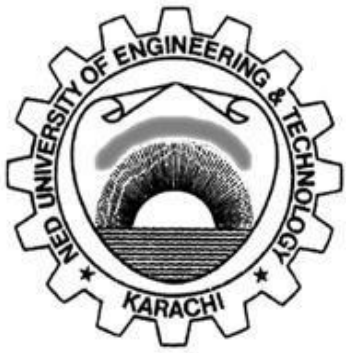


Department of Telecommunications Engineering
NED University of Engineering and Technology

LABORATORY WORKBOOK
For the Course

IT Fundamentals and Applications

(EF-101)

Instructor Name: _____

Student Name: _____

Roll Number: _____ **Batch:** _____

Semester: _____ **Year:** _____

Department: _____

Department of Telecommunications Engineering
NED University of Engineering & Technology

LABORATORY WORKBOOK

For The Course

IT Fundamentals and Applications

(EF-101)

Prepared By

Engr. Ghulam Fiza
(Lecturer)

Reviewed By

Dr. Sundus Ali
(Assistant Professor)

Approved By:

Board of Studies of Department of Telecommunications Engineering

CONTENTS

LAB SESSION 01

Objective:

To provide students with hands-on experience regarding understanding of the basic components of a computer system and its peripherals.

Components:

- Computers (laptops or desktops)
- Projector and screen
- Various computer peripherals (keyboard, mouse, printer, scanner, etc.)
- CPU chip (for demonstration purposes)
- Motherboard (for demonstration purposes)
- Power supply unit (for demonstration purposes)
- RAM modules (for demonstration purposes)
- Hard drive (for demonstration purposes)
- Various cables (power cables, USB cables, etc.)

Theory:

Computer

A computer is an electronic machine that takes input from the user (data), processes the given input, and generates output in the form of useful information. The most elementary computing concepts as shown in Figure 1.1 include receiving input known as data from the user, manipulating the input according to the given set of instructions and delivering the output known as information to the user.

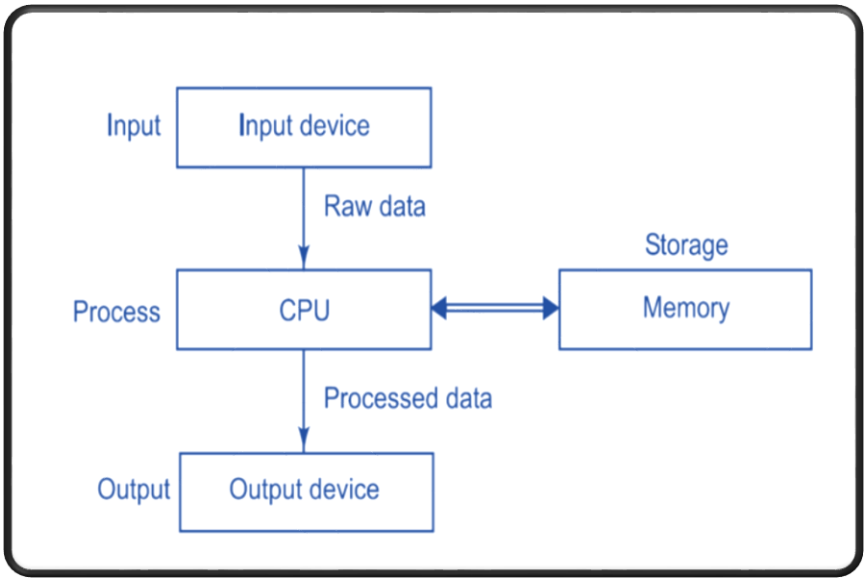


Figure 1.1 Elementary Computing Concept

Peripheral Devices

Peripheral devices are hardware components that connect to a computer and provide additional functionality and features. These devices are called "peripherals" because they are auxiliary to the central processing unit (CPU) and main components of the computer. Peripheral devices serve various purposes, such as input, output, or storage.

Central Processing Unit

The CPU (Central Processing Unit) is the core component of a computer responsible for executing instructions and performing data processing. The CPU consists of several functional units that work together to carry out these tasks. These functional units include:

- Arithmetic Unit
- Logic Unit
- Control Unit
- Main Memory Unit
- Registers

Procedure:

Remind yourself of the importance of handling computer components with care and ensuring that the computer is powered off and unplugged before any internal exploration.

a. Hands-on Peripheral Interaction

1. Ensure that all the necessary peripherals, including the keyboard and mouse, are connected to the computer.
2. Start by practicing typing on the keyboard.
3. Type a few sentences, words, or your name into a text document to get a feel for the keyboard.
4. Move the mouse on the mousepad to see how it moves the cursor on the screen.
5. Click the left and right mouse buttons to understand their functions.
6. Try dragging and releasing.
7. Open a sample document or application.
8. Practice using the mouse to: Click on icons, buttons, or links.
9. Right-click to access context menus.
10. Scroll using the mouse wheel.
11. If other peripherals are available (e.g., printer or scanner) explore and interact with these peripherals under supervision.

b. Opening a Computer Case

1. Before proceeding, ensure that the computer is powered off and unplugged from the electrical outlet.
2. Locate the computer case, which is the outer housing of the desktop computer.
3. Ensure you have a screwdriver ready for opening the case.
4. Observe the computer case for screws or fasteners that secure the side panel.
5. Commonly, there are two or more screws on the rear side of the case.
6. Use the screwdriver to carefully remove the screws from the side panel of the computer case.
7. Place the screws in a safe location, so they aren't lost.
8. Depending on the computer case design, you may need to slide the side panel back or unscrew a latch to open it.
9. Gently slide or open the side panel to reveal the internal components.

c. Interactive CPU Demonstration

1. Once the case is open, take a moment to visually inspect the interior of the computer.
2. Look for the major components, including the motherboard, CPU (Central Processing Unit), and RAM (Random Access Memory).
3. Identify the largest circuit board inside the computer. This is the motherboard.
4. Observe the various connectors, slots, and ports on the motherboard.
5. Locate the CPU, which is a small chip mounted on the motherboard.
6. Take note of its position and any markings or labels on the CPU.
7. Identify the RAM modules, which are usually small, rectangular chips mounted on the motherboard.
8. Note the number of RAM modules and their locations.

d. Closing a Computer Case

1. After inspecting the internal components, carefully close the computer case.
2. Make sure it is securely fastened and the screws are reattached.

Observations:

Results:

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 02

Objective:

To familiarize with DOS environment and its important commands.

Theory:

Understanding DOS

DOS, the acronym for Disk Operating System, is an operating system with a command-line interface used on personal computers. It provides a set of commands that enables the users to access or manipulate information on their disks, as well as simply interact with their computer.

Dos Commands

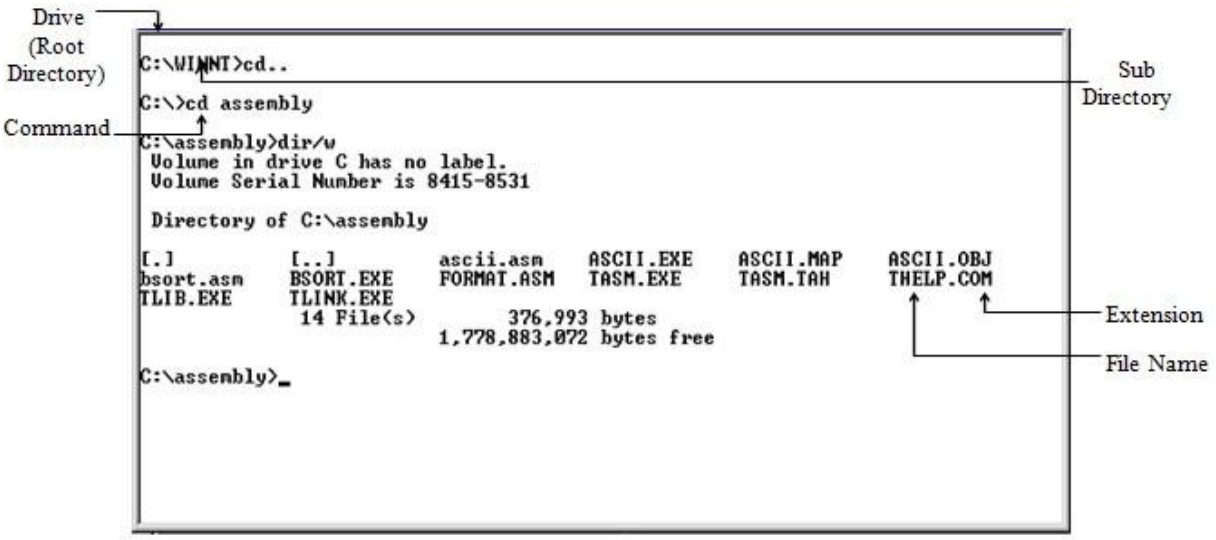


Figure 2.1: DOS environment

DOS commands are of two types namely internal commands and external commands.

- Internal commands are those which are built into command.com.
- External commands are those that must be located from a file loaded by command.com before it can be executed.

A brief description of important DOS commands is given below:

Changing drive

Type drive letter of the drive to which you want to switch to, on the command prompt followed by ':' Example: c:>d: This will change the current drive from C to D. Result: d:

Wild Cards

DOS recognizes two wild cards:

- The asterisk (*) represents one or more characters that a group of files has in common.
- The question mark (?) represents a single character that a group of files has in common.

Working with Directories

- **dir** to view the contents of a directory
  Prompt on screen: drive:>
  Syntax: dir
  Example: c:>dir This will list all the files and subdirectories on drive C, which is being prompted on screen. If another drive needs to be accessed, first change the drive, as previously explained and then enter the **dir** command.

  Use wild cards to display selected lists. For example:
  - dir *.* Displays all files and subdirectories on the drive
  - dir ?????.com Displays all files having names up to five characters and extension **'com'**

  dircommand can be modified using these wild cards and other switches as follows so that only one screen of selected information is displayed at a time.
  - dir/p Lists the directory contents page wise.
  - dir/w Lists the directory contents in a wider format, that is, only file names and extensions.
  - dir/w/p Lists the directory contents in a wider format, page wise.

- **cd** to change from one directory to another
  Prompt on screen: drive:>
  Syntax: cd drive:\path
  Example: c:>cd dos This will change the current directory from the root directory to **'dos'** directory on the prompted drive C.
  Result: c:\dos>

- **md** to create a new directory
  Prompt on screen: drive:>
  Syntax: md drive:\path\dirname
  Example: c:>md neduet This will create a new directory, named **'neduet'** on the drive being prompted, in this case drive C.

- **cd..** to switch back one level up in the directory structure
  Prompt on screen: drive:>
  Syntax: cd..
  Example: c:\dos\subdos>cd.. This will switch back to the directory, **'dos'** from the current directory, **'subdos'** of the prompted drive C.
  Result: c:\dos>

- **cd** to switch back to the root drive directory
  Prompt on screen: drive:>

  Syntax: cd\
  Example: c:\dos\subdos>cd\ This will switch back to the root drive,
  C Result: c:\>

- **rd** to remove a directory
  Prompt on screen: drive:>
  Syntax: rd drive:\path\dirname
  Example: c:\>rd neduet
  This will delete the directory, named **'neduet'** from the prompted drive C.

  Note that rd doesn't delete folders having subdirectories. For that purpose rd/s is to be used

- **tree** to view directory listing in a hierarchical structure. Prompt on screen: drive:>
  Syntax: tree drive:\path
  Example: tree d: This will display the hierarchical directory structure of drive D.

  Use /f to display the names of the folders as well as the files in each folder, in a hierarchical manner.

Working with Files

- **copy** to copy a file from one directory or drive to another Prompt on screen: drive:>
  Syntax: copy drive:\source path\filename drive:\destination path\ filename
  Example: copy c:\neduet\cfile.exe c:\windows
  This will copy the file, named **'cfile.exe'** from **'neduet'** directory to **'windows'** directory with the same name for the new file.

- **ren** to rename a file Prompt on screen: drive:>
  Syntax: ren drive:\path\old filename drive:\path\new filename
  Example: c:\>ren cfile.exe edit.exe
  This will rename the file **'cfile.exe'** to **'edit.exe'** in the prompted drive C.

- **del** to delete a file
  Prompt on screen: drive:>
  Syntax: del drive:\path\filename
  Example: c:\>del cfile.exe This will delete the file, **'cfile.exe'** present in the prompted drive C.

  Wild cards can also be used with any of the above commands to work with group of files.

- **type** to view a file on the screen. Only text based files can be viewed, as DOS supports only such files.

Prompt on screen: drive:>

Syntax: type drive:\path\filename

Example: c:\>type cfile.txt This will list **'cfile.txt'**.

Files longer than one screen scroll off to the top. To avoid this, use |more suffix following the command**.**

Example: c:\>type cfile.txt |more

Disk Management

- **format** to format a disk Prompt on screen: drive:> Syntax: format drive:

Example: c:\>format Z: This will format Flash memory stick in drive Z.

- **chkdsk** to get a report on statistics of the disk Prompt on screen: drive:>

Syntax: chkdsk drive:

Example: c:\>chkdsk c: This will generate a report on statistics of drive C.

- **vol** to view a disk's volume label and it's serial number Prompt on screen: drive:>

Syntax: vol drive:

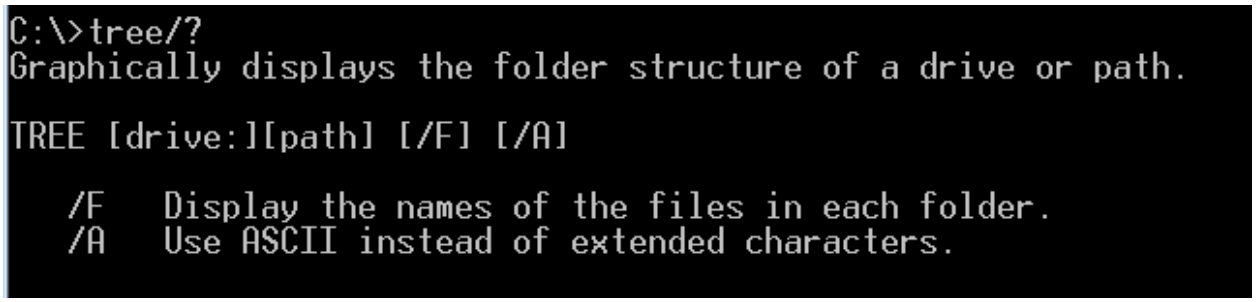
Example: c:\>vol d: This will display volume label of drive d.

- **ver** to find current version of DOS installed on your system Prompt on screen: drive:>

Syntax: ver

Example: c:\>ver This will display the current version of DOS on your system.

Tip:
You can take help for any DOS command by typing command name followed by /?
e.g.



LABORATORY TASKS:

1. Write command to display all files having extension ‘docx’.

Ans: _____

2. Write command to display all files of all types starting with letter ‘s’.

Ans: _____

3. Write a command to copy the file assignment of type docx from the directory ‘ITC’ of drive D to another directory “FCE” in drive F.

Ans: _____

4. Write a command to delete the directory as well as the files of the directory ‘world’ on drive E.

Ans: _____

5. Write command to copy all the files beginning with ‘m’ and whose file names has a ‘txt’ extension from drive A to the ‘\document’ directory on drive C.

Ans: _____

6. Write set of commands to create a directory ‘practical’ on drive F, and then move into it. Now list all files present in it, then go back to root drive F.

Ans: _____

_____

_____

_____

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 03

Objective:

To explore and utilize key features of Microsoft Office tools

Theory:

Microsoft Word:

Microsoft Word is designed to be user friendly. So, the MS Word often has more than one way to achieve much the same thing. To start a document, type the first few paragraphs of text. Don't worry yet about formatting or fonts or making text bold. Obviously the text we have on the screen so far is not what we have in mind for a nicely formatted document. We need to make the Title and the Headings large and bold. And, we need space after the paragraphs. We need colors, pictures, diagrams etc. in a nicely formatted document.

To achieve that, you can use formatting options discussed below.

Case Changing

To change case, select the text whose case to be changed; Press Shift+F3 to change the case. Each time F3 is pressed the user will toggle through three options: UPPERCASE, lowercase, and Title Case.

Indentation

To indent both the left and right sides of the paragraph, position the insertion point in the paragraph to be indented, or select multiple paragraphs to indent; Choose Page Layout, type or select a value in the Left and then the Right indentation text boxes. You can also increase or decrease the indentation by clicking on the up and down arrows beside the text boxes.

Text Alignment

Word automatically aligns text on the left margin (horizontal alignment) and to the top margin (vertical alignment). The user can choose to change the alignment to center, right, full justified, or back to left.

To change horizontal alignment, select the paragraph(s) to be changed; Press Ctrl+L (Left), Ctrl+E (Center), Ctrl+R (Right), or Ctrl+J (Justify) to change the alignment of the paragraph accordingly (or use the Align Left, Center, Align Right, or Justify buttons on the Standard toolbar).

To change vertical alignment, move the insertion point inside the section where the text is to be aligned; Choose Page Layout, Align. Select Center, Justify, or Top to change alignment.

Borders, Lines, and Shading

For a finished look, the user can add borders and shading to the documents. Just select the paragraph(s) to apply a border or shading to, then choose Insert, Table and click on Insert Table. In the title bar, there appears a Table Tools box. Click on this box. Click on Border and then on Border and Shading. Then click the desired line style; Click the Line Weight button, then click the line weight to be used; Click the drop-down arrow on the Borders button, then click the type of border to be applied; Click the drop-down arrow on the Shading Color button to display the palette of gray percent and colors, then click the percent of gray or color button.

Columns

To create columns of equal width, choose Page Layout; Select the text (or to format the entire document with columns, select the document); On the Standard toolbar, click the Columns button; Drag the pointer to select the number of columns needed. To remove columns, select the text for the columns to be removed; Click the Columns button on the Standard toolbar and select one column.

Drop Cap

A drop cap is large capital letters of the first word that is set into a paragraph to add visual interest. Just select the first letter, word, or section of the paragraph to be formatted with a drop cap; Choose Insert, Drop Cap; Select Dropped or In Margin; In the Drop Cap Options section, select the Font from the drop-down list, and increase or decrease the Lines to Drop if the default of three lines is not to be used; If the distance of the text from the drop cap is to be changed, use the increment buttons on the Distance from Text option, then choose OK. To remove drop caps, click the drop caps text, choose Drop Cap, click the None option in the Position section of the dialog box, then choose OK.

Customizing Paragraph Spacing

The user can customize the paragraph spacing in Word for the spacing between paragraphs and the spacing between the lines in specified paragraphs. To do this, place the insertion point in the paragraph to be modified, or highlight all of the contiguous paragraphs to be changed and right click on it; Choose

Paragraph to display the dialog box, then click the Indents and Spacing tab if it is not active; In the Spacing section, select Before and After and change the value(s) in the text box to increase or decrease by points the number of lines before or after a paragraph (6 points = 1 line); To change the line spacing within paragraphs, select the drop-down arrow for the Line Spacing list box, then select one of these options: 1.5 lines, Double, At least, Exactly, or Multiple; If one of the last three options is chosen, enter a number in the At text box. When finished, choose OK.

Inserting Special Characters

To insert special characters, place the insertion point at the point in the paragraph where a symbol or special character is to be placed; Choose Insert, Symbol to display the Symbol dialog box. Click the Symbols tab to select a symbol, or click the Special Characters tab to select a special character; On the Symbols tab, select the font set that contains the symbol to be inserted in the document in the Font drop-down list box. To view a symbol in the displayed table, click the symbol. The symbol is then displayed in an enlarged and highlighted view; To insert a symbol or special character, click the item to be inserted, then click the Insert button. Click the Close button to close the dialog box and return to the document.

Creating Custom Colors/Gradients/Textures

To create custom colors, right click on the graphic element to be colored. Along with the shortcut menu appears the Drawing toolbar. Then click the drop-down arrow of the Fill Color or Line Color button and select More Fill Colors to open the Colors dialog box; Click the Custom tab at the top of the Colors dialog box. Set desired color. For gradient settings, in the drop-down arrow beside the Fill Color button, select Gradient at the bottom of the color palette instead of More Fill Colors. Click the Gradient tab and make desired settings. Texture tab can also be selected to apply designs to the objects.

Inserting Clip Art/AutoShapes/Pictures/WordArt

To insert a Clip Art/ AutoShape, Picture or WordArt, position the insertion point in the document where the image is to be appeared. Choose Insert and then the desired option. Related dialog box appears, then make required selections. The inserted object can be customized in a number of ways. The user can change colors of the image, edit, reposition and resize them.

There are two ways to rotate or flip the inserted object. One is to right click on the object and then choose Format Picture/Shape. Then click on the 3-D Rotation button. From here, the desired rotation in the x, y and z axes may be chosen. The other gives access to that same tool and to the commands Rotate Left, Rotate Right, Flip Horizontal, and Flip Vertical. To do this, click the object to be adjusted choose Page Layout, click on Rotate and then choose the desired option.

These commands automatically execute. If Free Rotate is chosen (if we bring the cursor on the small green circle that appears above any object when we select it, this gives the free rotate option) continue with the following steps; Move the pointer over the rotate handle and, when it assumes the shape of the rotate tool icon, hold down the right mouse button; The pointer changes to four circling arrows. Drag right or left. While dragging, a dotted outline indicates where the object would be if the mouse button is released; until the object is deselected it can be continuously rotated, even if the mouse button has been released. Place the pointer over a circle handle, hold down the mouse button and continue to rotate. Release the mouse button when the object is rotated to the desired position, and click outside the object to deselect it.

Creating Headers and Footers

Headers and footers contain information repeated at the top or bottom of the pages in a document. To set Header and Footer, with the document open, choose Insert, then click on Header or Footer. Choose the desired format from the drop down list. The Header pane is displayed in the document; Type and format the information for the header; To include the Page Number, Number of Pages, or current Date or Time, click the corresponding button on the Header and Footer toolbar; To create a footer, click the Go to Footer Button on the toolbar and type and format the footer just as was done for the header. Click Close Header & Footer button to return to the document.

Document Map

The Document Map is a very functional way to move quickly through long or online documents. To use Document Map, choose View, Navigation Pane. Use the mouse to click the heading or text in the Document Map pane to move to that section of the document; Click the arrow on the left of any heading to expand or collapse the headings; To adjust the size of the Navigation pane, move the mouse pointer onto the right edge of the pane so the pointer changes to a resizing pointer, then click and drag the edge to the left or right. To use the keyboard, press F6 to move to the Navigation pane. Arrow keys and the

Enter key will move to the desired location in the document.

Drawing Tables

Drawing a table allows the user to place the rows and columns where he wants them. To create a table, click the Tables and Borders button to bring up the Tables and Borders toolbar and change the mouse pointer to a pencil; Drag the mouse pointer from one corner of the new table to the opposite corner to create the rectangle outline for the table; If a line is to be removed, click the Eraser button on the Tables and Borders toolbar and drag across the line.

If the user needs the data from a table, he may (rarely) want to convert the table to text with some sort of separator between the data for the former columns. To do this, select the entire table by positioning the mouse directly at the left top of the table until the pointer changes to a four directional arrow and then click it. A Table toolbar appears at the top of the document window. In the Layout tab, choose Convert to Text. Choose to separate the text with Paragraph Marks, Tabs, Commas, or type a new character in the other text box; then choose OK. Similarly the user may want to convert the lines of text into a table. To do this, select the rows of tabbed text to be converted; Choose Insert, click on Table and then choose Insert Table.

The cells of the table can be edited and formatted, like moving and copying cells, splitting and merging cells, adjusting row and column spacing/width, inserting and deleting rows and columns, by selecting appropriate options from the menu that appears on right clicking the table or the selected cells.

Paragraphs: Formatting Line and Page Breaks

The user would not want to have one line of a paragraph appear on the bottom or top of a page alone. This is Widow and Orphan control. Select the paragraph that you want to prevent from breaking onto two pages. On the Page Layout tab, click the Paragraph Dialog Box Launcher, and then click the Line and Page Breaks tab. Select the Keep lines together check box. Keep with Next will prevent a page break between the selected paragraph and the following paragraph. Page Break Before will insert a manual page break before the selected paragraph.

To insert page breaks, position the insertion point anywhere within the paragraph to be formatted, and choose Insert, and then click on Page Break.

Sections Breaks

Section breaks are important when the user need to apply different formatting choices to different parts of the document. To insert Section Breaks, position the insertion point where the section break is required, and then On the Page Layout tab, in the Page Setup group, click Breaks. Click the type of section break that you want to use. Different options are: Next Page- to start section at the top of the next page in the document; Continuous-to start section at the insertion point (for varying columns on one page); Even Page-to start section on next even- numbered page in the document (most often a left-facing page); Odd Page-to start section on next odd-numbered page in the document (most often a right-facing page).

Use the Continuous section break to balance columns on a page. Insert the break at the end of a document that is divided into columns where the last column does not fill to the end of the page.

To remove Section Breaks, press Ctrl+H to issue the Replace command, and then choose the More button to expand the Find and Replace dialog box; With the insertion point in the Find What text box, choose the Special button, and then select Section Break at the bottom of the list. Do not put anything in the Replace With text box; Choose Find Next to find the next section break. The section break is selected behind the dialog box. Continue searching by choosing the Find Next button, or remove this section break by choosing the Replace button; if all the section breaks are to be replaced, choose the Replace All button. Word will tell how many replacements were made; then choose the OK button and Close to return to the document.

Inserting Page Numbers

To insert page numbers, choose Insert, Page Numbers to display the list box, select Bottom of Page (Footer), or Top of Page (Header) and choose from the given options to align the page number horizontally on the page; in the Header & Footer Toolbar Select the Different First Page check box if a page number is not to appear on the first page of the document or there is any other formatting required for the first page; To change format options, choose the Format Page Number button and select from the options on the Page Number Format dialog box; Choose OK to return to the Page Numbers dialog box, and then choose OK or Close to return to the document and save and apply the changes.

Page numbering continues throughout the entire document. There may be instances where the user need to number sections differently. For example, he may want to start each section with page 1. For this, divide

the document into sections by inserting section breaks, and then position the insertion point in the section to apply unique numbering. Choose Insert, Page Numbers, and then choose the Format Page Number button. Make any changes to the options on the Page Number Format dialog box. To restart numbering for this section, in the Page Numbering section of the dialog box, click the Start At option and type or select the starting page number. Then choose OK to return to the document. Like page numbers, Headers and Footers also can be made different for different section.

Page Setup

Changing the default page layout formatting enables the user to choose whether the headers and footers are the same throughout the document, or change from odd page to even page. To change layout, select the header or footer to be formatted. This makes a toolbar appear at the top of the document window. Click one of the options: Different Odd and Even or Different First Page, depending on the needs for the headers and footers in the document; Select the Header/Footer drop-down arrow to choose between Top, Center, or Justified.

To change margins, choose Page Layout. Click on the Margins drop down arrow, either select or type in the margins for all sides in Custom Margins, including the From Edge settings for the Header and Footer in the Layout tab.; If the user will be binding the document and want the inside margin to remain constant, click Mirror Margins to toggle that feature on and off; In the Apply To box, indicate to what portion of the document these changes are to be applied; When finished, choose OK. The default measurement units for margins are inches. To change units, choose File, click Options. Click Advanced tab. In the Display section, change the unit in the show measurement in units of text box.

Paper size can also be changed. To change paper size, choose Page Layout and click on Size. Select an appropriate paper size from the drop down list; if the paper measurements are not included in the Paper Size box, type in the appropriate measurements in the Width and Height text boxes given in the Paper tab in the dialog box when we click on More paper sizes. In the Apply To box, indicate to what portion of the document these changes are to be applied; when finished, choose OK.

The Paper Source tab tells the printer where to go to get the first page of a document and then where to go to get all subsequent pages. To choose paper source, on the same dialog box, select the Paper Source tab; In the First Page list box, select the location or source for the paper of the first page of each document for the printer; In the Other Pages list box, select the location or source for the paper of all subsequent pages for the printer; In the Apply To box, indicate to what portion of the document these changes are to be applied; When finished, choose OK.

Creating Bookmarks

A bookmark is a named marker for a block of text, an entire table or a graphic, a cell or range of cells in a table, or simply a position in a document. First enter a bookmark where it is wanted, then the user can move to it or cite it as a reference in a field or formula. To create a book mark, click in the document at the location where the bookmark is to be inserted, or select the text or graphic to be named; Choose Insert, Bookmark to display the Bookmark dialog box; enter the new bookmark name in the Bookmark Name text box. The user can also select an existing bookmark name from the list, and Word moves the bookmark from its existing location to the place selected; Click the Add button to add the bookmark to the bookmark list and close the dialog box.

To delete a book mark, choose Insert, Bookmark; Select the name of the bookmark to be deleted from the Bookmark Name list box; Click the Delete button. The selected bookmark name is removed from the list box and the document; Click the Close button to close the dialog box and return to the document.

Hyperlinks

The user can create a link that jumps to a document, also called a page, on his computer and on a network or intranet. Two types of hyperlinks can be created: one that jumps to a bookmark in the same document and another that jumps to a URL.

First type is for named locations. Named locations can be in the same document the user is working with (an internal link), or in a different document (an external link). The named location targets a Bookmark inserted at the position to be jumped to within a document. To do this, use the Insert Hyperlink button on the Standard toolbar to open the Insert Hyperlink dialog box; choose either Place in this document or Existing File or Webpage button. Locate the Word document containing the Bookmark this link is to jump to. The user can also jump to a named range in Excel, a database object, or a specific PowerPoint slide; choose OK to confirm the choice or Cancel to negate it.

Second type is for URLs and other links. For this **c**lick the Insert Hyperlink button on the Standard toolbar to open the Insert Hyperlink dialog box; click on the Browse the Web or Browse for File button to locate

and select the document. The path and document name are automatically entered in the Address text box; choose OK to confirm the choice or Cancel to negate it.

Entering Text

Text entries can include a combination of alphabetical characters, numbers, and symbols. By default, when text is entered in a cell, the text automatically aligns on the left side of the cell. Occasionally, a number may be entered as a text entry. To make Excel accept numbers as text, type an apostrophe (') followed by the number.

Entering Numbers

Numbers are constant values containing only the following characters: 0 1 2 3 4 5 6 7 8 9 + - () , / $ % . E e. User can enter integers, such as 24 or 973; decimal fractions, such as 908.37 or 0.72; integer fractions, such as 3 1/4 or 2/3; or scientific notation, such as 5.87137E+3. (See also "Entering: Text" and "Entering: Dates and Times.") . By default, the numbers automatically aligns on the right side of the cell.

To enter a fraction, type an integer, followed by a space, and then the fraction. If only the fractional part is to be entered, type a zero, a space, and then the fraction; otherwise Excel may interpret the entry as a date.

Entering Data in a Selected Range

To speed data entry, one can preselect the range in which data is entered. Then, the active cell will move automatically to another cell in the range after a specified key is pressed.

To enter data in selected range, select the range in which data is to be entered. The first cell in the selected range is active and appears with a white background; Type the data to be entered in the first cell; Press Enter to move down one cell, press Shift+Enter to move up one cell, press Tab to move right one cell, or press Shift+Tab to move left one cell.

Entering Data Series

Excel includes a feature named AutoFill, which enables the user to enter sequences of values automatically. AutoFill can be used for dates, months, years, positive and negative numbers, certain sequences such as Qtr 1, Qtr 2, and so on.

To fill a range with a sequence of numbers, enter the numbers in the first two cells of the range. (These two cells can be in the same column or the same row.) ; Select the two cells, then position the mouse pointer over the handle in the lower-right corner of the selected range; the pointer changes to a cross; Drag the cross to the end of the range to be filled with the sequence and release the mouse button; AutoFill completes the sequence of numbers.

Entering Dates and Times

To quickly enter the current date in a cell, select the cell and press Ctrl+; (semicolon). To enter the current time in a cell, press Ctrl+: (colon). The date and time can be combined in a single cell by separating the date and time with a space.

To manually enter a date or time, enter the date into the cell using any of these formats: 11/6/97, 6-Nov-97, 6-Nov, Nov-97; or, enter the time into the cell using any of these formats: 21:41, 21:41:35, 9:41 PM, 9:41:35 PM. For any other styles, Excel will format the number using one of the formats listed above. If Excel does not recognize the entry as a valid date or time format, the entry is treated as text and, in an unformatted cell, aligns it to the left.

Selecting Multiple Ranges/Worksheets

While working in Excel, more than one range of data can also be selected. Instead of selecting ranges individually and formatting them, all ranges can be formatted at once. To select multiple ranges, select the first range of cells; Hold down the Ctrl key; then click and drag to select the next range of cells; Repeat Step 2 until all the desired ranges are selected.

Complete Worksheet(s) can also be selected, if the user wants to group the worksheets and perform actions on all selected worksheets.

To select a worksheet in the current workbook, click the sheet tab. The worksheet displays; if all cells are to be selected in the current worksheet, click the rectangle that appears at top left corner, at the intersection of the row headers and column headers; To select multiple Worksheets, hold down the Ctrl key; then repeat the above steps.

Hiding Worksheets/Workbooks

The user may want to hide a worksheet in a shared workbook so that others will not see the worksheet. To hide a Worksheet, right click the on sheet tab for the worksheet to be hidden; Choose Hide. To restore the worksheet, right click the on any sheet tab, choose Unhide. In the Unhide dialog box, select the name

of the worksheet to be displayed; then click OK.
To hide a Workbook, the mouse pointer in the workbook to be hidden, choose View, Hide. To restore the workbook, choose View, Unhide. In the Unhide dialog box, select the name of the workbook to be displayed; then click OK.

Inserting and Deleting Columns/Rows

As the user edits worksheet, he may need to insert or delete entire columns/rows in the worksheet. If a column/row is to be inserted, select the column/row header of the column/row to be move to the right when the new column/row is inserted; or, if deleting a column/row, select the column/row header of the column/row to be deleted; To insert a column, on the Home tab, choose Insert, Insert Sheet Columns/Insert Sheet Rows; or, to delete the selected column/row, choose Delete.

Inserting Worksheets

New worksheets can be easily inserted (as many as 255 total) at any time. To insert a Worksheet, open the workbook to which a new worksheet is to be added; Choose Insert, Sheet. The new worksheet is inserted just before the current worksheet; Drag the sheet tab of the new worksheet to where the worksheet is to be appeared.
The default number of worksheets that appears in a new workbook can also be changed. Choose File, Options; then click the General tab. In the Include this many sheets text box, type the number of worksheets to be contained in new workbooks; then click OK.

Naming Worksheets

The user can also assign his descriptive name of up to 31 characters to each worksheet. The characters / \ : ? * [ ] < > cannot be used in the names. To name a Worksheet, double-click the sheet tab for the worksheet to be renamed; Type the new name for the worksheet and press Enter.

Rotating Text

In Excel, the default orientation for text is horizontal, reading left to right. Text can also be rotated to any direction. This can be used for vertical titles for reports or to label the sides of charts, tables, or drawings. To rotate text, select the cell or range containing data to be rotated. Choose Format, Cells; then click the Alignment tab; In the Orientation area, drag the pointer in the second box up or down to change the orientation of the text (as displayed in the Orientation preview box); or, specify a value in the Degrees box between 90 and -90 degrees. Then click OK. Or, select the cell or range containing data to be rotated. In the Home tab, Alignment portion, choose the Orientation button. Select the desired option.

Shrinking Text to fit in a Cell

If the user needs to fit text in a cell without widening the column containing the text, he can shrink the size of the text by using the Shrink to Fit alignment option.
Select the cell or range containing data to be shrunk and right click on it. Choose Format, then click the Alignment tab; Select the Shrink to Fit option; then click OK.

Wrapping Text in a Cell

For long text entry in a cell, Excel can wrap the text so that it forms a paragraph that fits inside that cell by increasing cell's height to accommodate multiple lines of text. Select the cell or range containing data to be wrapped. Choose Format, then click the Alignment tab; Select the Wrap Text option; then click OK.

Charts/Graphs

Microsoft Excel no longer provides the chart wizard. Instead, you can create a basic chart by clicking the chart type that you want on the Insert tab in the Charts group. To create a chart that displays the details that you want, you can then continue with the next steps of the following step-by-step process. A chart has many elements. Some of these elements are displayed by default, others can be added as needed. You can change the display of the chart elements by moving them to other locations in the chart, resizing them, or by changing the format. You can also remove chart elements that you do not want to display. Instead of manually adding or changing chart elements or formatting the chart, you can quickly apply a predefined chart layout and chart style to your chart.

Conditional Formatting

By applying conditional formatting to your data, you can quickly identify variances in a range of values with a quick glance.
To apply conditional Formatting go to the Home tab, in the Styles group, click the arrow next to Conditional Formatting, and then click Color Scales. Hover over the color scale icons to see a preview of the data with conditional formatting applied. In a three-color scale, the top color represents higher values, the middle color represents the medium values, and the bottom color represents the lower values.

Now experiment with the conditional formatting's available styles. After you have applied a style, select your data, click Conditional formatting on the ribbon, and then click manage rules to manually fine tune your rules and formatting.

Entering Formulas

Formulas enable the user to perform calculations by using values in the worksheet. Arithmetic operators that can be used in formulas include + for addition, - for subtraction, * for multiplication, / for division, % for percentage, and ^ for exponentiation. For example:

=a1+b1 adds the contents of a1 and b1 in the cell where the formula is typed.

To fill the same formula in multiple cells, select the adjacent cells or ranges to be filled; With the range(s) still selected, type the formula or value in the active cell; Press Ctrl+Enter (rather than just Enter) to enter the formula or value.

To reference cells in other Worksheets, select the cell where the formula is to be appeared , and type an equal sign (=) to start the formula; Click the sheet tab containing the cell to be referenced in the formula; Select the cell or range to be referred to. The complete reference appears in the formula bar; Finish the rest of the formula; then press Enter to complete the formula.

Excel also provides functions like SUM, AVERAGE, etc. Select the cell where the function is to be appeared, and type an equal sign (=) to start the function; Type the function name (such as SUM, AVERAGE, etc) and a left parenthesis; Select the range of cells for the argument and press Enter. Excel automatically adds the closing parenthesis and enters the function. References to columns or rows can also be entered manually, using comas (,) for separation. If colon (:) is used between two references then, it acts as range operator, which produces one reference to all the cells between two references, including the two reference. Following are some of the most commonly used formula.

- SUM: Adds all the numbers in a range of cells.
  Syntax: =SUM(number1, number2, …..)
- AVERAGE: Returns the average (arithmetic mean) of the arguments.
  Syntax: =AVERAGE(number1, number2, …..)
- COUNT: Counts the number of cells that contain numbers and numbers within the list of arguments.
  Syntax: =COUNT(value1, value2, …..)
- COUNTA: Counts the number of cells that are not empty and the values within the list of arguments.
  Syntax: =COUNTA(value1, value2, …..)
- PRODUCT: Multiplies all the numbers given as arguments and returns the product.
  Syntax: =PRODUCT (number1, number2, …..)
- SUMIF: Adds the cells specified by a given criteria.
  Syntax: SUMIF (range, criteria, sum_range)
  Where 'Range' is the range of cells to be evaluated according to the given criteria. 'Criteria' is the criteria in the form of a number, expression, or text that defines which cells will be added. 'Sum_range' are the actual cells to sum. The cells in sum_range are summed only if their corresponding cells in range match the criteria. If sum_range is omitted, the cells in range are summed.
  For example: Suppose B1:B4 contain the values 100, 200, 300, and 400 respectively.
  C1:C4 contains 1, 4, 6, and 2 respectively. Now the expression SUMIF (B1:B4,">275", C1:C4) will add 6 and 2 as there corresponding values in cells B3 and B4 are greater than 275. Hence the result will be 8. If sum_range argument is omitted then the result becomes 700.
- FACT: gives factorial of a number
  Syntax: FACT (num)
- Large: returns kth largest value
  Syntax: Large (Array, K)
- SMALL: returns kth smallest value
  Syntax: small (array, k)
- Ceiling: returns nearest kth value integer

Formulas: Absolute, Relative, and Mixed Cell References

When a cell contains a formula with references to other cells, several methods can be used to handle those references.

Excel normally uses relative references for cell addresses in a formula, unless specified otherwise. When relative references are used, the cell references in a formula automatically adjust after the formula is copied to another cell or range. If cell B10 contains the formula =SUM(B3:B9), for example, and user copies this formula from cell B10 to cell C10, the new formula in cell C10 automatically adjusts to read =SUM(C3:C9).

To prevent a cell reference in a formula from changing when that formula is copied to another cell or range, use an absolute reference. Absolute references can be indicated by typing a dollar sign ($) in front of the column letter and the row number. In a sales worksheet, for example, if the user have a column of formulas that multiply a value by the commission percentage located in cell D7, he could use D7 to refer to that percentage in the first cell; then copy the formula down the column.

Combinations of these two types of references called mixed references can also be used. For example $C3 prevents the column from changing, C$3 adjusts the column to a new location but the row remains fixed when the formula is copied.

To use this, place the cell pointer in the cell where the formula is to be entered; To enter an absolute or mixed reference in a formula, type an equal sign (=) to start the formula (to enter a relative reference, just type the reference--no special treatment is needed). Then type or click the cell reference; Press F4 until the desired combination of dollar signs appears, and then type the arithmetic operator, such as a plus sign (+); Continue to type other values or cell references and operators as needed; then press Enter to complete the formula.

Creating a Text String

At times, the user may need to create a formula that joins the contents of two cells. Excel refers to this action as concatenation.

For example: If a worksheet includes first names in column B3 and last names in column C3, the user can enter a formula in a third column that joins the first name with the last name:

=B3&" "&C3 where ampersand (&) is the concatenation operator and (" ") simply indicates a space between the two text strings.

Displaying Formula Instead of Results

From anywhere in the worksheet, choose Formulas, click the Show Formulas button.

To display the results again, choose Formulas, click the Show Formulas button again. Press Ctrl+ ' (the grave accent, usually located on the same key as the tilde character) to toggle between viewing results and formulas.

Auto Calculate

Select the range to be summed. The Calculate button in the status bar automatically displays the sum of the selected range (or right click the Auto Calculate button to see more functions); Right-click the Auto Calculate button in the status bar; From the pop-up menu that appears, select the function to be used, such as Average or Count. The result of the function selected appears in the status bar.

Logical Functions

The logical functions enable the user to add decision-making and logical tests to the worksheets. The IF statement is useful for testing conditions and making decisions based on a cell's contents. The AND and OR functions can test multiple criteria or test conditions for use in IF functions. The following examples show the use:

- =AND(D15,G23<30) result is TRUE only when D15 is not zero and G23 is less than 30
- =IF(AND(D17>10,D17<30),"Valid","Invalid") returns Validif the contents of cell D17 is greater than 10 and less than 30; otherwise the formula returns Invalid
- =IF(NOT(OR(D17=10,D17=30)),"Not 10 or 30","Contains 10 or 30") tests whether cell D17 contains the result 10 or 30 and produces the message Not 10 or 30 when the cell does not contain either of those results; otherwise, the formula result is Contains 10 or 30
- =IF(OR(D17=10,D17=30),"Contains 10 or 30","Not 10 or 30") tests whether cell D17 contains the result 10 or 30 and produces the message Contains 10 or 30 when it does; otherwise, the formula produces the message Not 10 or 30

Math and Trigonometric Functions

Like logical functions, math and trigonometric functions can also be used in the Worksheets. Trigonometric functions use angles measured in radians.

Following are some examples:

- =ABS(A10) returns 18when cell A10 contains -18
- =ACOS(0.5) returns 1.047198 (radians)
- =DEGREES(0.5) returns 28.64789 (degrees)
- =LOG(12,3) returns 2.26186
- =ROUND(102.927,2) returns 102.93

Statistical Functions

Similarly, statistical functions can als0 be used in the Worksheets. For example:

- =COUNTIF(B1:B4,">100") returns 1if the range B1:B4 contains the numbers 57, 102, 84, and 98
- =MEDIAN(1,4,2,6,9) returns 4
- =MEDIAN(1,4,2,6,9,10) returns 5 (average of the two middle values, 4 and 6)

Print Titles

Page layout then go to Print titles then go to Rows to repeat at top then go to $1:$n

LABORATORY TASKS:

1. Create a word document (Properly Formatted), using mail merge wizard, which produces individual information reports for a list of students in the form of a table, including their first name, last name, father's name, home address, phone number, email addresses, and home page address as hyperlinks (if any). Enter few sample records. Table should be properly formatted. Remember to include your name and roll number in the header field. Get a printout to be attached here, of the page that contains only field name and not any records. Also attach 1 sample report containing actual data.

Attach the colored printout here.

2. Design a fruit name puzzle as given below. It has four fruit names. Mango, Apple, Pineapple, and Orange.

				M				
P	I	N	E	A	P	P	L	E
				N				
O	R	A	N	G	E			
				O				

WELL DONE !!!

For a user the puzzle will be blank. Enter Hints for solving the Puzzle. A box will contain the text "Keep Trying" if the puzzle is unsolved or incorrectly solved. If the user solve puzzle correctly then this box contains "WELL DONE !!!". If the user enters correct alphabet it should be highlighted as BLUE. If the user enters wrong alphabet it should be highlighted as RED. Format your work accordingly.

Hint: Use Conditional Formatting.

Attach the colored printout here.

3. Design a TEMPERATURE CONVERTER from Celsius to Fahrenheit and Fahrenheit to Celsius. Present the Converter with appropriate design. Formulas for conversion are as follows:

$[°C] = ([°F] - 32) \cdot 5/9$ $[°F] = [°C] \cdot 9/5 + 32$

Also report the weather conditions as HOT if the temperature exceeds 35°C, WARM if the temperature is between 20°C to 35°C and COLD if the entered temperature is below 20°C.

4. Create a magic square puzzle as per given example below, the sum of all the numbers in a row must be equal, simultaneously the sum of all the numbers in a column must be equal, and the sum of diagonal numbers should also be equal:

	12	3	9	24
	5	8	11	24
	7	13	4	24
24	24	24	24	24

WELL DONE!!!

Take the input from the user in all the squares. For a user all the squares will be blank initially and the box given below will contain the text "KEEP TRYING" unless the user enters all correct entries. If user solves the puzzle correctly then a message "WELL DONE" appears below. Format your work accordingly. **Attach the printed output**

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 04

Objective:

To understand basics of Python programming language and explore its development environment.

Theory:

Python is an object-oriented, high-level programming language that was created in 1991 by Guido Van Rossum, a Dutch programmer. Its straightforward syntax and superior ability to convey concepts in fewer lines of code have made it one of the most widely used computer languages today. An Integrated Learning Environment (IDE) or Integrated programming and Learning Environment (IDLE) is required in order to write a Python program and update it. The Python programming language comes with an IDE called Python IDLE, which makes it simple for users to edit, build, execute, and compile Python 2.x or Python 3.x algorithms. The typical Python distribution includes the Python IDLE, an integrated programming environment for Python.

Interactive Interpreter

A key part of various programming languages, including Python, is the interactive interpreter, which simply known as the interpreter. With instant feedback, it enables users to input and carry out instructions or statements at the same time. A script or program running in its entirety is not the same as this interactive mode.

File Editor

One of the most important tasks for a programmer is editing, and Python IDLE offers a feature-rich, full-featured editor. In Python IDLE, a file editor is a tool for writing and running Python scripts. Since the.py extension identifies the file as a Python file, it should be used when saving Python IDLE files. In addition, it has a number of features that make it user-friendly for beginners and improve workflow, such as code completion, call hints, and automatic indentation.

Breakpoint

Breakpoints are places in program where it allows the interpreter to terminate execution. However, before implementing a breakpoint, it is critical to ensure that the debugger mode is enabled. The breakpoint is created by first right-clicking on the portion of code you want to stop at, which will cause the code to turn yellow. Breakpoints can be established as many times as needed whereas to disable the breakpoint and reverse the previous action, right-click the same line again. One can step through the code in a debugger window and manually run breakpoint code by activating debugger mode.

Python Shell

The Python language allows its users access to a Python Shell so they can execute specific commands and observe the results. REPL, which stands for "Read, Evaluate, Print, Loop," is another name for the Python Standard Shell. It reads the corresponding command, then evaluates it, produces its outcome, and ultimately loops back to grasp the command once again. You must first use the command prompt (also called Terminal for MacOS or PowerShell for Windows) in order to launch Python Shell. After that, enter a Python command and hit "Enter." The appearance of the Python Shell/REPL window is shown in Figure 4.1.

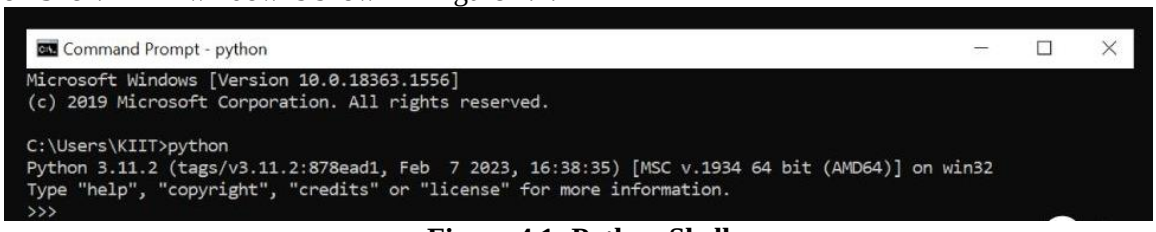


Figure 4.1: Python Shell

Double-click IDLE after typing it into the search bar then the Python IDLE will launch, allowing one to begin writing and running Python code. It should be noted that Python IDLE is pre-installed with the Python package and does not require separate download. It will display IDLE window as shown in Figure 4.2 that allows user to type and run the source code. Here, a few basic codes have been created, such as 3+2, which equals 5, yields the output in next line.

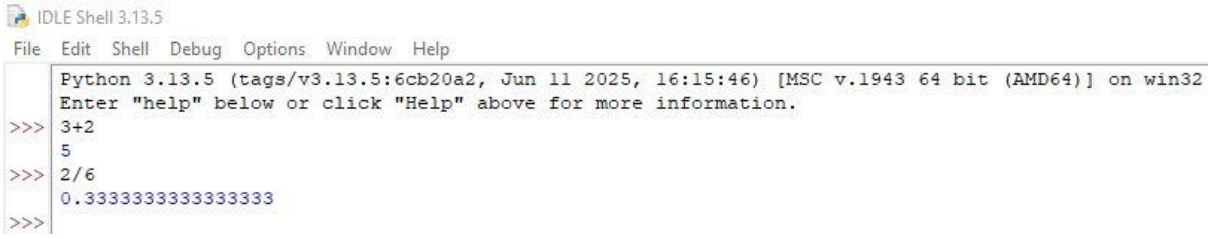


Figure 4.2: Python IDLE

Working with Files in Python IDLE

Python IDLE comes with a comprehensive editor that lets you run the code directly from within it. Additionally, it saves time when programming by having code completion and automatic indentation. The discussion on accessing a module, editing Python files, dealing with an existing file, and creating a new one is provided below:

Opening a New File

As previously mentioned, you can launch Python IDLE from the menu bar by double-clicking its icon. Select the File option. Next, from the drop-down menu, choose "New File".

Opening an Existing File

The procedures listed below should be followed in order to deal with an existing Python file:

Step 1: With a few exceptions, the process for opening an existing file is the same as mentioned earlier. From the start menu, choose IDLE twice.

Step 2: Select the file option from the drop-down menu and then click on open to open an existing Python file.

Step 3: Choose the particular file to open and proceed. Clicking the 'Open' option will then open the file you have selected in the IDLE. You can continue editing the document and save it by using the Ctrl + S or save button.

Access a Module in IDLE

The File and then Path Browser can be selected to view the source code for a Python module. This can be used to explore the modules that Python IDLE can recognize. Double-click on it, and the file editor will launch so you can view it. This call to sys.path will return paths that directly match the contents of the window. If you already know what the module is named, you may choose File then Module Browser and enter its name in the box that appears.

Editing a Python File

Once a Python file has been opened by the IDLE, you can modify it. There are several choices available to you when making changes to a file and saving your work. At the top of the window, the Python IDLE features a menu bar with several crucial functionalities. These include the ability to change the file's name, modify or monitor the folder in which the file is located, determine the Python software version, and more. To access the desired function, simply navigate to the top menu ribbon and select it. It will display terms like Ln and Col in the lower right corner of the window as shown in the Figure 4.3. In this case, the Ln indicates the quantity of lines of code present in the program, and the Col indicates specific column the cursor has been placed in. These assist in spotting bugs when coding. By closely examining the Python IDLE window, IDLE employs an asterisk to show that the document has any unsaved modifications. This feature serves as a reminder to save work after making edits to the file.

Figure 4.3: Representation of Line and Column Number

Execute a File in Python IDLE

The first step is to write your code so, in editor window, write a program. After completing that, *untitled will appear at the top of the editor window, indicating that the code or file being edited has not yet been saved which takes to the next step. It should be confirmed that the file has been saved when finished writing the program. As illustrated in Figure 4.4, save the file by clicking the 'File' option on the toolbar at the top of the window and choosing 'Save as'. While developing code, program can be saved at any point.

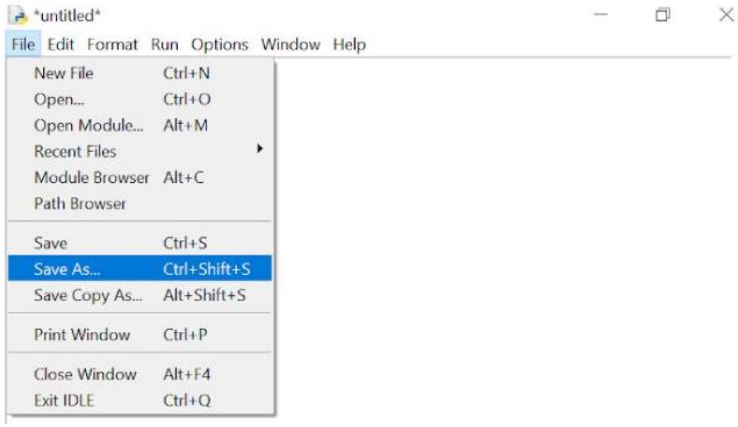


Figure 4.4: Saving the file

Saving your file with the.py suffix is crucial since it identifies it as a Python file. The file fails to run and execute without.py because the interpreter cannot be able to identify the file type and gives an error instead. The file can be saved in any folder or at any location of choice. As seen in Figure 4.5, the file can be easily run by pressing F5 on your keyboard or by selecting Run followed by Run Module from the menu bar at the top. It should be noted that selecting the latter approach will launch the interpreter before running your code. The output will be shown in the shell once the program has been executed.

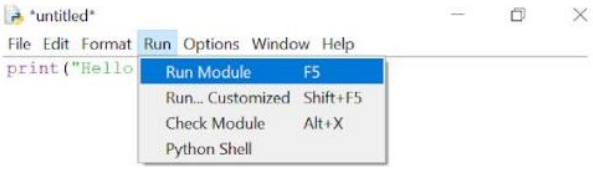


Figure 4.5: Execution of File

Debugging in Python IDLE

An unexpected issue that arises in program is called a bug. Sometimes they are simple to fix, but other times they are difficult to comprehend. It may be hard to keep track of some of them, and simply scanning the program cannot be enough to catch them, hence, debuggers can help in this situation. Python IDLE offers a debugger to make debugging the code easier and more effective. Following are the steps to debug a program in IDLE:

1. **Create a Breakpoint:** In order to stop the execution of code, create a breakpoint at that line to begin debugging. By clicking in the left margin next to the line number, a breakpoint can be set and it will be shown by a red dot.
2. **Start Debugging:** To run the program in debug mode, choose 'Debug' from the menu, then click 'Go' or F5. The shortcut Ctrl + F5 may also be used for the same purpose.
3. **Debug Mode**: The window will indicate that the code is functioning and that Debug Mode is enabled once you execute it in this manner.
4. **Debugger Controls:** The execution will be stopped and the debugger controls will activate when the debugger reaches the breakpoint. Typical controls consist of:

- **Step Into (F7):** This allows you to step into a function by executing that particular line of code and stopping if it determines a function.

- **Step Over (F8):** Even if the line of code is inside a function, it runs the current line and terminates at the next line.
- **Continue (F5):** Continue running the code until the subsequent breakpoint is reached.
- **Quit (Ctrl + F6):** This pauses the script and exits the debugger.

5. **Inspect Variables:** The values of variables can be examined when debugging. The call stack and values of variables at every stage are shown in Python IDLE's 'Stack Viewer' window.
6. **Navigate Through the Code:** To determine the origin of any errors in the code, use the debugger controls. One can examine variable values, move through the code, and comprehend how it runs.
7. **Modify Code and Continue:** The code can be altered directly in the editor window while debugging. Once the modifications have been made, choose the 'Run' option from the menu and then click the 'Run Module' button or the F5 key to continue running the updated code.
8. **Finish Debugging:** The debugging can be completed after locating and fixing the bugs in the code. Either choose the 'Debug' option from the menu and select the 'Quit' button, or let the script run through to the end. Alternatively, this can be accomplished by using the shortcut Ctrl + F6.
9. **Examine Output** The final stage of debugging is to examine the output in the Python IDLE Shell to make sure the script functions error-free and as expected.

Errors and Exceptions

When an error output in a Python interpreter is encountered, IDLE allows to navigate straight from the menu bar to the problematic line. The cursor must be moved to mark the line in the code then select Debug, and choose Go to file or line from the menu bar as shown in Figure 4.6.

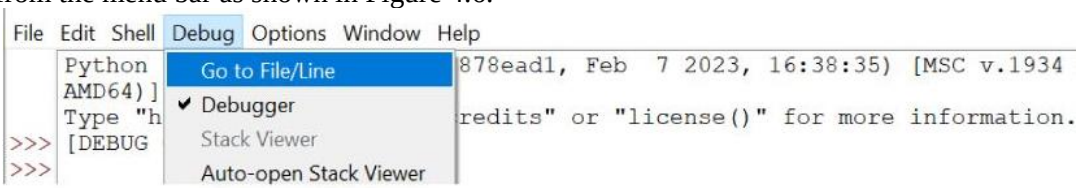


Figure 4.6: Navigating to the error

Customizing Python IDLE

IDLE has a few features that allow for customization such as modifying features like the font, windows, highlights, keys, and extensions. By choosing a setting, the IDLE working experience can be enhanced.

LABORATORY TASKS:

1. Open CMD or Terminal. Type python or python3. Observe the output of following program.

```
print("Hello from terminal shell")
exit()
```

Ans:_____
_____
_____
_____

2. Type the following programs in Python IDE and execute it. What is the output? Observe the output and identify the difference.
   i. 4 + 4
   ii. print("Hello from Python Shell")

Ans:_____
_____
_____
_____

3. Use IDLE's file editor to create and save a Python script and execute it. Write the output.

```
name = "Alice"
print("Welcome", name)
```

Ans:_____
_____
_____
_____

4. Execute the following program. Note what error appears and how IDLE points to it. Correct it and re-run.

```
name = "Ali
print("Welcome", name)
```

Ans:_____
_____
_____
_____

5. Use IDLE's file editor to create and save a following program. Right-click to set a breakpoint on z = x + y. Run in debug mode and observe the debugger pause.

```
x = 10
y = 5
z = x + y
print(z)
```

Ans:_____
_____
_____
_____

6. Execute the following lines in Python Shell and mention what REPL means in this context.
   5 * 6
   "Python" + " IDLE"
   len("Welcome")

Ans:_____
_____
_____
_____

7. Try running this program with incorrect indentation:
   def greet():
   print("Hello")

Ans:_____
_____
_____
_____

8. In IDLE, go to File → Module Browser. Type random and explore the list of available functions.

Ans:_____
_____
_____
_____

9. Customize IDLE Settings. Change Font size to 14 pt, Theme to IDLE Dark, Window Preferences. Write your observations.

Ans:_____
_____
_____
_____

10. Open File → Path Browser. List 5 standard modules and explore its contents.

Ans:_____
_____
_____
_____

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 05

Objective:

To understand variables, data types, input/output functions, and basic operators used in the Python programming.

Theory:

Variable

A Python variable is a specific part of memory set aside for storing values that provide the computer with information to process. Python assigns a datatype to each value that is assigned to the variable. In Python, there are several data types, including numbers, lists, tuples, strings, dictionaries, etc. Any name, including alphabets, can be used to declare variables. A built-in Python function known as the del() function can be used to remove a variable from the memory. The main reason this delete method was created was to address the problem of limited memory that may arise from Python storing unneeded variable values.

```
# Declare a variable and initialize it
f = 0
print (f)
# re-declaring the variable works
f = 'Ali'
print (f)
```

Assigning Different Values to Multiple Variables at Once

Multiple variables can be initialized with different values in the same line using one of two methods. The first option is to use commas to separate the variable names, and then use the assignment operator to equate them with the values that intend to assign. In the second approach, variables can be assigned their corresponding values using a list. To assign values in the correct sequence, just equate Python variables, separating them by commas, to a list. It should be noted that the distinct values can also be equated.

Constants

One kind of variable whose value is unchangeable is called a constant. Constants are typically defined and allocated on a module in Python. As shown in Figure 5.1, the module in this context refers to a new file that is imported into the main file and contains variables, functions, etc. Constants are written inside the module with underscores between words and all capital letters.

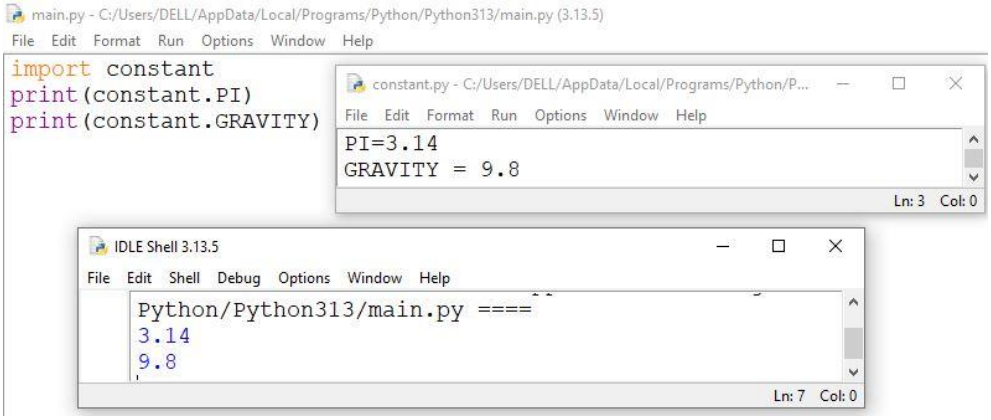


Figure 5.1: Using constants

Input/Output Functions

- input() is used to take user input (always as a string)
- print() is used to display output

Data Types

Python variables can store a wide variety of data and values, including arrays, multi-dimensional arrays, and numeric values. It may even be a sophisticated data structure, such as binary trees, maps, or graphs. A few typical data types are displayed in Figure 5.2.

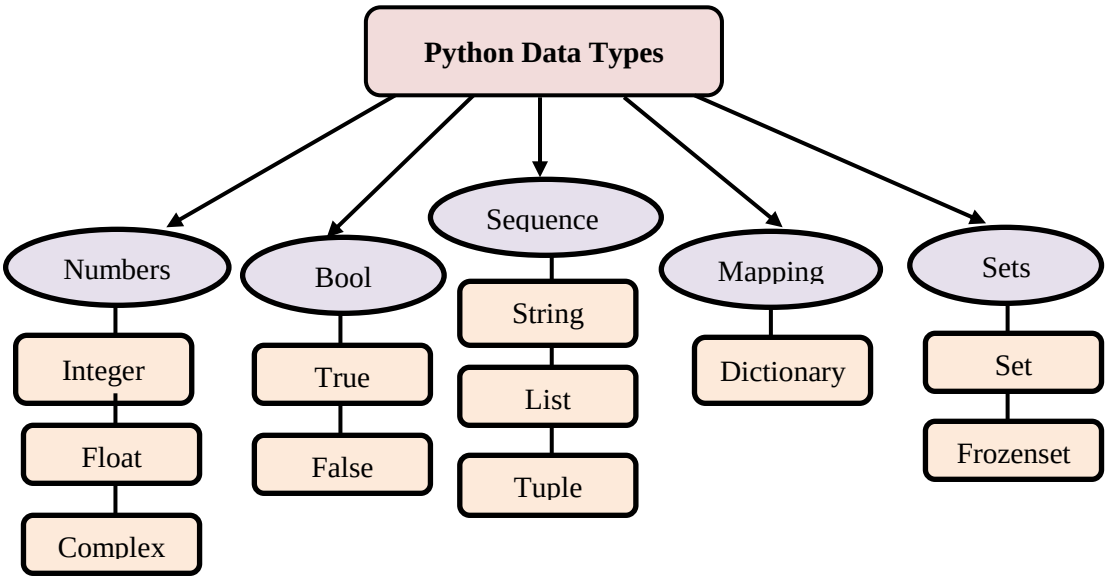


Figure 5.2: Data types in Python

1. **Numeric Types:**
   - **int:** Represents integer values (e.g., 5, -10, 0).
   - **float:** Represents floating-point numbers with decimal places (e.g., 3.14, -2.5, 0.0).
   - **complex:** Represents complex numbers in the form of a + bj, where a and b are real numbers and j is the iota used to represent imaginary numbers (e.g., 2+3j).
2. **Sequence Types:**
   - **str:** String represents a text or a word enclosed in single quotes (') or double quotes ("") (e.g., "Unstop", 'variable', "text").
   - **list:** Represents an ordered collection of items enclosed in square brackets ([]), where each item can be of any datatype (e.g., [1, 2, 3], ['a', 'b', 'c'], [1, 'two', 3.0]).
   - **tuple:** Similar to lists but immutable, meaning their elements cannot be modified once defined. They are enclosed in parentheses (()) (e.g., (1, 2, 3), ('a', 'b', 'c'), (1, 'two', 3.0)).
3. **Mapping Type:**
   - **dict:** Represents an unordered collection of key-value pairs enclosed in curly braces ({}). You can access any value if you have the other one. Keys are unique and used to access values (e.g., {'name': 'John', 'age': 30}).
4. **Set Types:**
   - **Set:** It represents an unordered collection of unique elements enclosed in curly braces ({}). Duplicates are automatically removed. Eg: set_variable = {1, 2, 3, "cherry", True}
   - frozenset: Similar to sets but immutable. Once defined, its elements cannot be modified (e.g., frozenset({1, 2, 3}), frozenset({'a', 'b', 'c'})).

5. **Boolean Type:**
   It represents the truth values, True or False. You can even use 0 and 1 in their places where 0 represents False, and 1 represents True. Eg: boolean_variable = True
6. **None Type:**
   It represents the absence of a value or a null value. Eg: none_variable = None

Arithmetic Operators

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b
- Subtraction	Subtracts right hand operand from left hand operand.	a – b
* Multiplication	Multiplies values on either side of the operator	a * b
/ Division	Divides left hand operand by right hand operand	b / a
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a
** Exponent	Performs exponential (power) calculation on operators	a**b
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity)	9//2 = 4 and 9.0//2.0= 4.0, -11.0//3 = -4.0

Comparison Operators

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b)
!=	If values of two operands are not equal, then condition becomes true.	(a != b)
<>	If values of two operands are not equal, then condition becomes true.	(a <> b). This is similar to != operator.

>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b)
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b)
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b)
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b)

Assignment Operators

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+=	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*=	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/=	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%=	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**=	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//=	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Logical Operators

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b)
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b)
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b)

Membership Operators

Membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below.

Operator	Description	Example
In	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

LABORATORY TASKS:

1. Write a Python program to input two numbers from the user and display their sum.
2. Write a program to declare different types of variables (int, float, string, list, dict) and print their values and types using the type() function.
3. Write a program to convert a string "123" to integer and float, and display their types and values.
4. Write a program that uses and, or, and not to evaluate simple conditions and display the results.
5. Write a program to check if 'a' is in the string "apple" and if 10 is in the list [5, 10, 15].
6. Write a program to demonstrate the use of +=, -=, *=, /=, and **= on a variable.
7. Compare two values using ==, !=, <, >, <=, and >= and display the results.

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 06

Objective:

To apply string operations and utilize built-in methods for string handling in Python.

Theory:

String

In Python, a string is a collection of characters encapsulated in single quotes ('), double quotations ("), or even triple quotes). Strings cannot be altered after they are formed because they are immutable. The following lists the three methods for making a string.

str1 = 'Hello' #Single quotes method
str2 = "Python" #Double quotes method
str3 = '''Triple-quoted strings can span multiple lines.'''

Key Features of Strings in Python

- **Indexed and Iterable:** Use indexing (s[0], s[-1]) to access characters.
- **Immutable:** New strings must be produced; old ones cannot be changed.
- **Supports Various Operations:** Python enables a number of string operations, including formatting, slicing, concatenation, and more.
- **Dynamic Length:** It varies according to the content and has no set size.
- **Unicode Support:** Characters from various languages can be handled by it.
- **Rich Built-in Methods:** The Python library includes a number of methods (.lower(),.replace(),.find(), etc.) for manipulating strings, including finding, adapting, and analyzing text.

Concatenation for String Manipulation in Python

The technique of connecting two or more strings to form a new string is called concatenation. The addition operator (+) is the main tool for doing this, however the join() function is also an option.

String Comparison in Python

Two strings can be compared to determine if they are equal or what their lexicographical order is. The Python logical operators (and, or, not) and the relational operators (==,!=, <, >, <=, and >=) for these comparisons are employed.

Slicing for String Manipulation in Python

Slicing is a popular manipulation method that uses index positions to extract a section of a string. The character's position in a string is denoted by its index, which starts at 0 for the first character and so forth. The string's start, end, and step values must be specified when slicing. The syntax is given as:

Syntax:

string[start:end:step]

- **start**– The index where the slice begins (inclusive).
- **end**– The index where the slice stops (exclusive).
- **step**– The interval between characters (optional).

Code Example:

```
>>> text = "PythonProgramming"
>>> print(text[0:6])  # Extracts 1st to 6th charcaters
Python
>>> print(text[:6])   # Equivalent to text[0:6]
Python
>>> print(text[7:])   # Extracts everything after the 7th character (including it)
rogramming
>>> print(text[::2])  # Extracts every 2nd character beginning from 1st
PtoPormig
```

String Replacement Manipulation

By substituting new values for particular characters or substrings, string replacement enables you to change portions of a string. The built-in Python function replace() is the most popular way to perform replacement.

String Reversion Manipulation

When a string is reversed, its characters are flipped, making the last one in the sequence the first, and so on. Reversal manipulation techniques are frequently required for data transformation tasks and text formatting. Python offers a variety of methods for accomplishing this, including recursion, iterative techniques such as the for and while loops, slicing (the most effective), and more. The slicing notation is shown as an example below:

Code Example:

```
>>> #Creating a string
>>> text = "Python"
>>> #Using slicing with print to reverse and print
>>> print(text[::-1]) # Using slicing
nohtyP
```

String Formatting

The process of dynamically adding values to a string in an organized manner and formatting string data to satisfy particular specifications is known as string formatting. Using f-strings is the most popular method for modifying string format in Python.

Code Example:

```
>>> #Creating a string and integer variable
>>> name = "Ali"
>>> age = 35
>>> # Using f-strings
>>> print(f"My name is {name} and I am {age} years old.")
My name is Ali and I am 35 years old.
```

The Length of a String

Finding the length or count of the characters in a string is a crucial Python string manipulation task. The len() function, which counts all characters, involving unique symbols and spaces, is the most simplest approach to accomplishing this.

Conversion of String in Python

Converting strings to other data types for compatibility and other purposes is another significant kind of string manipulation in Python. For efficient conversion, Python has built-in functions such as:

- Use str() to convert an integer or float to a string.
- Use int() to transform a string to an integer.
- Use list() to turn a string into a list of characters.

Methods for String Manipulation

Python has a number of built-in string methods designed primarily for effective text analysis, modification, and manipulation (i.e., string type variables). Among the crucial functions are (but are not restricted to):

- Case Conversion: Change text to capitalized words (.capitalize()), lowercased words (.lower()), or uppercased words (.upper()).
- Searching & Finding: Use .index(), .find(), .endswith() or .startswith() to look for substrings.
- Replacing & Modifying: Use .replace() to swap out words, .strip() to remove spaces, or .split() to break down text.
- Joining Strings: To combine a list of strings into a single string, use the .join() function.
- Content/Character Type Check: Determine if a string is whitespace (.isspace()), numeric (.isdigit()),

or alphabetic (.isalpha()).

LABORATORY TASKS

1. Write a program to generate a university email id that takes the user's name and roll no: as input, then displays an email using concatenation method.

Ans:_____
_____
_____

2. Write a program for a support system that takes an email address as input from the user and extracts the domain name using the find method.

Ans:_____
_____
_____

3. Create a program that simulates a simple online admission/registration system. The system should prompt the user to enter their full name and age. You must validate these inputs using following string methods:
   For validating the user's name: isalpha(), .istitle(), .isspace(), .strip(), .isupper() / .islower()
   For validating the user's age: .isdigit(),.isnumeric(),.isdecimal(),.strip()

Ans:_____
_____
_____

4. Write a program that takes a CNIC number in the format "42101-1234567-1" and displays only the first six characters. The remaining digits should be replaced with "XXXXXX-X" to protect user privacy. Use string slicing and replace methods.

Ans:_____
_____
_____

5. Write a Python program that prompts the user (seller) to enter a short product description. The program should count and display the number of characters entered.

Ans:_____
_____
_____

6. Write a program that takes a customer's full address stored in a single string and converts it into a label-friendly format. Use appropriate string operations to split the address into parts and rejoin them with " | " as a separator and print the output.

Ans:_____
_____
_____

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 07

Objective:

To implement conditional statements in Python using if, if-else, and nested conditions.

Theory:

Making decisions involves anticipating circumstances that may arise during program execution and defining appropriate course of action based on those circumstances. Multiple expressions that result in TRUE or FALSE are evaluated by decision structures. If the result is TRUE or FALSE, it must decide which statement to run and what action to take. The general form of a common decision-making framework that is present in the majority of computer languages is shown in Figure 7.1.

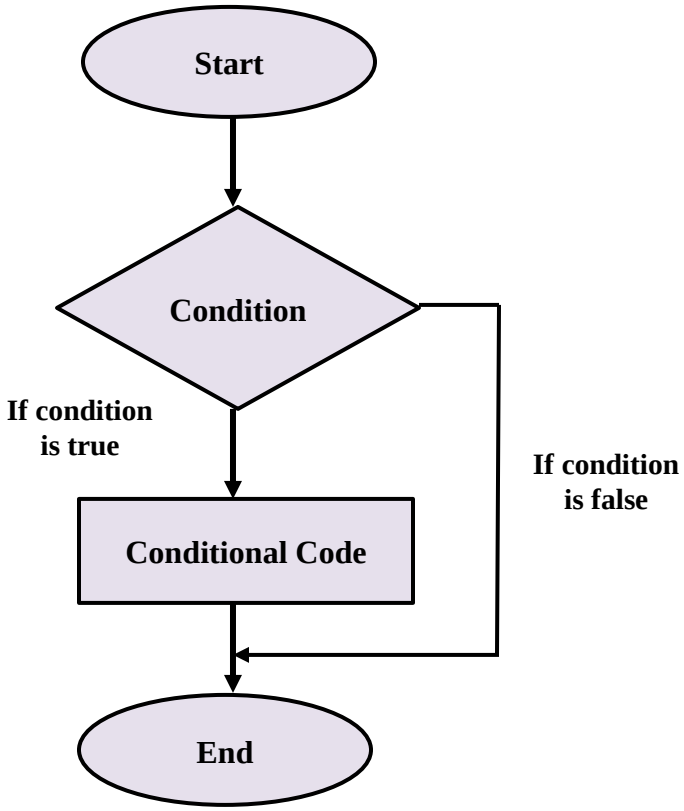


Figure 7.1: Flow chart for conditional statements

Any non-zero or non-null value is presumed to be TRUE by the Python programming language, and FALSE if it is either zero or null. The following categories of decision-making statements are available in the Python programming language:

Sr.No.	Statement & Description
1	**if statements** A boolean expression is followed by one or more sentences to form an if statement. If a condition is true, it checks it and then runs the related piece of code. **Syntax:** if condition: # code to execute if condition is true

2	**if...else statements**
	When the boolean expression is FALSE, an optional else sentence that comes after an if statement takes effect.
	Syntax:
	if condition:
	# code to execute if condition is true
	else:
	# code to execute if condition is false

3	**If-Elif-Else Statements**
	Multiple conditions can be checked sequentially using the if-elif-else statement. It examines the subsequent condition in the elif block if the initial condition is false, and it proceeds until it either finds a true condition or gets to the else block.
	Syntax:
	if condition1:
	# code to execute if condition1 is true
	elif condition2:
	# code to execute if condition2 is true
	else:
	# code to execute if all conditions are false

4	**nested if statements** You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Exception Handling Control Statements in Python

They offer an organized method for dealing with exceptions or errors that arise while a program is running. Programs may avoid unplanned crashes and make sure they handle unanticipated problems smoothly by controlling exceptions. Python has a number of control statements, including try, except, and finally, for managing exceptions.

Try-Except Statement

The try-except statement enables the testing of a piece of code for possible errors. To handle the error, the program proceeds to the except block if an exception arises inside the try block.

Syntax:

```
try:
  # code that might cause an exception
except ExceptionType:
 # code to handle the exception
```

Finally Statement

The code block provided by the finally statement will execute irrespective of an exception was triggered. It is frequently applied to cleaning tasks like resource releases and file closures.

Syntax:

```
try:
 # code that might cause an exception
except ExceptionType:
 # code to handle the exception
finally:
 # code that always executes, regardless of exceptions
```

LABORATORY TASKS

1. Write a program that accepts marks from the user and prints the grade using if-elif-else. Also, check if the input is a valid number using exception handling.
2. Create a program to check if the entered email contains '@' and ends with '.com'. Display appropriate messages. Use string methods.
3. Develop a program where the user enters the total order amount. If it exceeds Rs. 5000, apply a 10% discount; else, no discount. Print the final bill.
4. Ask the user to input the number of days a book is returned late. Calculate the fine: Rs. 10/day up to 7 days, otherwise Rs. 20/days.
5. Create a menu of 5 items using a list. Ask the user for an item. Use membership operators to check if it's available, and print the result.
6. Input height and weight from the user and calculate BMI. Based on the value, use if-elif-else to print whether the user is underweight, normal, or overweight.

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 08

Objective:

To implement conditional statements in Python using if, if-else, and nested conditions.

Theory:

Loop Control Statements in Python

Constructs known as looping control statements enable us to continually run a block of code in response to a certain circumstance. These statements are crucial for automating repetitious operations, which improves the effectiveness and simplicity of Python programs. For loops and while loops are the two major kinds of looping control statements that Python offers.

For Loop

A range of integers or a sequence (such as a tuple, list or string) can be iterated over using the for loop. For every item in the sequence, it runs the chunk of code.

Syntax:

for variable in sequence:
  # code to execute for each item in the sequence

Code Example:

```
File  Edit  Format  Run  Options  Window  Help
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

```
IDLE Shell 3.13.5
File  Edit  Shell  Debug  Options  Window  Help
l\Programs\Python\Python313\hello.py ==========
=
apple
banana
cherry
                                                Ln: 19  Col: 0
```

While Loop

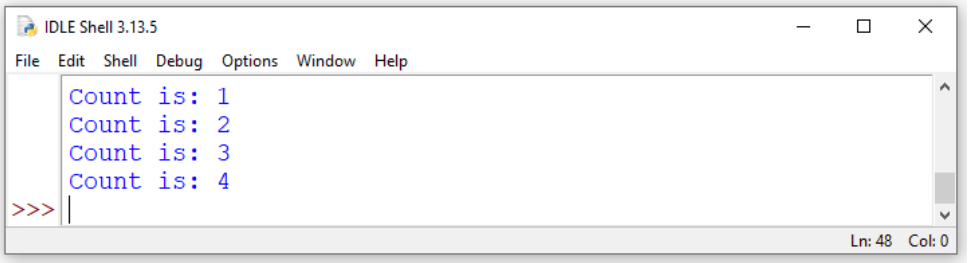
A block of code is executed by the while loop as long as a given condition is met. It is especially helpful in situations when the number of iterations is unknown in advance.

Syntax:

while condition:
  # code to execute as long as the condition is true

Code Example:

```
File  Edit  Format  Run  Options  Window  Help
count =0
while count <5:
    print("Count is:", count)
    count+=1
```



Control Flow Altering Statements

It is possible to modify the typical order of execution in conditional blocks or loops by using control flow changing statements. These statements, which include continue, break and pass, offer strong methods for working with conditionals and loops by either skipping, exiting, or doing nothing in a block of code.

Break Statement

The break statement instantly ends the enclosing loop by stopping additional iterations and terminating the loop.

Syntax:

break

Continue Statement

The continue statement jumps straight to the following iteration, bypassing the current one. It is helpful in bypassing the specific loop conditions.

Syntax:

Continue

Pass Statement

In order for the code to proceed to the following line, the pass statement serves as a placeholder that does nothing. It is frequently employed in the absence of any code to run but syntax demands a statement.

Syntax:

pass

LABORATORY TASKS

1. Write a program for a class teacher who wants to record attendance of students based on roll numbers from 1 to 5 using for loop that prints "Present" next to each student's roll number.
2. Write a program for a website that gives users 3 attempts to enter a correct 4-digit OTP using a while loop to let the user input an OTP. Exit the loop if correct OTP is entered, else repeat for a maximum of 3 attempts.
3. Write a program for a store inventory system that skips out-of-stock items during listing. From a list of items, use continue to skip any item marked "Out of Stock" and print the rest.
4. A factory automation system needs to stop processing if a "Malfunction" is detected. Write a program that iterates over a list of machine statuses. Use break to stop processing if "Malfunction" is found.
5. A software developer is building a menu, but some features are not ready yet. Use for loop to iterate over menu options and use pass for the "Coming Soon" options.
6. A secure login allows a maximum of 5 wrong password attempts. Use a while loop to simulate login attempts and terminate the loop on correct input or after 5 tries.

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 09

Objective:

To study collection data types including lists, tuples, sets, and dictionaries.

Theory:

Python List

One of Python's most flexible datatypes is the list, which may be expressed as a list of values (items) separated by commas enclosed in square brackets. The important thing to remember about lists is that they all are not required to be of the same type.

For example:

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5]; list3 = ["a", "b", "c", "d"]

List indices begin at 0 and can be concatenated, sliced and so on, similar to string indices.

Accessing Values in Lists

For accessing values in lists, use the index or indices to determine the value located at that index and the square brackets for slicing. For example:

```
File  Edit  Format  Run  Options  Window  Help
list1= ['physics','chemistry',1997,2000];
list2= [1,2,3,4,5,6,7];
print ("list1[0]: ", list1[0])
print ("list2 [1:5]:", list2[1:5])
```

```
IDLE Shell 3.13.5
File  Edit  Shell  Debug  Options  Window  Help
    56
    42
    23
    85
    45
>>>
    = RESTART: C:/Users/DELL/AppData/Local/Progra
    ms/Python/Python313/Array.py
    list1[0]:  physics
    list2 [1:5]: [2, 3, 4, 5]
>>>
                                        Ln: 18  Col: 0
```

The append() method can be used to add to items in a list, and the slice on the left-hand side of the assignment operator can be used to alter one or more list entries. If it is known exactly which element or elements you are deleting, the del statement can be used; if not, then the remove() method can be used.

Basic List Operations

Similar to strings, lists react to the + and * operators, which also indicate concatenation and repetition, but the outcome is a new list rather than a string.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

Indexing, Slicing, and Matrixes in List

Since lists are sequences, they are indexed and sliced in the same manner as strings. Considering the following input:
L = ['spam', 'Spam', 'SPAM!']

Python Expression	Results	Description
L[2]	SPAM!	Offsets start at zero
L[-2]	Spam	ative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

Built-in List Functions & Methods
Python includes the following list functions:

1. **cmp(list1, list2):** Compares elements of both lists.
2. **len(list):** Gives the total length of the list.
3. **max(list):** Returns item from the list with max value.
4. **min(list):** Returns item from the list with min value.
5. **list(seq):** Converts a tuple into list.

Python includes following list methods:

1. **list.append(obj):** Appends object obj to list
2. **list.count(obj):** Returns count of how many times obj occurs in list
3. **list.extend(seq):** Appends the contents of seq to list
4. **list.index(obj):** Returns the lowest index in list that obj appears
5. **list.insert(index, obj):** Inserts object obj into list at offset index
6. **list.pop(obj=list[-1]):** Removes and returns last object or obj from list
7. **list.remove(obj):** Removes object obj from list
8. **list.reverse():** Reverses objects of list in place
9. **list.sort([func]):** Sorts objects of list, use compare func if given

Python Tuple

A series of immutable Python objects is called a tuple. Tuples, like lists, are sequences. Tuples and lists vary in that tuples rely on parenthesis, whereas lists employ square brackets, and tuples cannot be altered. Simply putting several comma-separated items together creates a tuple. It is also possible to place these comma-separated values within parenthesis. For example:
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5);
tup3 = "a", "b", "c", "d";
Two parentheses with nothing inside them represent the empty tuple: tup1 = ();
Although there exists only one value in a tuple, a comma must be included in order to write it:
tup1 = (50,);
Tuple indices, like string indices, begin at 0 and can be concatenated, sliced, and so on.

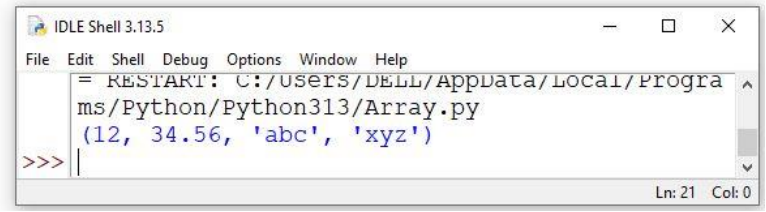
Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index.

Updating Tuples

Tuples are immutable which means the values of tuple elements cannot be updated or changed. Portions of existing tuples can be taken to create new tuples as the following example demonstrates:

```
File  Edit  Format  Run  Options  Window  Help
tup1 = (12, 34.56);
tup2= ('abc', 'xyz');
# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print (tup3);
```



Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded. To explicitly remove an entire tuple, use the del statement.

Basic Tuples Operations

Tuples respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string. In fact, tuples respond to all of the general sequence operations that are used on strings.

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Indexing, Slicing, and Matrixes in Tuples

Because tuples are sequences, indexing and slicing work the same way for tuples as they do for strings. Assuming following input:

```
L = ('spam', 'Spam', 'SPAM!')
```

Python Expression	Results	Description
L[2]	'SPAM!'	Offsets start at zero
L[-2]	'Spam'	Negative: count from the right
L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

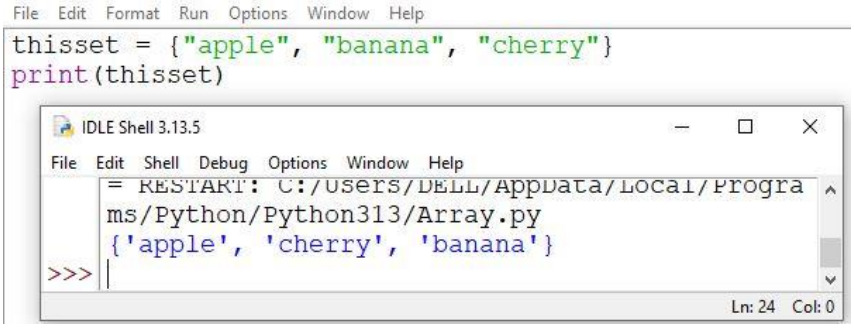
Tuple Built-in Functions

Python includes the following tuple functions:

1. **cmp(tuple1, tuple2):** Compares elements of both tuples.
2. **len(tuple):** Gives the total length of the tuple.
3. **max(tuple):** Returns item from the tuple with max value.
4. **min(tuple):** Returns item from the tuple with min value.
5. **tuple(seq)**: Converts a list into tuple.

Python Sets

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets. Sets are unordered, so the items will appear in a random order.



Access Items

Items in a set cannot be accessed by using an index because sets are unordered hence the elements in a set do not have an index. However, "for" loop can be used to iterate over the set elements or "in" keyword can be used to determine whether a given value is contained in a set. The new items can be added but they cannot be changed once a set is formed.

Add Items

The add() method is used to add one item to a set whereas, the update() method is used to add more than one item to a set.

Get the Length of a Set

To determine how many items a set has, use the len() method.

Remove Item

To remove an item in a set, use the remove(), or the discard() method. If the item to remove does not exist, remove() will raise an error. However, if the item to remove does not exist, discard() will not raise an error. The pop(), method is also used to remove an item. Sets are unordered so when using the pop() method, the item that gets removed will not be identified. The clear() method empties the set and the del keyword will delete the set completely.

Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python, dictionaries are written with curly brackets, and they have keys and values.

```
thisdict = { "brand": "Ford",
"model": "Mustang", "year": 1964}
print(thisdict)
```

Accessing Items

The items of a dictionary can be accessed by referring to its key name, inside square brackets:

```
x = thisdict["model"]
```

There is also a method called get() that will give the same result:

```
x = thisdict.get("model")
```

LABORATORY TASKS

1. Write a program to find the highest temperature from a list of daily temperature readings.
2. Write a program to calculate the total bill by summing item prices stored in a list.
3. Write a program to create a tuple storing a student's name, roll number, and GPA using different data types.
4. Write a program to add a new data item to an existing tuple, simulating an update to a user's fixed profile record.
5. Write a program to merge two dictionaries representing separate user profiles into one unified record.
6. Write a program to check if a requested data field (e.g., "email") exists in a user profile dictionary to ensure valid data access.

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 10

Objective:

To perform file handling operations in Python including reading, writing, and appending text files.

Theory:

File Handling

The key function for working with files in Python is the open() function. The open() function takes two parameters; filename, and mode. There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode:

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

To open a file for reading, it is enough to specify the name of the file: f = open("demofile.txt"). This is the same as: f = open("demofile.txt", "rt"). Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Open a File on the Server

Assume the following file located in the same folder as Python: demofile.txt
Hello! Welcome to demofile.txt This file is for testing purposes. Good Luck!
To open the file, use the built-in open() function. The open() function returns a file object, which has a read() method for reading the content of the file:

```
f = open("demofile.txt", "r")
print(f.read())
```

Read Only Parts of the File

By default the read() method returns the whole text, but it can also be specified that how many characters you want to return. For example, if you want to return the 5 characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Read Lines

One line can be returned by using the readline() method: For example, in order to read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

By calling readline() two times, you can read the two first lines:

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

By looping through the lines of the file, you can read the whole file, line by line:

```
f = open("demofile.txt", "r") for x
in f:
print(x)
```

Close Files

It is a good practice to always close the file when you are done with it.

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Write to an Existing File

To write to an existing file, a parameter must be added to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Open the file "demofile.txt" and append content to the file:

```
f = open("demofile.txt", "a")
f.write("Now the file has more content!")
f.close()
```

```
#open and read the file after the appending:
f = open("demofile.txt", "r")
print(f.read())
```

Open the file "demofile.txt" and overwrite the content:

```
f = open("demofile.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

```
#open and read the file after the appending:
f = open("demofile.txt", "r")
print(f.read())
```

Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

LABORATORY TASKS

1. Design a Python program that allows the user to add a diary entry. The program should prompt the user to enter their name and a message. Append the entry to a file called diary.txt with the current date and time.
   Format: [2025-07-16 14:30] Ali: Today I learned file handling in Python!

2. Create a Python script that reads source.txt line-by-line and writes each line to a new file called copy.txt.
3. Ask the user for a word, then read a file (data.txt) and tell how many times that word appears in the text (case-insensitive).

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 11

Objective:

To explore NumPy library for efficient data manipulation and array handling.

Theory:

Numpy

NumPy, short for "Numerical Python," provides a foundation for numerical computations in Python. It offers support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to perform efficient operations on these data structures. NumPy's powerful array-processing capabilities make it ideal for handling large datasets and performing operations such as linear algebra, statistical analysis, and Fourier transformations. Data science and machine learning applications benefit greatly from NumPy's speed and efficiency, which are crucial when working with big data.

Installing NumPy

Getting started with these libraries is straightforward with Python's package manager, pip. For others, they can be installed via pip in the command line:

Pip install numpy

This command will download and install the latest versions of NumPy, allowing to begin importing and using them in your projects. If a virtual environment (highly recommended for managing project-specific dependencies) is used, make sure the environment is activated before running the install commands. Once installed, these libraries can be imported into Python scripts as follows:

Import numpy as np
Import pandas as pd

Features of NumPy:

1. **The NumPy Array (ndarray)**

   The **ndarray**, or N-dimensional array, is NumPy's primary data structure. Unlike Python lists, ndarrays are stored in contiguous memory locations, allowing for faster access and manipulation. NumPy arrays are homogeneous, meaning all elements in the array must be of the same data type (e.g., all integers or all floats), which optimizes memory use and computational speed.

   Example: Creating a simple NumPy array

   ```
   import numpy as np
   # creating a ID NumPy array
   arr= np.array([1,2,3,4,5])
   print(arr)
   ```

   NumPy arrays can be one-dimensional, two-dimensional, or even multi-dimensional, enabling complex data structures like matrices and tensors.

2. **Array Operations and Broadcasting**

   NumPy supports element-wise operations, which makes mathematical computations easy and fast. Unlike traditional loops in Python, NumPy performs operations on entire arrays simultaneously (vectorization), leveraging efficient, low-level implementations to reduce computation time.

   Example: Element-wise operations with broadcasting

   ```
   # Create two arrays
   arr1 = np.array([1, 2, 3])
   arr2 = np.array([4, 5, 6])
   # Perform element-wise addition
   result = arr1 + arr2
   print(result)
   ```

   Broadcasting is another powerful feature in NumPy that allows for operations on arrays of different shapes. When performing operations, NumPy automatically "broadcasts" the smaller array to match the shape of the larger one, making complex operations more convenient.

   Example of broadcasting:

   ```
   # Add a scalar to each element of the array
   ```

```
arr = np.array([1, 2, 3])
print(arr + 10)  # Broadcasting adds 10 to each element
```

3. **Array Slicing and Indexing**
   Efficient slicing and indexing capabilities make it easy to access and modify specific portions of arrays. NumPy arrays allow for both integer indexing and Boolean indexing, enabling precise control over data access and manipulation.
   Example: Slicing and Boolean indexing
```
arr = np.array([1, 2, 3, 4, 5])
# Slice elements from index 1 to 3
print(arr[1:4])
# Boolean indexing to filter elements
print(arr[arr > 2])  # Prints elements greater than 2
```
4. **Mathematical and Statistical Functions**
   NumPy offers a wide range of mathematical functions, from basic operations like addition and subtraction to advanced operations like trigonometric, logarithmic, and exponential functions. It also includes statistical functions that calculate mean, median, variance, and standard deviation, allowing for efficient data analysis.
   Example: Using mathematical and statistical functions
```
arr = np.array([1, 2, 3, 4, 5])
# Calculate the mean and standard deviation
mean_val = np.mean(arr)
std_dev = np.std(arr)
print(f"Mean: {mean_val}, Standard Deviation: {std_dev}")
```
5. **Linear Algebra and Matrix Operations**
   NumPy is equipped with a suite of linear algebra functions that enable matrix multiplication, inversion, eigenvalue calculation, and more. These features make it an excellent choice for tasks involving linear algebra, which is fundamental in machine learning.
   Example: Matrix multiplication
```
# Define two matrices
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
# Perform matrix multiplication
result = np.dot(matrix1, matrix2)
print(result)
```

LABORATORY TASKS

1. Create an array of at least 8 numeric elements (e.g., measurements, voltages, or temperatures).
2. Use NumPy functions to calculate and display:

   i. Mean, Median, and Standard Deviation
   ii. Minimum and Maximum values
   iii. Sum and Product of all elements

3. Represent the following system of linear equations using NumPy arrays:

   $3x+2y=11$
   $4x+y=10$

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	

LAB SESSION 12

Objective:

To develop a mini Python project applying programming concepts to Telecommunication scenarios.

Theory:

Problem Description:

Design and implement a Python program that helps to analyze and compare the performance of simple communication links based on basic parameters like signal strength, noise, and data rate.

The purpose of this exercise is to understand how programming logic can be used to process numerical data and draw meaningful conclusions about system performance, a key skill in telecommunications engineering.

Program Requirements:

Your Python program should:

1. Accept input from the user (or use predefined values) for multiple communication links:
    - Signal Power (in watts)
    - Noise Power (in watts)
    - Data Rate (in kbps)
2. Calculate a simple quality measure for each link, such as:
    - Signal-to-Noise Ratio (SNR) = Signal Power / Noise Power
    - Performance Index = SNR × Data Rate
3. Compare the calculated performance values of different links.
4. Display:
    - The calculated SNR and performance index for each link
    - The name of the link with best performance

Requirements for Lab Session Completion:

1. Define the scenario for example, comparing 2 or 3 communication links.
2. Draw an algorithm or flowchart to show the steps from input → calculation → output.
3. Write the Python code using:
    - Variables
    - Arithmetic operations
    - Conditional statements (if, elif, else)
    - Loops (for, while)
    - Functions (optional)

4. Display all results in a clear and readable format.
5. Verify results by doing manual calculations for one case.

You need to calculate:

$$\text{SNR}=\frac{Signal\ Power}{Noise\ Power}$$

$$\text{Performance}=SNR \times DataRate$$

and determine which link performs best.

Expected Outcomes:

- Working Python program for basic telecom analysis
- Understanding of algorithmic problem-solving
- Application of loops, conditionals, and arithmetic in a practical scenario

NED University of Engineering & Technology
Department of Telecommunications Engineering

Course Code and Title: EF-101 IT Fundamentals and Applications

Laboratory Session No. _____ Date:_____

Software Use Rubric					
Criterion	**Level of Attainment**				
	Below Average (1)	**Average (2)**	**Good (3)**	**Very Good (4)**	**Excellent (5)**
Identification of software menu (syntax, components, commands, tools, layout etc.).	Can't identify software menus.	Rarely identifies software menus.	Occasionally identifies software menus.	Able to identify software menus.	Perfectly able to identify software menus.
Skills to use software (schematic, syntax, commands, tools, layout) efficiently.	Can't use software efficiently.	Rarely uses software efficiently.	Occasionally uses software efficiently.	Often uses software efficiently.	Efficiently uses software (syntax, commands, tools, layout)
Adherence to safety procedures and handling of equipment (computing unit, peripheral devices, and other equipment in lab).	Doesn't handle equipment with required care and safety.	Rarely handles equipment with required care and safety.	Occasionally handles equipment with required care and safety.	Often handles equipment with required care and safety.	Handles equipment with required care and safety.
Ability to troubleshoot software errors (detection and debugging).	Not able to troubleshoot the errors	Rarely able to troubleshoot the errors	Occasionally able to troubleshoot the errors	Often able to troubleshoot the errors	Fully able to troubleshoot the errors
Analysis and interpretation of results/outputs.	Not able to analyze and interpret results/outputs.	Rarely able to perform the analysis and interpretation.	Occasionally able to perform the analysis and interpretation.	Often able to perform the analysis and interpretation.	Perfectly able to perform the analysis and interpretation.

Weighted CLO (Score)	
Remarks	
Instructor's Signature with Date	