

# **Scripting Language - Python (4330701)**

## **Laboratory Manual**

**Semester-3**  
**Diploma in Computer Engineering**

Enrolment No	
Name	
Branch	Computer Engineering
Academic Term	2024-25 (Odd Term)
Institute	AVPTI, Rajkot



**Directorate of Technical Education**  
**Gandhinagar - Gujarat**

### DTE's Mission:

- To provide globally competitive technical education;
- Remove geographical imbalances and inconsistencies;
- Develop student friendly resources with a special focus on girls' education and support to weaker sections;
- Develop programs relevant to industry and create a vibrant pool of technical professionals.

### Institute's Vision:

*To cater skilled engineers having potential to convert global challenges into opportunities through embedded values and quality technical education.*

### Institute's Mission:

- Impart quality technical education and prepare diploma engineering professionals to meet the need of industries and society.
- Adopt latest tools and technologies for promoting systematic problem solving skills to promote innovation and entrepreneurship
- Emphasize individual development of students by inculcating moral, ethical and life skills.

### Department's Vision:

*Develop globally competent Computer Engineering Professionals to achieve excellence in an environment conducive for technical knowledge, skills, moral values and ethical values with a focus to serve the society*

### Department's Mission:

- To provide state of the art infrastructure and facilities for imparting quality education and computer engineering skills for societal benefit.
- Adopt industry-oriented curriculum with an exposure to technologies for building systems & application in computer engineering
- To provide quality technical professional as per the industry and societal needs, encourage entrepreneurship, nurture innovation and life skills in consonance with latest interdisciplinary trends.

## **Certificate**

This is to certify that Mr./Ms. ....  
Enrolment No. .... of Semester: 3<sup>rd</sup> of Diploma in  
Computer Engineering of Institute AVPTI, Rajkot (GTU Code: 602) has  
satisfactorily completed the term work in course Scripting Language -  
Python (4330701) for the Academic Year: 2024-25 (Odd Term) as prescribed  
in the GTU curriculum.

Place: Rajkot

Date: .....

**Faculty Signature**

## **Programme Outcomes (POs):**

Following programme outcomes are expected to be achieved through the practical of the course:

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis:** Identify and analyse well-defined *engineering* problems using codified standard methods.
3. **Design/development of solutions:** Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
4. **Engineering Tools, Experimentation and Testing:** Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.
5. **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.
6. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
7. **Life-long learning:** Ability to analyse individual needs and engage in updating in the context of technological changes *in field of engineering*.

## **Program Specific Outcomes (PSOs)**

- Able to apply the knowledge gained from Mathematics, Basic Sciences in general and all computer science courses in particular to identify, formulate and solve real life complex engineering problems faced in industries and society.
- The ability to employ modern computer languages, environments and platforms in creating innovative career paths in Hardware, Networking and Software Development technologies.

# **A. V. PAREKH TECHNICAL INSTITUTE, RAJKOT**

## **COMPUTER ENGINEERING DEPARTMENT**

### **ASSESSMENT RUBRICS FOR PRACTICAL/MICRO PROJECT COMPONENTS**

**SUBJECT & CODE: Scripting Language-Python (4330701)**

▪ **CONTINUOUS ASSESSMENT (25 Marks):**

- **Laboratory Work and Questionnaire Component (25 Marks):**

Component	Criteria	Marks	Assessment
<b>Laboratory Work and Questionnaire</b>	<b>Excellent</b>	<b>23-25</b>	Demonstrates exceptional proficiency in both laboratory work and questionnaire assessments, consistently applying skills and understanding effectively.
	<b>Proficient</b>	<b>18-22</b>	Shows a strong command of both laboratory work and questionnaire assessments, with minor areas for improvement.
	<b>Satisfactory</b>	<b>13-17</b>	Achieves a satisfactory level of performance in laboratory work and questionnaire assessments, with room for improvement in some areas.
	<b>Needs Improvement</b>	<b>8-12</b>	Demonstrates limited proficiency in both laboratory work and questionnaire assessments, with significant areas for improvement.
	<b>Inadequate</b>	<b>0-7</b>	Fails to meet acceptable standards in both laboratory work and questionnaire assessments; significant improvement is required.

# **A. V. PAREKH TECHNICAL INSTITUTE, RAJKOT**

## **COMPUTER ENGINEERING DEPARTMENT**

### **ASSESSMENT RUBRICS FOR PRACTICAL/MICRO PROJECT COMPONENTS**

**SUBJECT & CODE: Scripting Language-Python (4330701)**

▪ **END SEMESTER EXAMINATION (25 Marks):**

- **Viva Examination (25 Marks):**

Component	Criteria	Marks	Assessment
<b>Viva Examination</b>	<b>Excellent</b>	<b>23-25</b>	Demonstrates exceptional proficiency in the viva exam, displaying an in-depth understanding and providing comprehensive and insightful answers.
	<b>Proficient</b>	<b>18-22</b>	Displays a strong grasp of the viva exam topics, providing clear and well-reasoned answers, with minor areas for improvement.
	<b>Satisfactory</b>	<b>13-17</b>	Provides satisfactory responses during the viva exam, covering the essential topics, with room for improvement in some areas.
	<b>Needs Improvement</b>	<b>8-12</b>	Demonstrates limited understanding of the viva exam topics, providing answers that may lack clarity or depth, with significant areas for improvement.
	<b>Inadequate</b>	<b>0-7</b>	Fails to meet acceptable standards in the viva exam, providing answers that are unclear, incorrect, or lacking substance; significant improvement is required.

# **A. V. PAREKH TECHNICAL INSTITUTE, RAJKOT**

## **COMPUTER ENGINEERING DEPARTMENT**

### **ASSESSMENT RUBRICS FOR PRACTICAL/MICRO PROJECT COMPONENTS**

**SUBJECT & CODE: Scripting Language-Python (4330701)**

▪ **MICRO-PROJECT (10 Marks):**

• **Micro-Project (10 Marks):**

Component	Criteria	Marks	Assessment
<b>Micro-Project Work</b>	<b>Excellent</b>	<b>9 - 10</b>	Demonstrates exceptional proficiency in the micro project, delivering an innovative, well-executed, and thoroughly documented project with outstanding results.
	<b>Proficient</b>	<b>7 - 8</b>	Displays a strong competence in the micro project, delivering a well-executed and documented project with good results, with minor areas for improvement.
	<b>Satisfactory</b>	<b>5 - 6</b>	Successfully completes the micro project, meeting the basic requirements and delivering a project that meets expectations but may lack innovation, with room for improvement in some aspects.
	<b>Needs Improvement</b>	<b>3 - 4</b>	Demonstrates limited proficiency in the micro project, with a project that may have significant issues, incomplete components, or lacks adherence to guidelines; significant areas for improvement.
	<b>Inadequate</b>	<b>0 - 2</b>	Fails to meet acceptable standards in the micro project, delivering a project that is significantly flawed or incomplete; significant improvement is required.

## Practical Outcome - Course Outcome Matrix

<b><u>Course Outcomes (COs):</u></b>						
<p><b><u>CO1:</u> Develop programs to solve the given simple computational problems.</b></p> <p><b><u>CO2:</u> Apply control flow structures to solve the given problems.</b></p> <p><b><u>CO3:</u> Implement data structures lists, tuples, sets and dictionaries to solve the given problems.</b></p> <p><b><u>CO4:</u> Apply modular programming approach to solve given problems using user-defined functions.</b></p> <p><b><u>CO5:</u> Perform string manipulation and file operations to solve a given problem.</b></p>						
Sr. No.	Experiment/Practical Outcome	CO1	CO2	CO3	CO4	CO5
1	<p><b><u>Install and configure the Python environment.</u></b>            Install and configure the Python environment. Run basic Python commands to verify the Python environment.</p>	√	-	-	-	-
2	<p><b><u>Write a Python Program Based on Input-Output.</u></b>            Write a program to read your name, contact number, email, and birthdate and print those details on the screen.</p>	√	-	-	-	-
3	<p><b><u>Write a Python Program Based on Variables, Operators and Expressions</u></b></p> <p>i. Write a program to convert temperature from Celsius to Fahrenheit. Equation to convert Celsius to Fahrenheit:</p> $F = (9/5) * C + 32$ <p>ii. Write a program to compute the slope of a line between two points (x1, y1) and (x2, y2).</p> $Slope = \frac{y_2 - y_1}{x_2 - x_1}$ <p>iii. Write a program to calculate simple and compound interest.</p> $Simple\ Interest = \frac{P * R * T}{100}$ $Compound\ Interest = P * \left(1 + \frac{R}{100 * n}\right)^{n * T}$ <p>iv. Write a program to find a maximum of given three numbers.</p> <p>v. Write a program to calculate area and volume of Sphere.</p> $Area\ of\ Sphere = 4 \pi r^2$ $Volume\ of\ Sphere = \frac{4}{3} \pi r^3$	√	-	-	-	-



	<p>vi. Write a program that computes the real roots of a given quadratic equation (Use math library).</p> <p><math>Discriminant \Delta = b^2 - 4 a c</math></p> <p><math>Real\ Roots = \frac{-b \pm \sqrt{\Delta}}{2 a}</math></p>															
4	<p><b><u>Write a Python Program Based on Decision Making Structures</u></b></p> <p>i. A year is a Leap year if it is divisible by 4, unless it is a century year that is not divisible by 400 (1800 and 1900 are not leap years, 1600 and 2000 are leap years). Write a program that calculates whether a given year is a leap year or not.</p> <p>ii. Many companies pay time-and-a-half for any hours worked above 40 hours in a given week. Write a program to input the number of hours worked and hourly rate and calculate the total wages for the week.</p> <p>iii. Write a program to read the marks and assign a grade to a student. Grading system: A (<math>\geq 90</math>), B (80-89), C (70-79), D (60-69), E (50-59), F (<math>&lt; 50</math>). (Use if-elif-else)</p>	-	✓	-	-	-										
5	<p><b><u>Write a Python Program Based on Loops</u></b></p> <p>i. Write a program to read n numbers from users and calculate the average of those n numbers.</p> <p>ii. Write programs to print below patterns:</p> <table border="1"><tr><td style="text-align: center;">*</td><td>1</td></tr><tr><td style="text-align: center;">* *</td><td>1 2</td></tr><tr><td style="text-align: center;">* * *</td><td>1 2 3</td></tr><tr><td style="text-align: center;">* * * *</td><td>1 2 3 4</td></tr><tr><td style="text-align: center;">* * * * *</td><td>1 2 3 4 5</td></tr></table> <p>iii. Write a program to sum the following series:</p> <p><math>\frac{1}{3} + \frac{3}{5} + \frac{5}{7} + \frac{7}{9} + \frac{9}{11} + \frac{11}{13} + \dots + \frac{95}{97} + \frac{97}{99}</math></p> <p>iv. A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because <math>6 = 3 + 2 + 1</math>, the next is <math>28 = 14 + 7 + 4 + 2 + 1</math>. There are four perfect numbers that are less than 10,000. Write a program to find these four numbers.</p>	*	1	* *	1 2	* * *	1 2 3	* * * *	1 2 3 4	* * * * *	1 2 3 4 5	-	✓	-	-	-
*	1															
* *	1 2															
* * *	1 2 3															
* * * *	1 2 3 4															
* * * * *	1 2 3 4 5															

6	<p><b><u>Write a Python Program Based on List</u></b></p> <p>i. Write a program to perform the below operations on the list:</p> <ul style="list-style-type: none"> <li>● Create a list.</li> <li>● Add/Remove an item to/from a list.</li> <li>● Get the number of elements in the list.</li> <li>● Access elements of the list using the index.</li> <li>● Sort the list.</li> <li>● Reverse the list.</li> </ul> <p>ii. Write a program to read n numbers from a user and print:</p> <ul style="list-style-type: none"> <li>● Number of positive numbers.</li> <li>● Number of negative numbers.</li> <li>● Number of zeros.</li> <li>● Number of odd numbers.</li> <li>● Number of even numbers.</li> <li>● Average of all numbers.</li> </ul> <p>iii. Write a program to eliminate duplicate values in the list.</p>	-	-	√	-	-
7	<p><b><u>Write a Python Program Based on Tuples, Sets and Dictionaries</u></b></p> <p>i. Write a program to perform below operations on tuple:</p> <ul style="list-style-type: none"> <li>● Create a tuple with different data types.</li> <li>● Print tuple items.</li> <li>● Convert tuple into a list.</li> <li>● Remove data items from a list.</li> <li>● Convert list into a tuple.</li> <li>● Print tuple items.</li> </ul> <p>ii. Write a program to perform below operations on set:</p> <ul style="list-style-type: none"> <li>● Create two different sets with the data.</li> <li>● Print set items.</li> <li>● Add/remove items in/from a set.</li> <li>● Perform operations on sets: union, intersection, difference, symmetric difference, check subset of another set.</li> </ul> <p>iii. Write a program to perform below operations on dictionary:</p> <ul style="list-style-type: none"> <li>● Create a dictionary.</li> <li>● Print dictionary items.</li> <li>● Add/remove key-value pair in/from a dictionary.</li> <li>● Check whether a key exists in a dictionary.</li> <li>● Iterate through a dictionary.</li> <li>● Concatenate multiple dictionaries.</li> </ul>	-	-	√	-	-

	<p>iv. Write a program that is given a dictionary containing the average daily temperature for each day of the week, and prints all the days on which the average temperature was between 40 and 50 degrees.</p> <p>v. Write a program to repeatedly prompt the user to enter the capital of a state. Upon receiving the user's input, the program reports whether the answer is correct. Assume the states and their capitals are stored in dictionaries as key-value pairs.</p>					
8	<p><b><u>Write a Python Program Based on Functions</u></b></p> <p>i. Write a program that defines a function (shuffle) to scramble a list into a random order, like shuffling a deck of cards.</p> <p>ii. Write a program that defines a function to return a new list by eliminating the duplicate values in the list.</p> <p>iii. Write a program to print Fibonacci sequence up to n numbers using recursion. Fibonacci sequence is defined as below:  <i>Fibonacci Sequence= 1 1 2 3 5 8 13 21...</i>  <i>where <math>n^{th} \text{ term } x_n = x_{n-1} + x_{n-2}</math></i></p> <p>iv. Write a program that defines a function to determine whether input number n is prime or not. A positive whole number <math>n &gt; 2</math> is prime, if no number between 2 and <math>\sqrt{n}</math> (inclusive) evenly divides n. If n is not prime, the program should quit as soon as it finds a value that evenly divides n.</p> <p>v. Write a program that defines a function to find the GCD of two numbers using the algorithm below. The greatest common divisor (GCD) of two values can be computed using Euclid's algorithm. Starting with the values m and n, we repeatedly apply the formula: <math>n, m = m, n \% m</math> until m is 0. At that point, n is the GCD of the original m and n (Use Recursion).</p>	-	-	-	✓	-
9	<p><b><u>Write a Python Program Based on Modules</u></b></p> <p>i. Write a program that plays the popular scissor-rock-paper game. (A scissor can cut a paper, a rock can knock a scissor, and a paper can wrap a rock.) The program randomly generates a number 0, 1, or 2 representing scissor, rock, and paper. The program prompts the user to enter a number 0, 1, or 2 and displays a message indicating whether the user or the computer wins, loses, or draws.</p>	-	-	-	✓	-

	<p>ii. Write a program to print the dates of all the Sundays in a given year.</p> <p>iii. Write a program to display a graph for ReLU (Rectified Linear Unit) function. ReLU function is defined as below:</p> $y = \max(0, x)$ <p>Consider the range of <math>x</math> from -5 to 5.</p> <p>iv. Write a program to create a list representing the results of 100 students in a test, where each element represents a student's marks (between 0 to 10), and display a histogram for the result.</p> <p>v. Create a user defined module with simple functions for: addition, subtraction, multiplication, division, modulo, square, factorial. Write a program to import the module and access functions defined in the module.</p>					
10	<p><b><u>Write a Python Program Based on String Processing</u></b></p> <p>i. Write a program to check whether a given string is palindrome or not.</p> <p>ii. Write a program to read a string containing letters, each of which may be in either uppercase or lowercase, and return a tuple containing the number of vowels and consonants in the string.</p> <p>iii. Write a program to read a date in the format DD/MM/YYYY and print the same date in MM-DD-YYYY format.</p> <p>iv. Write a program that checks whether two words are anagrams. Two words are anagrams if they contain the same letters. For example, <i>silent</i> and <i>listen</i> are anagrams.</p>	-	-	-	-	✓
11	<p><b><u>Write a Python Program Based on File Handling</u></b></p> <p>i. Write a program to perform the below operations on files:</p> <ul style="list-style-type: none"> <li>● Create a text file and write a string to it.</li> <li>● Read an entire text file.</li> <li>● Read a text file line by line.</li> <li>● Write a string to a file.</li> <li>● Write a list of strings to a file.</li> <li>● Count the number of lines, words in a file.</li> </ul> <p>ii. Write a program that reads a text file and counts the occurrences of each alphabet in the file. The program should prompt the user to enter the filename.</p> <p>iii. Write a program that reads a text file and displays all the numbers found in the file.</p>	-	-	-	-	✓

	iv. Write an automated censor program that reads the text from a file and creates a new file where all of the four-letter words have been replaced by "****". You can ignore punctuation, and you may assume that no words in the file are split across multiple lines.					
--	---	--	--	--	--	--

### Progressive Assessment Sheet / Index

Sr. No	Experiment Name/Practical Outcome	CO	Date	Marks (25)	Sign
1	Install and configure the Python environment.	CO1			
2	Write a Python Program Based on Input-Output				
3	Write a Python Program Based on Variables, Operators and Expressions				
4	Write a Python Program Based on Decision Making Structures	CO2			
5	Write a Python Program Based on Write a Python Program Based on Loops				
6	Write a Python Program Based on List	CO3			
7	Write a Python Program Based on Tuples, Sets and Dictionaries				
8	Write a Python Program Based on Functions	CO4			
9	Write a Python Program Based on Modules				
10	Write a Python Program Based on String Processing	CO5			
11	Write a Python Program Based on File Handling				

**Practical No. 1: Install and configure the Python environment.**

- i. Install and configure the Python environment. Run basic Python commands to verify the Python environment.

**A. Objectives:**

A development environment is required to write, compile, run, and debug any application. This practical will help student to set up an environment for executing Python scripts using Python interpreter.

**B. Relevant Program Outcomes (POs):**

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- iii. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

**C. Competency and Practical Skills:**

This practical is expected to develop the following skills for the industry-identified competency ‘**Develop simple applications using scripting language Python.**’

- i. Installing and configuring development environment for Python.
- ii. Programming skills.
- iii. Debugging skills.

**D. Relevant Course Outcomes (COs):**

- i. Develop programs to solve the given simple computational problems.

**E. Practical Outcomes:**

- i. Install and configure Python development environment.

**F. Relevant Affective domain Outcomes (ADOs):**

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices

## G. Prerequisite Theory:

Python is an open-source, high-level, general-purpose programming language used for software development. It is one of the most popular programming languages in the world today. It is a widely used programming language in various domains, such as

- Automation
- Artificial Intelligence, Machine Learning
- Data Science
- Blockchain
- Big Data
- Server-side Web Development
- Tools Development
- Game Programming

History of Python:

- In February 1991, Van Rossum implemented initial version of Python and published the code (version 0.9.0).
- Python 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce.
- Python 2.0 added new features such as list comprehensions, garbage collection systems.
- Python 3.0 (also called "Python 3000" or "Py3K") was released on December 3, 2008. It was designed to rectify fundamental design flaws in the language. Python had accumulated new and redundant ways to program the same task, Python 3.0 had an emphasis on removing duplicative constructs and modules, in keeping with the Zen of Python: "There should be one— and preferably only one —obvious way to do it".
- Python is now being developed and maintained by a large team of volunteers and is available for free from the Python Software Foundation.

## H. Resources Required:

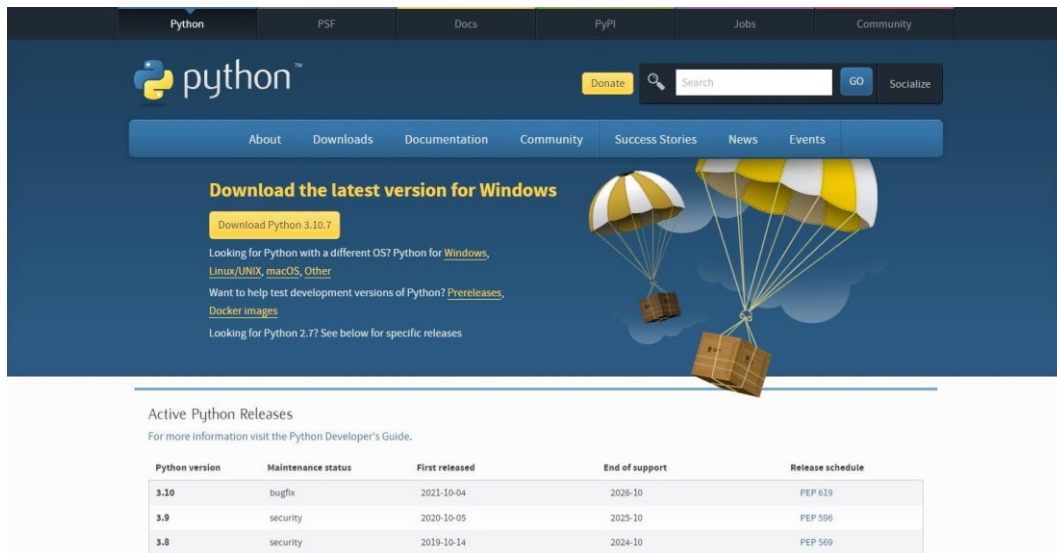
Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE



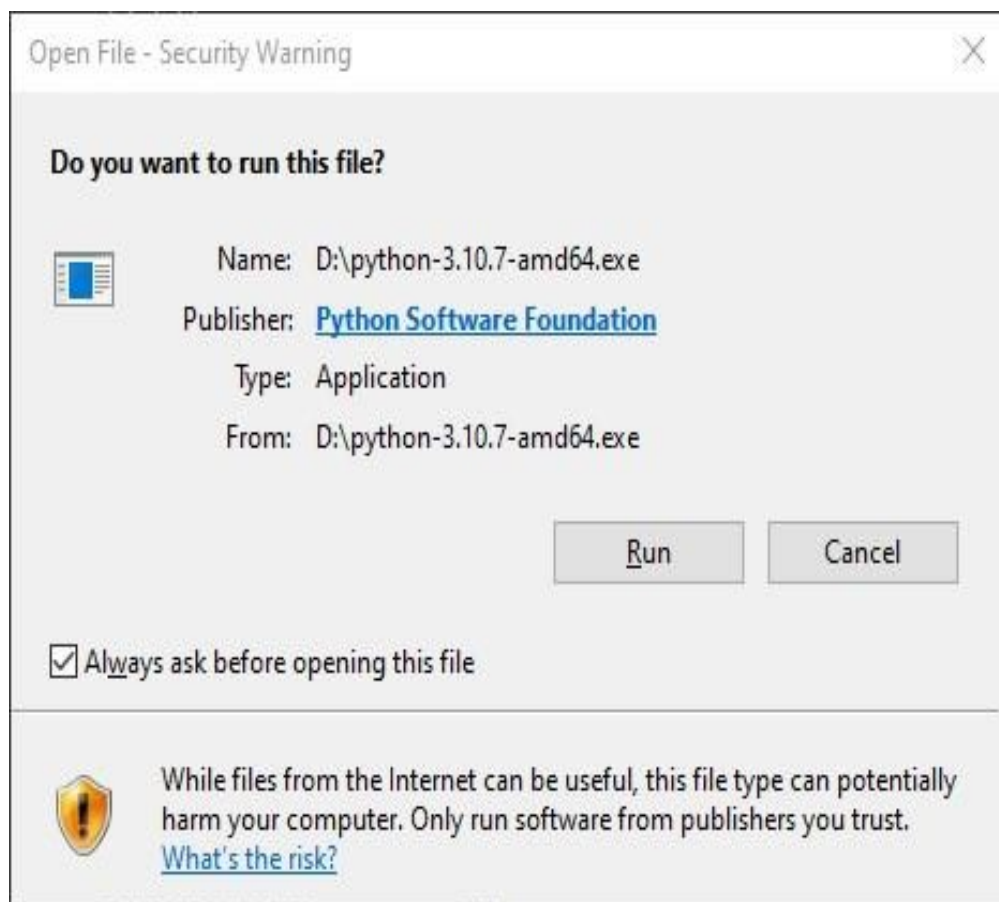
## I. Procedure:

Below steps describes installation of Python Interpreter on Windows operating system.

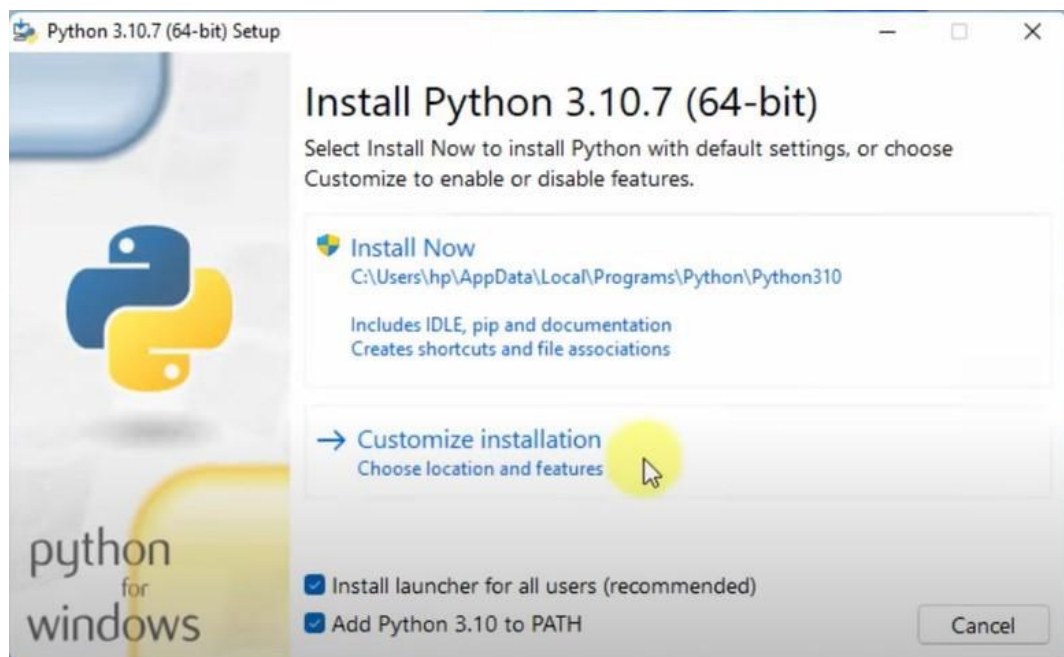
- i. Visit <https://www.python.org/downloads> and download the latest release of Python. If you want to install specific version of Python then previous versions can be downloaded from links available in the page.



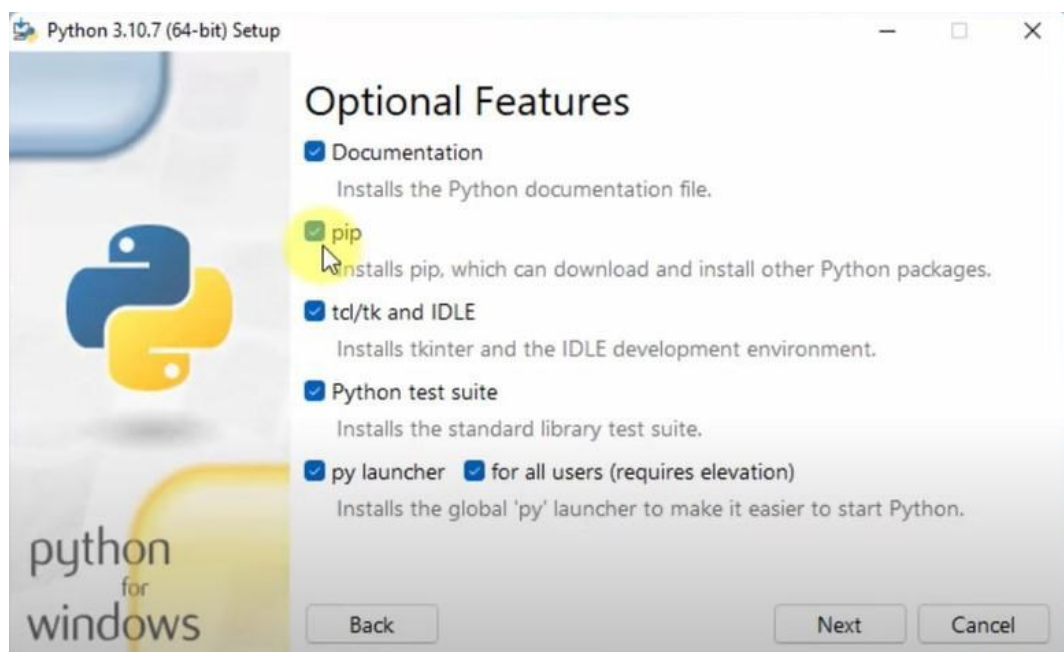
- ii. Double-click the executable file, which is downloaded. It will open **Open File - Security Warning** dialog box. Click on Run.



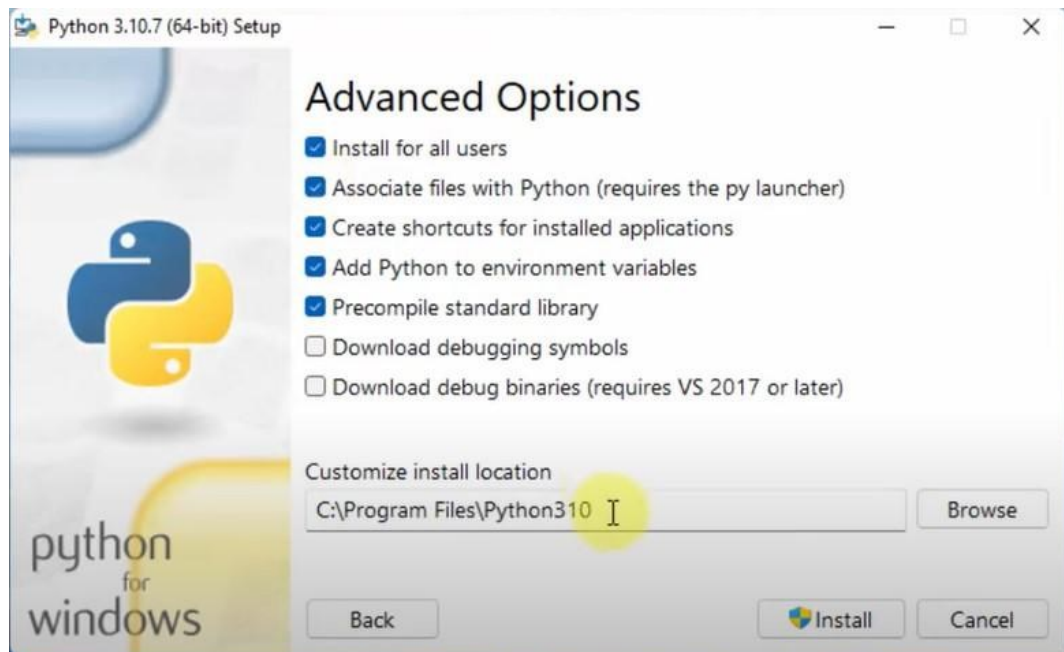
- iii. In the next **Python 3.10.7 (64-bit) Setup** dialog box, click on *Customize installation*.



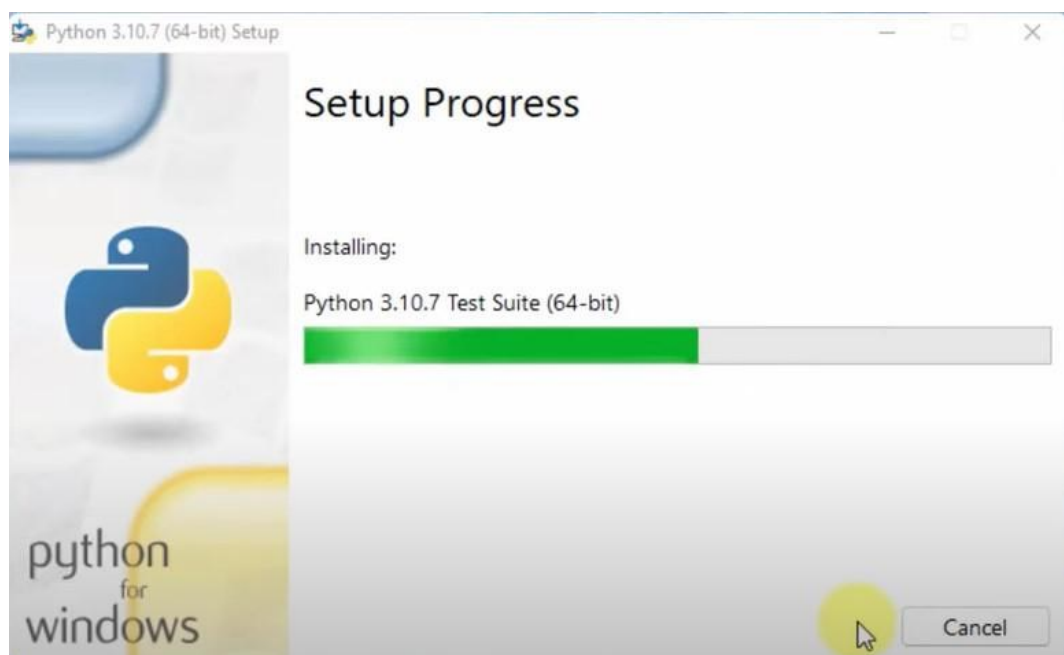
- iv. In the next dialog box, select *pip* and other optional features to be installed and click on *Next*.



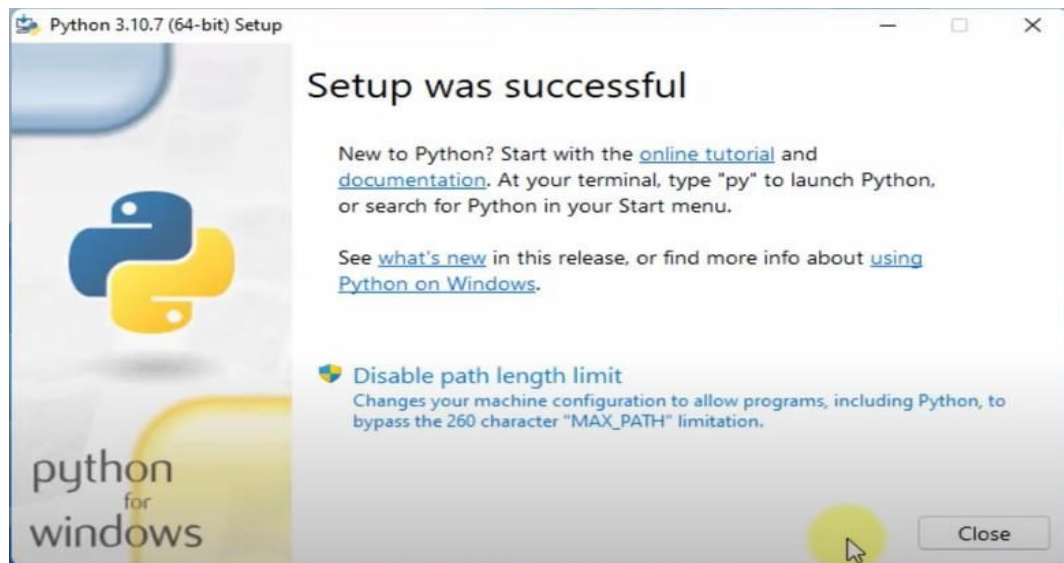
- v. In the next dialog box, select advanced options to be installed. If you wish, you can change installation path by changing Customize install location. Click on Install. Select Add Python to environment variables option to run Python from command prompt.



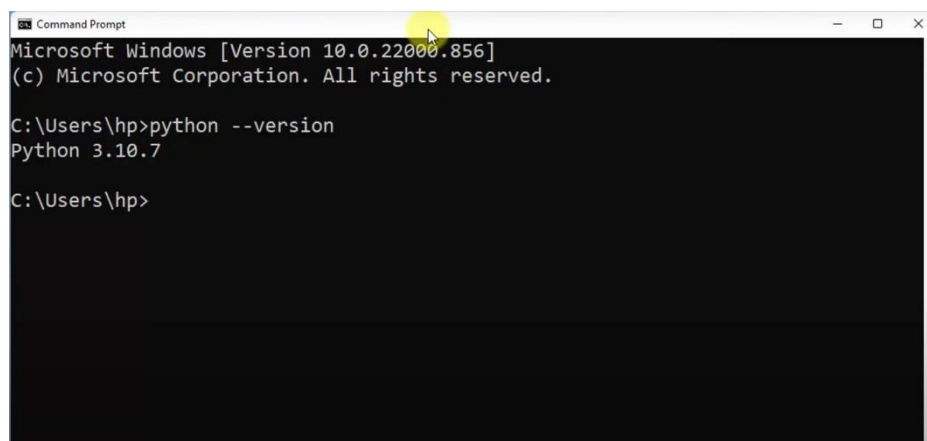
- vi. Wait for the installation to complete.



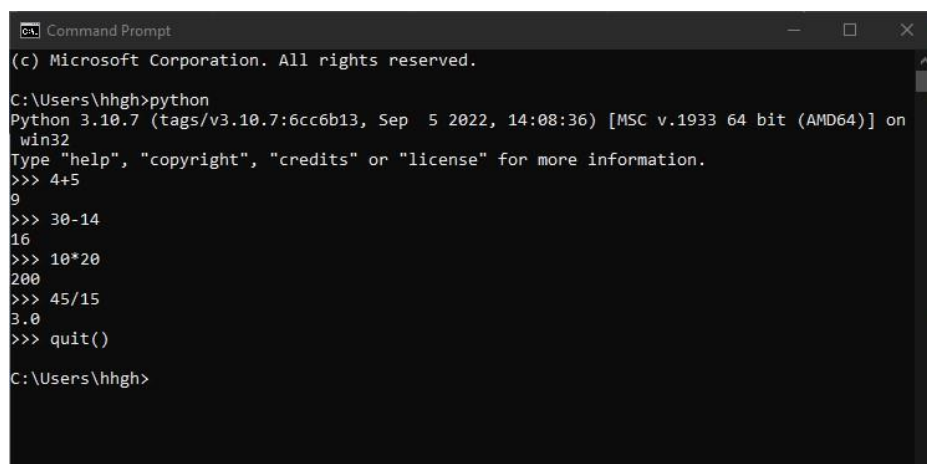
vii. After completion, you will get *Setup was successful*. Click on *Close*.



viii. To verify installation, open command prompt and run *python --version* command. It will show current version of Python interpreter.



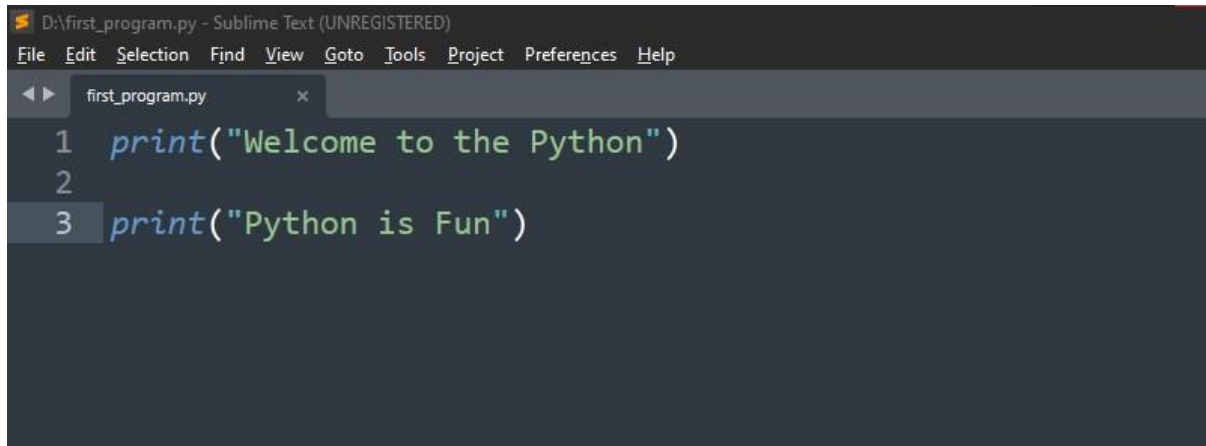
ix. In the command prompt, run *python* command. Now you can use prompt as interpreter, where you can run python commands. Run simple mathematical operations on prompt as shown in below figure. Run *quit()* to exit from command prompt for python.



To install Python Interpreter on Linux, follow Guide for Installation of Python Interpreter on Linux (<https://www.geeksforgeeks.org/how-to-install-python-on-linux>).

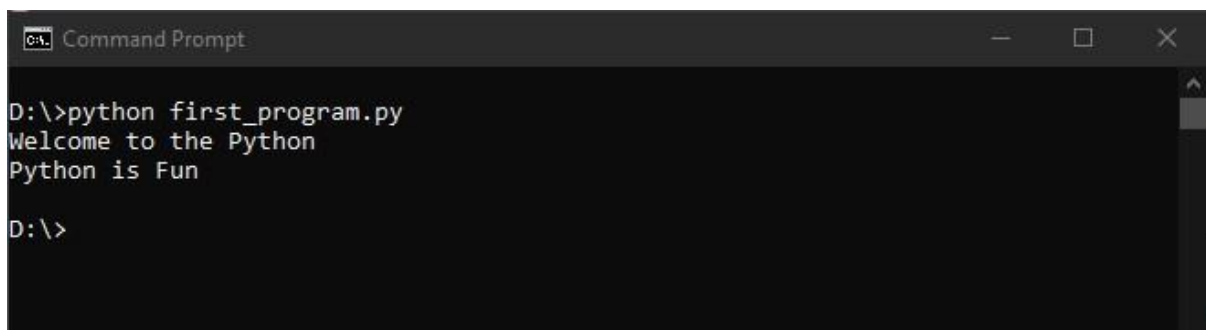
### **Writing First Program:**

Write simple print program and store it as *first\_program.py*. Run the Python script from command prompt.



```
D:\first_program.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
first_program.py x
1 print("Welcome to the Python")
2
3 print("Python is Fun")
```

### **Output:**



```
C:\> Command Prompt
D:\>python first_program.py
Welcome to the Python
Python is Fun
D:\>
```

Signature of Faculty

## Practical No. 2: Write a Python Program Based on Input-Output

- i. Write a program to read your name, contact number, email, and birthdate and print those details on the screen.

### A. Objectives:

In many applications program needs some inputs from user as well as display some information on output. This practical will help students to write Python scripts that takes inputs from user as well display on output.

### B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.

### C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency ‘**Develop simple applications using scripting language Python.**’

- i. Programming skills.
- ii. Debugging skills.

### D. Relevant Course Outcomes (COs):

- i. Develop programs to solve the given simple computational problems.

### E. Practical Outcomes:

- i. Write Python script to read data from user and display data to user.

### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipments.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices

## G. Prerequisite Theory:

- **Input Function** allows to read input from the user on the console. You can use the `input()` function to ask the user to input a value. When the input function is called it stops the program and waits for the user's input. When the user presses enter, the program resumes and returns what the user typed.

```
input(prompt=None)
```

when, the prompt is passed as a string value, it will be displayed as a prompt to the user to enter a value. Default value of prompt is None.

- **Output Function** allows user to display results on console. You can use `print()` function to display results/message.

```
print(value, ..., sep = ' ', end = '\n' , file = sys.stdout, flush = False)
```

value - value to be printed

... - it means we can pass multiple values

sep - separator used to separate multiple values

end - value of end parameter is printed at the last

file - file where output is printed

For example:

```
variable = input("Enter a value: ")
print("variable = ", variable)
print("Type of variable ", type(variable))
```

Output of the above code:

```
Enter a value: 5
variable = 5
Type of variable <class 'str'>
```

Note:

- This function first takes the input from the user and converts it into a string. The type of the returned object always will be <type 'str'>.
- If your input datatype is not string then, you need to explicitly convert input value to other data type using typecasting.

```
variable = int(input("Enter a value: "))
print("variable = ", variable)
print("Type of variable ", type(variable))
```

Output of the above code:

```
Enter a value: 7
variable = 7
Type of variable <class 'int'>
```

#### H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE



**I. Source code and Output:**

- i. Write a program to read your name, contact number, email, and birthdate and print those details on the screen.

**Source Code:**

**Output:**

**Signature of Faculty**

**Practical No. 3: Write a Python Program Based on Variables, Operators and Expressions**

- i. Write a program to convert temperature from Celsius to Fahrenheit. Equation to convert Celsius to Fahrenheit:

$$F = (9/5) * C + 32.$$

- ii. Write a program to compute the slope of a line between two points (x1, y1) and (x2, y2).

$$Slope = \frac{y_2 - y_1}{x_2 - x_1}$$

- iii. Write a program to calculate simple and compound interest.

$$Simple\ Interest = \frac{P * R * T}{100}$$

$$Compound\ Interest = P * \left(1 + \frac{R}{100 * n}\right)^{n * T}$$

- iv. Write a program to find a maximum of given three numbers.

- v. Write a program to calculate area and volume of Sphere.

$$Area\ of\ Sphere = 4 \pi r^2$$

$$Volume\ of\ Sphere = \frac{4}{3} \pi r^3$$

- vi. Write a program that computes the real roots of a given quadratic equation (Use math library).

$$Discriminant\ \Delta = b^2 - 4 a c$$

$$Real\ Roots = \frac{-b \pm \sqrt{\Delta}}{2 a}$$

**A. Objectives:**

Variables, Operators and Expressions are core part of any programming language.

- i. Variables are used to store data.
- ii. Operators are used to perform various types of operations on data.
- iii. Anything that you write in Python script is an expression.
- iv. This practical will allow students to practise writing Python scripts that use variables, operators, and expressions to solve simple problems.

## B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.

## C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

## D. Relevant Course Outcomes (COs):

- i. Develop programs to solve the given simple computational problems.

## E. Practical Outcomes:

- i. Write Python scripts to solve given problem using variables, operators and expressions.

## F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

## G. Prerequisite Theory:

### Data types:

A Data type is a set of values, and a set of operators that may be applied to those values. For example, the integer data type consists of the set of integers, and operators for addition, subtraction, multiplication, and division, among others. Python has the following data types built-in by default:

- None Type: None
- Boolean Type: bool
- Numeric Types: int, float, complex

- Text Type: str
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Binary Types: bytes, bytearray, memoryview

### Variables and Expressions:

**Variables** are identifiers that are used to reference values stored in the memory. They are called “variables”, because they may reference different values. The statement for assigning a value to a variable is called an assignment statement. In Python, the equal sign (=) is used as the assignment operator. The syntax for assignment statements is as follows:

*variable = expression*

An **expression** represents a computation involving values, variables, and operators that, taken together, evaluate to a value. A variable can also be used in both sides of the = operator. For example, consider the following code:

```
y = 5 * (6 / 2) + 3 * 2
z = 10
z = z + 5
print("Y =", y)
print("z =", z)
radius = 10
area = radius * radius * 3.14
print("Area =", area)
```

Output of the above code is as below:

```
Y = 21.0
z = 15
Area = 314.0
```

### Operators:

An operator is a symbol that represents an operation that may be performed on one or more operands (Variables, constants etc.). Python has following types of operators:

- Arithmetic operators
- Bitwise operators
- Logical operators
- Assignment operators
- Comparison operators
- Identity operators
- Membership operators

**Arithmetic Operators:**

Operator	Name	Example	Description
+	Addition	a + b	Sum of two operands
-	Subtraction	a - b	Difference of two operands
*	Multiplication	a * b	Multiply tow operands
/	Division (float)	a / b	Quotient of operands
//	Division (int/floor)	a // b	Integer Quotient of operands
%	Modulo	a % b	Reminder of operands
**	Power	a ** b	first operand raised to power second operand

**Logical Operators:**

Operator	Name	Example	Description
and	Logical AND	a and b	Logical AND operation of a and b
Or	Logical OR	a or b	Logical OR operation of a and b
not	Logical not	not a	Logical NOT operation on a

**Bitwise Operators:**

Operator	Name	Example	Description
&	Bitwise AND	a & b	Bitwise AND operation between a and b
	Bitwise OR	a   b	Bitwise OR operation between a and b
^	Bitwise XOR	a ^ b	Bitwise XOR operation between a and b
~	Bitwise NOT	~ a	Bitwise NOT operation on a
<<	Left shift	a << b	Left shift bits of a by b steps
>>	Right shift	a >> b	Right shift bits of a by b steps

**Assignment Operators:**

Operator	Name	Example	Description
=	Assign	a = b	Value of right operand is assigned to left operand
+=	Add then assign	a += b	Same as a = a + b
-=	Subtract then assign	a -= b	Same as a = a - b
*=	Multiply then assign	a *= b	Same as a = a * b
/=	Divide then assign (Quotient)	a /= b	Same as a = a / b
%=	Divide then assign (Reminder)	a %= b	Same as a = a % b
//=	Divide then assign (Int Quotient)	a //= b	Same as a = a // b
&=	Bitwise AND then assign	a &= b	Same as a = a & b
=	Bitwise OR then assign	a  = b	Same as a = a   b
^=	Bitwise NOT then assign	a ^= b	Same as a = a ^ b
>>=	Bitwise Right shift then assign	a >>= b	Same as a = a >> b
<<=	Bitwise Left shift then assign	a <<= b	Same as a = a << b

**Comparison Operators:**

Operator	Name	Example	Description
==	Equal	a == b	Returns True if a is equal to b
!=	Not equal	a != b	Returns True if a is not equal to b
<	Less than	a < b	Returns True if a is less than b
>	Greater than	a > b	Returns True if a is greater than b

<=	Less than or equal to	a <= b	Returns True if a is less than or equal to b
>=	Greater than or equal to	a >= b	Returns True if a is greater than or equal to b

**Identity Operators:**

Operator	Name	Example	Description
is	Identical	a is b	Returns True is a and b are identical
is not	not identical	a is not b	Returns True is a and b are not identical

**Ternary Operator:**

Ternary operators allows us to write conditional expressions in a single line without using if-else condition. It evaluate something based on a condition being true or false.

value\_if\_true if condition else value\_if\_false

```
a, b = 6, 9
max = a if (a > b) else b
print("Max: ", max)
```

**H. Resources Required:**

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE

**I. Source code and Output:**

- i. Write a program to convert temperature from Celsius to Fahrenheit. Equation to convert Celsius to Fahrenheit:

$$F = (9/5) * C + 32.$$

**Source Code:**

**Output:**



- ii. Write a program to compute the slope of a line between two points (x1, y1) and (x2, y2).

$$\text{Slope} = \frac{y_2 - y_1}{x_2 - x_1}$$

**Source Code:**

**Output:**

- iii. Write a program to calculate simple and compound interest.

$$\text{Simple Interest} = \frac{P * R * T}{100}$$

$$\text{Compound Interest} = P * \left(1 + \frac{R}{100 * n}\right)^{n * T}$$

**Source Code:**

**Output:**

iv. Write a program to find a maximum of given three numbers.

**Source Code:**

**Output:**

- v. Write a program to calculate area and volume of Sphere.

$$\text{Area of Sphere} = 4 \pi r^2$$

$$\text{Volume of Sphere} = \frac{4}{3} \pi r^3$$

**Source Code:**

**Output:**

- vi. Write a program that computes the real roots of a given quadratic equation (Use math library).

$$\text{Discriminant } \Delta = b^2 - 4 a c$$

$$\text{Real Roots} = \frac{-b \pm \sqrt{\Delta}}{2 a}$$

**Source Code:**

**Output:**

**Signature of Faculty**

### **Practical No. 4: Write a Python Program Based on Decision-Making Structures**

- i. A year is a Leap year if it is divisible by 4, unless it is a century year that is not divisible by 400 (1800 and 1900 are not leap years, 1600 and 2000 are leap years). Write a program that calculates whether a given year is a leap year or not.
- ii. Many companies pay time-and-a-half for any hours worked above 40 hours in a given week. Write a program to input the number of hours worked and hourly rate and calculate the total wages for the week.
- iii. Write a program to read the marks and assign a grade to a student. Grading system: A ( $\geq 90$ ), B (80-89), C (70-79), D (60-69), E (50-59), F ( $< 50$ ). (Use if-elif-else)

#### **A. Objectives:**

Conditional statements are used to perform different actions based on different conditions. This practical will help student to practice writing Python scripts using Decision making structures.

#### **B. Relevant Program Outcomes (POs):**

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

#### **C. Competency and Practical Skills:**

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

#### D. Relevant Course Outcomes (COs):

- i. Apply control flow structures to solve the given problems.

#### E. Practical Outcomes:

- i. Write Python scripts using decision making statements.

#### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

#### G. Prerequisite Theory:

##### Decision Making Statements:

Controls statements are used to control are used to control the flow of execution of program based on certain conditions. In Python, there are following decision making statements:

- if statement
- if...else statement
- nested if...else statement
- if...elif...else statement

##### if statement:

if statement allow us to run a block of code if certain condition is true. If condition is false it will not execute block of code.

```
if condition:  
    Block of code
```

##### if...else statement:

if...else executes a block of code if certain condition is true and another block of code if condition is false.

```
if condition:  
    Block of code  
else:  
    Block of code
```

##### Nested if...else statement:

There are often times when selection among more than two sets of statements (suites) is needed. For such situations, if statements can be nested, resulting in multi-way selection.

```

if condition:
    if condition:
        Block of code
    else:
        Block of code
else:
    if condition:
        Block of code
    else:
        Block of code

```

**if...elif...else statement:**

It is similar to multiple if...else statements. It executes different blocks of code based on different conditions.

```

if condition:
    Block of code
elif condition:
    Block of code
elif condition:
    Block of code
else:
    Block of code

```

**H. Resources Required:**

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE



**I. Source code and Output:**

- i. A year is a Leap year if it is divisible by 4, unless it is a century year that is not divisible by 400 (1800 and 1900 are not leap years, 1600 and 2000 are leap years). Write a program that calculates whether a given year is a leap year or not.

**Source Code:**

**Output:**

- ii. Many companies pay time-and-a-half for any hours worked above 40 hours in a given week. Write a program to input the number of hours worked and hourly rate and calculate the total wages for the week.

**Source Code:**

**Output:**

- iii. Write a program to read the marks and assign a grade to a student. Grading system:  
A ( $\geq 90$ ), B (80-89), C (70-79), D (60-69), E (50-59), F ( $< 50$ ). (Use if-elif-else)

**Source Code:**

**Output:**

**Signature of Faculty**

## Practical No. 5: Write a Python Program Based on Loops

- i. Write a program to read  $n$  numbers from users and calculate the average of those  $n$  numbers.
- ii. Write programs to print below patterns:

*	1
* *	1 2
* * *	1 2 3
* * * *	1 2 3 4
* * * * *	1 2 3 4 5

- iii. Write a program to sum the following series:

$$\frac{1}{3} + \frac{3}{5} + \frac{5}{7} + \frac{7}{9} + \frac{9}{11} + \frac{11}{13} + \dots + \frac{95}{97} + \frac{97}{99}$$

- iv. A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because  $6 = 3 + 2 + 1$ , the next is  $28 = 14 + 7 + 4 + 2 + 1$ . There are four perfect numbers that are less than 10,000. Write a program to find these four numbers.

### A. Objectives:

Loops are used to run same block of code again and again certain number of times. This practical will help student to practice writing Python scripts using Loops.

### B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

### C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

### D. Relevant Course Outcomes (COs):

- i. Apply control flow structures to solve the given problems.

### E. Practical Outcomes:

- i. Write Python scripts using loops control structures.

### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

### G. Prerequisite Theory:

Loops in Python are used to execute the same block of code a specified number of times. Python supports below loop statements:

- while loop
- for loop

while loop:

while statement is an iterative control statement that repeatedly executes a set of statements based on a provided Boolean expression (condition). All iterative control needed in a program can be achieved by use of the while statement

```
while condition:
    Block of code           # statements to execute
                           # till condition is true
```

- As long as the condition of a while statement is true, the statements within the loop are (re)executed.
- Once the condition becomes false, the iteration terminates and control continues with the first statement after the while loop.
- Note that it is possible that the first time a loop is reached, the condition may be false, and therefore the loop would never be executed.

for loop:

In Python for loop is used to iterate through each value in sequence. For loops are used for constructing definite loop.

A for statement is an iterative control statement that iterates once for each element in a specified sequence of elements.

```
For k in sequence:
    Block of code           # statements to execute for each item
                           # in the sequence
```

Variable k is referred to as a loop variable.

### **The Built-in range Function**

Python provides a built-in range function that can be used for generating a sequence of integers that a for loop can iterate over.

The values in the generated sequence include the starting value, up to but not including the ending value. For example, range(1, 11) generates the sequence [1, 2, 3, 4, 5, 6, 7, 8, 9].

```
sum = 0
for k in range(1,11):
    sum = sum + k
print("Sum:",sum)
```

### **break statement:**

- break statement in a loop is used to immediately terminate a loop. Control of the program flows to the statement immediately after the body of the loop.
- If the break statement is inside a nested loop (loop inside another loop), the break statement will terminate the inner most loop.
- Break statement makes the program simpler and easier to read.

```
n = eval(input("Enter an integer >= 2: "))
factor=2
while factor <= n:
    if n % factor == 0:
        break
    factor += 1
print("The smallest factor other than 1 for",n,"is:",factor)
```

### **continue statement:**

- continue statement in a loop is used to skip the rest of the code inside a loop for the current iteration only.
- When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop

will begin.

- Continue statements breaks out of an iteration (but not the loop), while the break statement breaks out of the loop.

```
For val in "Hello World":  
    if val=="l":  
        continue  
    print(val)
```

## H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE

**I. Source code and Output:**

- i. Write a program to read n numbers from users and calculate the average of those n numbers.

**Source Code:**

**Output:**



ii. Write programs to print below patterns:

* * * * * * * * * * * * * * *	1 1 2 1 2 3 1 2 3 4 1 2 3 4 5
---	---

**Source Code:**

**Output:**

iii. Write a program to sum the following series:

$$\frac{1}{3} + \frac{3}{5} + \frac{5}{7} + \frac{7}{9} + \frac{9}{11} + \frac{11}{13} + \dots + \frac{95}{97} + \frac{97}{99}$$

**Source Code:**

**Output:**

- iv. A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because  $6 = 3 + 2 + 1$ , the next is  $28 = 14 + 7 + 4 + 2 + 1$ . There are four perfect numbers that are less than 10,000. Write a program to find these four numbers.

**Source Code:**

**Output:**

**Signature of Faculty**

## Practical No. 6: Write a Python Program Based on Lists

- i. Write a program to perform the below operations on the list:
  - Create a list.
  - Add/Remove an item to/from a list.
  - Get the number of elements in the list.
  - Access elements of the list using the index.
  - Sort the list.
  - Reverse the list.
- ii. Write a program to read n numbers from a user and print:
  - Number of positive numbers.
  - Number of negative numbers.
  - Number of zeros.
  - Number of odd numbers.
  - Number of even numbers.
  - Average of all numbers.
- iii. Write a program to eliminate duplicate values in the list.

### A. Objectives:

In Python, list data structure allows user to store multiple elements of similar data type under a single variable. List provide below advantages:

- No need to use multiple variables to store different data.
- Easy to traverse data in list using loops.
- Easy to sort data stored in list

This practical will help student to practice writing Python scripts using list.

### B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

### **C. Competency and Practical Skills:**

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

### **D. Relevant Course Outcomes (COs):**

- i. Implement data structures lists, tuples, sets and dictionaries to solve the given problems.

### **E. Practical Outcomes:**

- i. Write Python scripts to store data in list to solve given problem.

### **F. Relevant Affective domain Outcomes (ADOs):**

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

### **G. Prerequisite Theory:**

- A list is a linear data structure, meaning that its elements have a linear ordering. That is, there is a first element, a second element, and so on.
- List in Python is mutable. It means we can add items to list, remove items from list or modify items in list at any point of time.
- List can store elements of same type as well as of different types.
- Each item in the list is identified by its index value. In Python, list index starts from 0.
- Number of elements in the list is called Length of list. As we add/remove elements from the list, length of the list changes. If there are n elements in the list then index in the list starts from 0 and ends at n-1.

### **Creating List:**

```
# Create an empty list l1
l1 = []
print("L1:", l1)
# Create a list l2 with 10 integers
l2 = [55, 89, 98, 68, 21, 7, 132, 76, 49, 36]
print("L2:", l2)
# Create a list l3 containing 7 strings as its items
l3 = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple',
      'banana']
print("L3:", l3)
# Create a list l4 containing mix data types as its items
l4 = ['orange', 99, 34.50, 'kiwi', 10, 'apple']
print("L4:" l4)
```

### **Length of a List:**

`len()` function returns the number of elements in the list.

```
l2 = [55, 89, 98, 68, 21, 7, 132, 76, 49, 36]
print(len(l2))
```

Accessing list using Index:

An element in a list can be accessed through the index operator ( `[]` ) and its index value.

```
mylist = [5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45,
          99.993, 11123]
print("Item at index", 0, ":", mylist[0])
print("Item at index", 1, ":", mylist[1])
print("Item at index", 2, ":", mylist[2])
```

List Slicing:

The index operator allows you to select an element at the specified index. The slicing operator returns a slice of the list using the syntax `list[start : end : stepsize]`.

```
l = [55, 89, 98, 68, 21, 7, 132, 76, 49, 36]
print("l[2:4]:", l[2:4])      # items from index 2 to 4-1=3
print("l[:5]:", l[:5])       # items from beginning to index 5-1=4
print("l[7:]:", l[7:])        # items from index 7 to last item
print("l[:]:", l[:])          # all the items in the list
print("l[4:-3]:", l[4:-3])    # items from 4 to 10-3-1=6
print("l[-4:-2]:", l[-4:-2])  # items from 10-4=6 to 10-2-1=7
```

Accessing list using while loop:

Using for loop:

```
mylist = [5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 99.993, 11123]

print("\nAccessing List using for Loop:")
for item in mylist:
    print(item)
```

Using while loop:

```
mylist = [5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 99.993, 11123]

print("Accessing List using while Loop:")
i = 0
while(i < len(mylist)):
    print(i, ":", mylist[i])
    i = i + 1
```

## H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE

## I. Source code and Output:

- i. Write a program to perform the below operations on the list:
  - Create a list.
  - Add/Remove an item to/from a list.
  - Get the number of elements in the list.
  - Access elements of the list using the index.
  - Sort the list.
  - Reverse the list.

**Source Code:**

**Output:**



ii. Write a program to read n numbers from a user and print:

- Number of positive numbers.
- Number of negative numbers.
- Number of zeros.
- Number of odd numbers.
- Number of even numbers.
- Average of all numbers.

**Source Code:**

**Output:**

- iii. Write a program to eliminate duplicate values in the list.

**Source Code:**

**Output:**

**Signature of Faculty**

**Practical No. 7: Write a Python Program Based on Tuples, Sets and Dictionaries**

- i. Write a program to perform below operations on tuple:
  - Create a tuple with different data types.
  - Print tuple items.
  - Convert tuple into a list.
  - Remove data items from a list.
  - Convert list into a tuple.
  - Print tuple items.
- ii. Write a program to perform below operations on set:
  - Create two different sets with the data.
  - Print set items.
  - Add/remove items in/from a set.
  - Perform operations on sets: union, intersection, difference, symmetric difference, check subset of another set.
- iii. Write a program to perform below operations on dictionary:
  - Create a dictionary.
  - Print dictionary items.
  - Add/remove key-value pair in/from a dictionary.
  - Check whether a key exists in a dictionary.
  - Iterate through a dictionary.
  - Concatenate multiple dictionaries.
- iv. Write a program that is given a dictionary containing the average daily temperature for each day of the week, and prints all the days on which the average temperature was between 40 and 50 degrees.
- v. Write a program to repeatedly prompt the user to enter the capital of a state. Upon receiving the user's input, the program reports whether the answer is correct. Assume the states and their capitals are stored in dictionaries as key-value pairs

**A. Objectives:**

Other than List Python provides three inbuilt data types: Tuple, Set and Dictionary. This practical will help student to practice writing Python scripts using these data structures.

**B. Relevant Program Outcomes (POs):**

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.

- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

### C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

### D. Relevant Course Outcomes (COs):

- i. Implement data structures lists, tuples, sets and dictionaries to solve the given problems.

### E. Practical Outcomes:

- i. Write Python scripts to perform operation on Tuples, Sets and Dictionaries.

### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

### G. Prerequisite Theory:

#### Tuples:

- Tuples are like lists, but their elements are fixed; that is, once a tuple is created, you cannot add new elements, delete elements, replace elements, or reorder the elements in the tuple. Hence, Tuples are Immutable objects.
- If the contents of a list in your application shouldn't change, you can use a tuple to prevent elements from being added, deleted, or replaced accidentally.

- Furthermore, tuples are more efficient than lists due to Python's implementations.

```
# create an empty tuple
t1 = ()
print("t1:", t1)

# create a tuple with 4 elements
t2 = (1, 3, 5, 7)
print("t2:", t2)

# create a tuple from list
t3 = tuple([1, 2, 3, 4,5])
print("t3:", t3)

# create a tuple from list variable
l = [2, 4, 6, 8, 10]
t4 = tuple(l)
print("t4:", t4)
```

### Sets:

- Sets are like lists in that you use them for storing a collection of elements.
- Unlike lists, however, the elements in a set are non-duplicates and are not placed in any particular order.
- If your application does not care about the order of the elements, using a set to store elements is more efficient than using lists due to Python's implementations.

```
# create an empty set
s1 = set()
print("s1:", s1)

# create a set with three elements
s2 = {1, 3, 5}
print("s2:", s2)

# create a set from list, removes duplicate items
s3 = set([1, 3, 5, 7, 9, 7, 1])
print("s3:", s3)
```

## Dictionaries:

- A dictionary is a container object that stores a collection of key/value pairs.
- It enables fast retrieval, deletion, and updating of the value by using the key.
- A dictionary is a collection that stores the values along with the keys. The keys are like an index operator.
- In a list, the indexes are integers. In a dictionary, the key must be a hash table object.
- A dictionary cannot contain duplicate keys. Each key maps to one value.

```
# Create an empty dictionary
dict1 = {}
print("Dict1:", dict1)

# create a dictionary with 2 key:value pairs
dict2 = {"111-34-3434": "John", "132-56-6290": "Peter"}
print("Dict2:", dict2)
```

### Add, modify, retrieve and delete value from dictionary.

```
# define customer dictionary with 3 key:value pairs
customer = { "name": "John", "custid": "1234", "country":
"USA"}
print("customer dictionary:", customer)

# add new key:value pair to customer dictionary
customer["contactno"] = "123456789"
print("customer dictionary after adding contactno:", customer)

# Modify value for the key: country
customer["country"] = "UK"
print("customer dictionary after changing country:", customer)

print("Name of the customer:", customer["name"], ", Customer
id:", customer["custid"])

del customer['country']
print("customer dictionary after deleting country:", customer)
```

### Looping through items in dictionary

```
customer = {'name': 'John', 'custid': '1234', 'country': 'UK',  
'contactno': '123456789'}  
  
for key in customer:  
    print(key, ":", customer[key])
```

### H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE

**I. Source code:**

i. Write a program to perform below operations on tuple:

- Create a tuple with different data types.
- Print tuple items.
- Convert tuple into a list.
- Remove data items from a list.
- Convert list into a tuple.
- Print tuple items.

**Source Code:**

**Output:**



ii. Write a program to perform below operations on set:

- Create two different sets with the data.
- Print set items.
- Add/remove items in/from a set.
- Perform operations on sets: union, intersection, difference, symmetric difference, check subset of another set.

**Source Code:**

**Output:**

iii. Write a program to perform below operations on dictionary:

- Create a dictionary.
- Print dictionary items.
- Add/remove key-value pair in/from a dictionary.
- Check whether a key exists in a dictionary.
- Iterate through a dictionary.
- Concatenate multiple dictionaries.

**Source Code:**

**Output:**

- iv. Write a program that is given a dictionary containing the average daily temperature for each day of the week, and prints all the days on which the average temperature was between 40 and 50 degrees.

**Source Code:**

**Output:**

- v. Write a program to repeatedly prompt the user to enter the capital of a state. Upon receiving the user's input, the program reports whether the answer is correct. Assume the states and their capitals are stored in dictionaries as key-value pairs

**Source Code:**

**Output:**

**Signature of Faculty**

### Practical No. 8: Write a Python Program Based on Function

- i. Write a program that defines a function (shuffle) to scramble a list into a random order, like shuffling a deck of cards.
- ii. Write a program that defines a function to return a new list by eliminating the duplicate values in the list.
- iii. Write a program to print Fibonacci sequence up to n numbers using recursion. Fibonacci sequence is defined as below:

*Fibonacci Sequence*= 1 1 2 3 5 8 13 21...

$$\text{where } n^{\text{th}} \text{ term } x_n = x_{n-1} + x_{n-2}$$

- iv. Write a program that defines a function to determine whether input number  $n$  is prime or not. A positive whole number  $n > 2$  is prime, if no number between 2 and  $\sqrt{n}$  (inclusive) evenly divides  $n$ . If  $n$  is not prime, the program should quit as soon as it finds a value that evenly divides  $n$ .
- v. Write a program that defines a function to find the GCD of two numbers using the algorithm below. The greatest common divisor (GCD) of two values can be computed using Euclid's algorithm. Starting with the values  $m$  and  $n$ , we repeatedly apply the formula:  $n, m = m, n \% m$  until  $m$  is 0. At that point,  $n$  is the GCD of the original  $m$  and  $n$  (Use Recursion).

#### A. Objectives:

A function is a block of reusable code that is used to perform a specific action. Functions provide below advantages:

- Reduce duplication of the code.
- Modularisation of the code.
- Improve clarity of the code.
- Information hiding.

This practical will help student to practice writing Python scripts using user defined functions and in-built functions.

#### B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

### C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

### D. Relevant Course Outcomes (COs):

- i. Apply modular programming approach to solve the given problems using user-defined functions.

### E. Practical Outcomes:

- i. Write Python scripts to develop user-defined functions to solve given problem.

### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipments.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

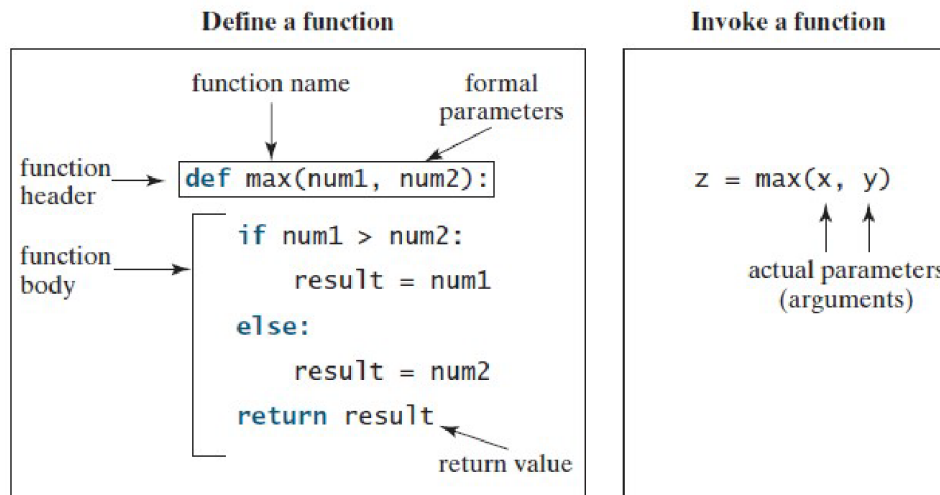
### G. Prerequisite Theory:

- Functions can be used to define reusable code and organize and simplify code.
- A Function is a named group of instructions performing some task. A Function can be invoked (called) as many times as needed in a given program.

#### Defining a Function

```
def functionName(List of parameters):  
    # Function body
```

A function definition consists of the function's header and body:



- The function header begins with the `def` keyword, followed by the function's name and parameters (surrounded by parentheses), and ends with a colon.
- The variables in the function header are known as formal parameters or simply parameters.
- A parameter is like a placeholder: When a function is invoked, you pass a value to the parameter. This value is referred to as an actual parameter or argument.
- Parameters are optional: a function may or may not have any parameters.
  - For example, the `random.random()` function has no parameters.
  - In above example `max(num1, num2)` function has 2 parameters: `num1` and `num2`.
- The function body contains a collection of statements that define what the function does.
- Some functions return a value, while other functions perform desired operations without returning a value. If a function returns a value, it is called a value-returning function.
- For example, the function body of the `max` function uses an `if` statement to determine which number is larger and return the value of that number.
- A return statement using the keyword `return` is required for a value-returning function to return a result.
- The function terminates when a return statement is executed.

### Calling a Function

Calling a function executes the code in the function.

```
larger = max(3, 4)
```

calls `max(3, 4)` and assigns the result of the function to the variable `larger`.

### Passing Parameter to a function

When you invoke a function with arguments, each argument's reference is passed by value to the parameter in the function.

```
def main():
    x = 1
    print("Before the call x:",x)
    increment(x)
    print("After the call x:",x)

def increment(n):
    print("\tInside the function before increment n:",n)
    n = n + 1
    print("\tInside the function after the increment n:",n)

main()
# Call the main function
```

#### H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE



**I. Source code and Output:**

- i. Write a program that defines a function (shuffle) to scramble a list into a random order, like shuffling a deck of cards.

**Source Code:**

**Output:**

- ii. Write a program that defines a function to return a new list by eliminating the duplicate values in the list.

**Source Code:**

**Output:**

- iii. Write a program to print Fibonacci sequence up to n numbers using recursion.  
Fibonacci sequence is defined as below:

*Fibonacci Sequence*= 1 1 2 3 5 8 13 21...

$$\text{where } n^{\text{th}} \text{ term } x_n = x_{n-1} + x_{n-2}$$

**Source Code:**

**Output:**

- iv. Write a program that defines a function to determine whether input number  $n$  is prime or not. A positive whole number  $n > 2$  is prime, if no number between 2 and  $\sqrt{n}$  (inclusive) evenly divides  $n$ . If  $n$  is not prime, the program should quit as soon as it finds a value that evenly divides  $n$ .

**Source Code:**

**Output:**

- v. Write a program that defines a function to find the GCD of two numbers using the algorithm below. The greatest common divisor (GCD) of two values can be computed using Euclid's algorithm. Starting with the values m and n, we repeatedly apply the formula:  $n, m = m, n \% m$  until m is 0. At that point, n is the GCD of the original m and n (Use Recursion).

**Source Code:**

**Output:**

**Signature of Faculty**

### Practical No. 9: Write a Python Program Based on Modules

- i. Write a program that plays the popular scissor-rock-paper game. (A scissor can cut a paper, a rock can knock a scissor, and a paper can wrap a rock.) The program randomly generates a number 0, 1, or 2 representing scissor, rock, and paper. The program prompts the user to enter a number 0, 1, or 2 and displays a message indicating whether the user or the computer wins, loses, or draws.
- ii. Write a program to print the dates of all the Sundays in a given year.
- iii. Write a program to display a graph for ReLU (Rectified Linear Unit) function. ReLU function is defined as below:

$$y = \max(0, x)$$

Consider the range of  $x$  from -5 to 5.

- iv. Write a program to create a list representing the results of 100 students in a test, where each element represents a student's marks (between 0 to 10), and display a histogram for the result.
- v. Create a user defined module with simple functions for: addition, subtraction, multiplication, division, modulo, square, factorial. Write a program to import the module and access functions defined in the module.

#### A. Objectives:

Python provides various in-built modules to be used in program. Use can also create user-defined modules as per the application requirement. This practical will help students to practice use of in-built modules and use-defined modules.

#### B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

### C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills, Debugging skills.

### D. Relevant Course Outcomes (COs):

- i. Apply modular programming approach to solve the given problems using user-defined functions.

### E. Practical Outcomes:

- i. Write Python scripts to use in-built modules to solve given problem.

### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipment's.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

### G. Prerequisite Theory:

Python module is a file containing a set of functions that you can include in your application. A module can define functions, classes, and variables. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

User can define their own modules and import them in the source code or they can import already available modules.

#### Create a module:

Create a simple calc.py in which we define two functions, one add and another subtract.

```
# A simple module, calc.py
def add(x, y):
    return (x+y)
def subtract(x, y):
    return (x-y)
```

#### Import a module:

We can import the functions, and classes defined in a module to another module using the import statement in some other Python source file.

```
# importing module calc.py
import calc
print(calc.add(10, 2))
```

Python's from statement lets you import specific attributes from a module without importing the module as a whole.

```
# importing sqrt() and factorial from the module math
from math import sqrt, factorial
# if we simply do "import math", then math.sqrt(16) and
# math.factorial() are required.
print(sqrt(16))
print(factorial(6))
```

Some of the Python modules:

- **random:** *random* module is an in-built module of Python that is used to generate random numbers
- **math:** *math* module provides set of methods and constants for mathematical tasks.
- **datetime:** *datetime* module provides classes and functions to work with date and time.
- **matplotlib:** *matplotlib* is a comprehensive library for creating static, animated, and interactive visualizations in Python.

## H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE



**I. Source code and Output:**

- i. Write a program that plays the popular scissor-rock-paper game. (A scissor can cut a paper, a rock can knock a scissor, and a paper can wrap a rock.) The program randomly generates a number 0, 1, or 2 representing scissor, rock, and paper. The program prompts the user to enter a number 0, 1, or 2 and displays a message indicating whether the user or the computer wins, loses, or draws.

**Source Code:**

**Output:**

- ii. Write a program to print the dates of all the Sundays in a given year.

**Source Code:**

**Output:**

- iii. Write a program to display a graph for ReLU (Rectified Linear Unit) function. ReLU function is defined as below:

$$y = \max(0, x)$$

Consider the range of x from -5 to 5.

**Source Code:**

**Output:**

- iv. Write a program to create a list representing the results of 100 students in a test, where each element represents a student's marks (between 0 to 10), and display a histogram for the result.

**Source Code:**

**Output:**

- v. Create a user defined module with simple functions for: addition, subtraction, multiplication, division, modulo, square, factorial. Write a program to import the module and access functions defined in the module.

**Source Code:**

**Output:**

**Signature of Faculty**

### **Practical No. 10: Write a Python Program Based on String Processing**

- i. Write a program to check whether a given string is palindrome or not.
- ii. Write a program to read a string containing letters, each of which may be in either uppercase or lowercase, and return a tuple containing the number of vowels and consonants in the string.
- iii. Write a program to read a date in the format DD/MM/YYYY and print the same date in MM-DD-YYYY format.
- iv. Write a program that checks whether two words are anagrams, Two words are anagrams if they contain the same letters. For example, silent and listen are anagrams.

#### **A. Objectives:**

This practical will help students to practise string processing using various string processing functions.

#### **B. Relevant Program Outcomes (POs):**

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

#### **C. Competency and Practical Skills:**

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

#### **D. Relevant Course Outcomes (COs):**

- i. Perform string manipulation and file operations to solve the given problems.

#### **E. Practical Outcomes:**

- i. Write Python scripts using string processing functions.

#### **F. Relevant Affective domain Outcomes (ADOs):**

- i. Maintain tools and equipments.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

#### **G. Prerequisite Theory:**

Python has a set of built-in methods that you can use on strings.

- `string_name.capitalize()`
  - Converts the first character of the string to a capital (uppercase) letter, while making all other characters in the string lowercase letters.
- `string_name.count(substring, start=..., end=...)`
  - Returns the number of occurrences of a substring in the given string.
- `string_name.find(sub, start, end)`
  - Returns the lowest index or first occurrence of the substring if it is found in a given string. If it is not found, then it returns -1.
- `string_name.isalnum()`
  - Checks whether all the characters in a given string are either alphabet or numeric (alphanumeric) characters.
- `string_name.isalpha()`
  - Check whether all characters in the String are an alphabet.
- `string_name.isdecimal()`
  - Returns true if all characters in a string are decimal, else it returns False.
- `string_name.isdigit()`
  - Returns "True" if all characters in the string are digits, Otherwise, It returns "False".
- `string_name.isidentifier()`
  - Checks whether a string is a valid identifier or not. The method returns True if the string is a valid identifier, else returns False.
- `string_name.islower()`
  - Returns True if all alphabets in a string are lowercase alphabets. If the string contains at least one uppercase alphabet, it returns False.

- `string_name.isupper()`
  - Returns True if all alphabets in a string are uppercase alphabets. If the string contains at least one lowercase alphabet, it returns False.
- `string_name.isspace()`
  - Returns “True” if all characters in the string are whitespace characters, Otherwise, It returns “False”.
- `string_name.isnumeric()`
  - Returns “True” if all characters in the string are numeric characters, otherwise returns “False”.
- `string_name.join(iterable)`
  - Joins elements of the sequence separated by a string separator. This function joins elements of a sequence and makes it a string.
- `string_name.lower()`
  - Converts all uppercase characters in a string into lowercase characters and returns it.
- `String_name.upper()`
  - Converts all lowercase characters in a string into uppercase characters and returns it.
- `string_name.swapcase()`
  - converts all uppercase characters to lowercase and vice versa of the given string and returns it.
- `string_name.rstrip([chars])`
  - Returns a copy of the string with trailing characters removed (based on the string argument passed). If no argument is passed, it removes trailing spaces.



## H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE

## I. Source code and Output:

- i. Write a program to check whether a given string is palindrome or not.

**Source Code:**

**Output:**

- ii. Write a program to read a string containing letters, each of which may be in either uppercase or lowercase, and return a tuple containing the number of vowels and consonants in the string.

**Source Code:**

**Output:**

- iii. Write a program to read a date in the format DD/MM/YYYY and print the same date in MM-DD-YYYY format.

**Source Code:**

**Output:**

- iv. Write a program that checks whether two words are anagrams, two words are anagrams if they contain the same letters. For example, silent and listen are anagrams.

**Source Code:**

**Signature of Faculty**

## Practical No. 11: Write a Python Program Based on File Handling

- i. Write a program to perform the below operations on files:
  - Create a text file and write a string to it.
  - Read an entire text file.
  - Read a text file line by line.
  - Write a string to a file.
  - Write a list of strings to a file.
  - Count the number of lines, words in a file.
- ii. Write a program that reads a text file and counts the occurrences of each alphabet in the file. The program should prompt the user to enter the filename.
- iii. Write a program that reads a text file and displays all the numbers found in the file.
- iv. Write an automated censor program that reads the text from a file and creates a new file where all of the four-letter words have been replaced by "\*\*\*\*". You can ignore punctuation, and you may assume that no words in the file are split across multiple lines.

### A. Objectives:

This practical will help students to perform file operations using file handling functions.

### B. Relevant Program Outcomes (POs):

- i. **Basic and Discipline specific knowledge (PO1):** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the Computer Engineering problems.
- ii. **Problem analysis (PO2):** Identify and analyse well-defined Computer Engineering problems using codified standard methods.
- iii. **Design/development of solutions (PO3):** Design solutions for Computer Engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
- iv. **Engineering Tools, Experimentation and Testing (PO4):** Apply modern Computer Engineering tools and appropriate technique to conduct standard tests and measurements.
- v. **Project Management (PO6):** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- vi. **Life-long learning (PO7):** Ability to analyse individual needs and engage in updating in the context of technological changes in field of engineering.

### C. Competency and Practical Skills:

This practical is expected to develop the following skills for the industry-identified competency '**Develop simple applications using scripting language Python.**'

- i. Problem analysis skills.
- ii. Programming skills.
- iii. Debugging skills.

### D. Relevant Course Outcomes (COs):

- i. Perform string manipulation and file operations to solve the given problems.

### E. Practical Outcomes:

- i. Write Python scripts using file processing functions.

### F. Relevant Affective domain Outcomes (ADOs):

- i. Maintain tools and equipments.
- ii. Follow Coding standards and practices.
- iii. Follow ethical practices.

### G. Prerequisite Theory:

Python has several functions for creating, reading, updating, and deleting files.

#### Opening a file:

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:

```
f = open("demofile.txt", "rt")
```

#### Read a file:

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

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

You can return one line by using the `readline()` method:

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

### Write to a file:

You can use `write()` method to write to a file:

```
f = open("demofile2.txt", "a")  
f.write("Now the file has more content!")  
f.close()
```

### 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()
```

## H. Resources Required:

Sr. No	Instrument /Components	Configuration/Specification
1.	Computer System	Processor: x86 64-bit CPU (Intel / AMD architecture). RAM: 4GB Operating System: Modern Operating System: Windows 7 or 10 Mac OS X 10.11 or higher, 64-bit Linux: RHEL 6/7, 64-bit
2.	Python Interpreter	Python Version: 3 or higher
3.	Text Editor	Editor: IDLE

## I. Source code and Output:

i. Write a program to perform the below operations on files:

- Create a text file and write a string to it.
- Read an entire text file.
- Read a text file line by line.
- Write a string to a file.
- Write a list of strings to a file.
- Count the number of lines, words in a file.

**Source Code:**

**Output:**



- ii. Write a program that reads a text file and counts the occurrences of each alphabet in the file. The program should prompt the user to enter the filename.

**Source Code:**

**Output:**

- iii. Write a program that reads a text file and displays all the numbers found in the file.

**Source Code:**

**Output:**

- iv. Write an automated censor program that reads the text from a file and creates a new file where all of the four-letter words have been replaced by “\*\*\*\*”. You can ignore punctuation, and you may assume that no words in the file are split across multiple lines.

**Source Code:**

**Output:**

**Signature of Faculty**