: What are types of operators in Python?

Q 10: What are the different loops in Python?

Q 11: What are the control statements in Python?

Q 12: What is a function?

Q 13: How do we call a Python function?

Q 14: How to write a function in Python?

Q 15: Write a Python program to find greatest of two numbers.

Q 16: Write a Python program to print Fibonacci numbers.

Q 17: Write a Python program to find smallest number in a list.

Q 18: Write a Python program to count positive and negative numbers in a list.

Q 19: Write a Python program to split and join a string.

Q 20: Write a Python program to remove a key from a dictionary.

*** ***** ***



REFERENCES (For All Units of the Course)

Books:

1) Mittelbach, F., Goossens M.,The LaTeX Companion, 2nd ed. Addison-Wesley(2004), ISBN 0-201-36299-6.
2) Lamport L.: LATEX, A Document Preparation System, User's Guide and Reference Manual, 2nd ed., Addison-Wesley (1994), ISBN 0-201-52983-1.
3) Knuth, D. E., & Bibby, D. (1984). The texbook (Vol. 3). Reading: Addison-Wesley.
4) Heck, A. (2005). Learning LATEX by Doing. AMSTEL Institute.
5) Goossens, M., Rahtz, S. P., Rahtz, S., &Mittelbach, F. (1997). The LATEX graphics companion: illustrating documents with TEX and PostScript (Vol. 1). Addison-Wesley Professional.
6) Kottwitz, S. (2011). LaTeX beginner's guide. Packt Publishing Ltd.
7) Mittelbach, F., Goossens, M., Braams, J., Carlisle, D., & Rowley, C. (2004). The LATEX companion. Addison-Wesley Professional.
8) Kopka, H., & Daly, P. W. (1999). A Guide to La T E X: document preparation for beginners and advanced users.
9) Jackson, A. L., &Geckeis, C. K. (2003). How to prepare your curriculum vitae. McGraw-Hill.
10) Grätzer, G. (2013). Math into LATEX. Springer Science & Business Media.
11) Verma, A. K., (2018). SCILAB: A beginner's Approach.
12) Nagar, S., (2018). Introduction to Scilab: For Engineers and Scientists.
13) Barry, P. (2016). Head First Python: A Brain-Friendly Guide. " O'Reilly Media, Inc.".
14) Matthes, E. (2015). Python crash course: a hands-on, project-based introduction to programming. No Starch Press.
15) Zelle, J. M. (2004). Python programming: an introduction to computer science. Franklin, Beedle & Associates, Inc..