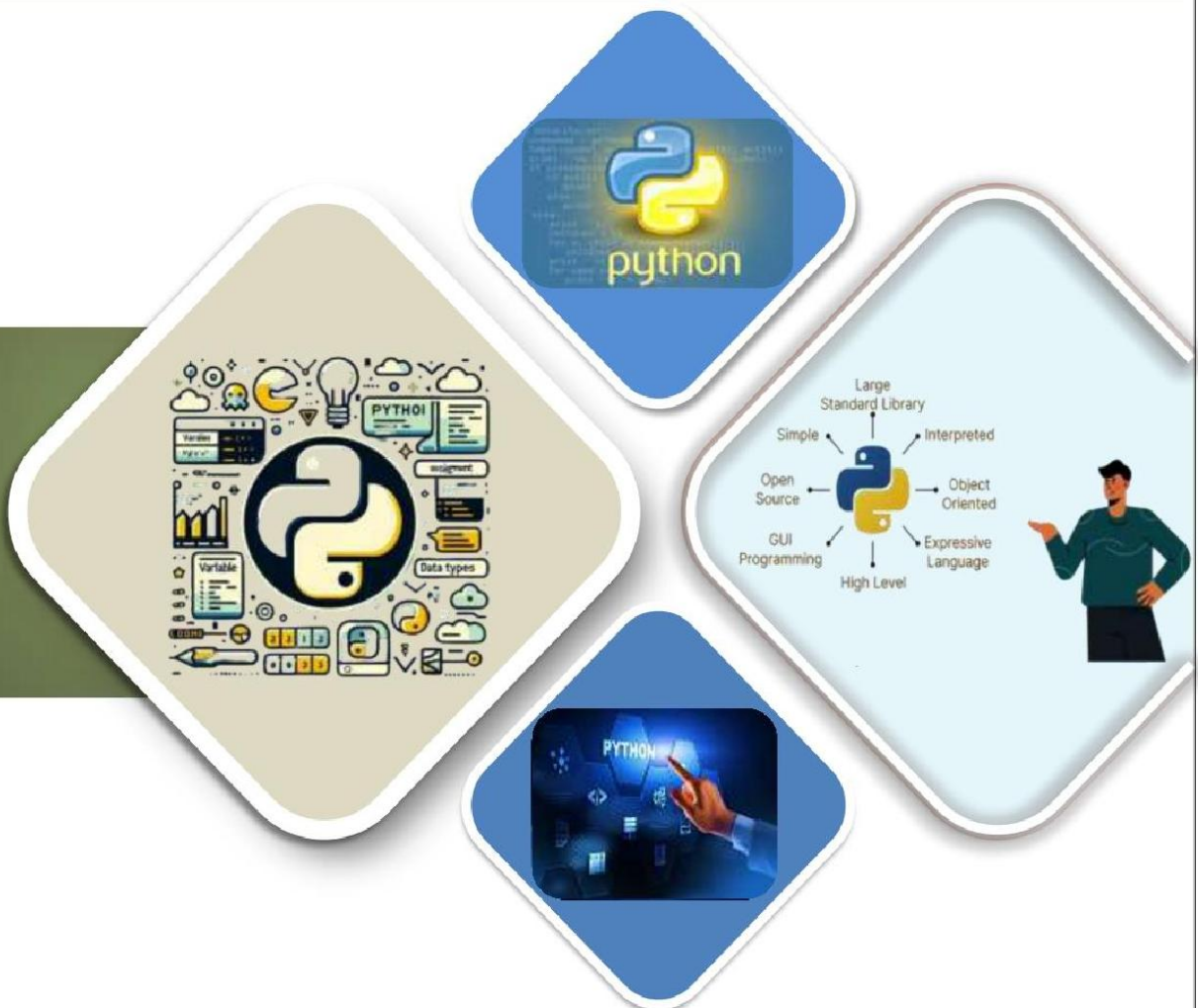


SCHEME : K

Name : _____
Roll No. : _____ Year : 20__ 20__
Exam Seat No. : _____

LABORATORY MANUAL FOR BASIC PYTHON PROGRAMMING (313011)



ELECTRONICS ENGINEERING GROUP



**MAHARASHTRA STATE BOARD OF
TECHNICAL EDUCATION, MUMBAI
(Autonomous) (ISO 9001: 2015) (ISO/IEC 27001:2013)**

VISION

To ensure that the Diploma level Technical Education constantly matches the latest requirements of Technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the challenging technological & environmental challenges.

QUALITY POLICY

We, at MSBTE are committed to offer the best-in-class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation, and monitoring system along with adequate faculty development programmes.

CORE VALUES

MSBTE believes in the following:

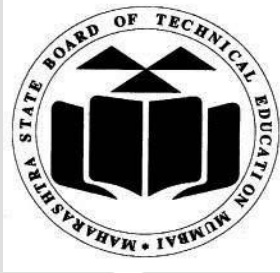
- Skill development in line with industry requirements.
- Industry readiness and improved employability of Diploma holders.
- Synergistic relationship with industry.
- Collective and Cooperative development of all stake holders.
- Technological interventions in societal development.
- Access to uniform quality technical education.

**A Laboratory Manual
for**

Basic Python Programming

(313011)

“K-SCHEME”



Semester-III

(AO/ DE/ EJ/ ET/ EX/ IC/ IE/ IS/ ML/ MU)

Maharashtra State Board of Technical Education, Mumbai

(Autonomous) (ISO 9001:2015) (ISO/IEC 27001:2013)



**MAHARASHTRA STATE BOARD OF TECHNICAL
EDUCATION**

Certificate

This is to certify that Mr. / Ms..... Roll No.....
, of Third Semester of Diploma in
of Institute.....
(Institute Code:) has completed the term work satisfactorily in Course
Basic Python Programming (313011) for the academic year 20.....-20.... as
prescribed in the curriculum.

Place:

Enrollment No:

Date:

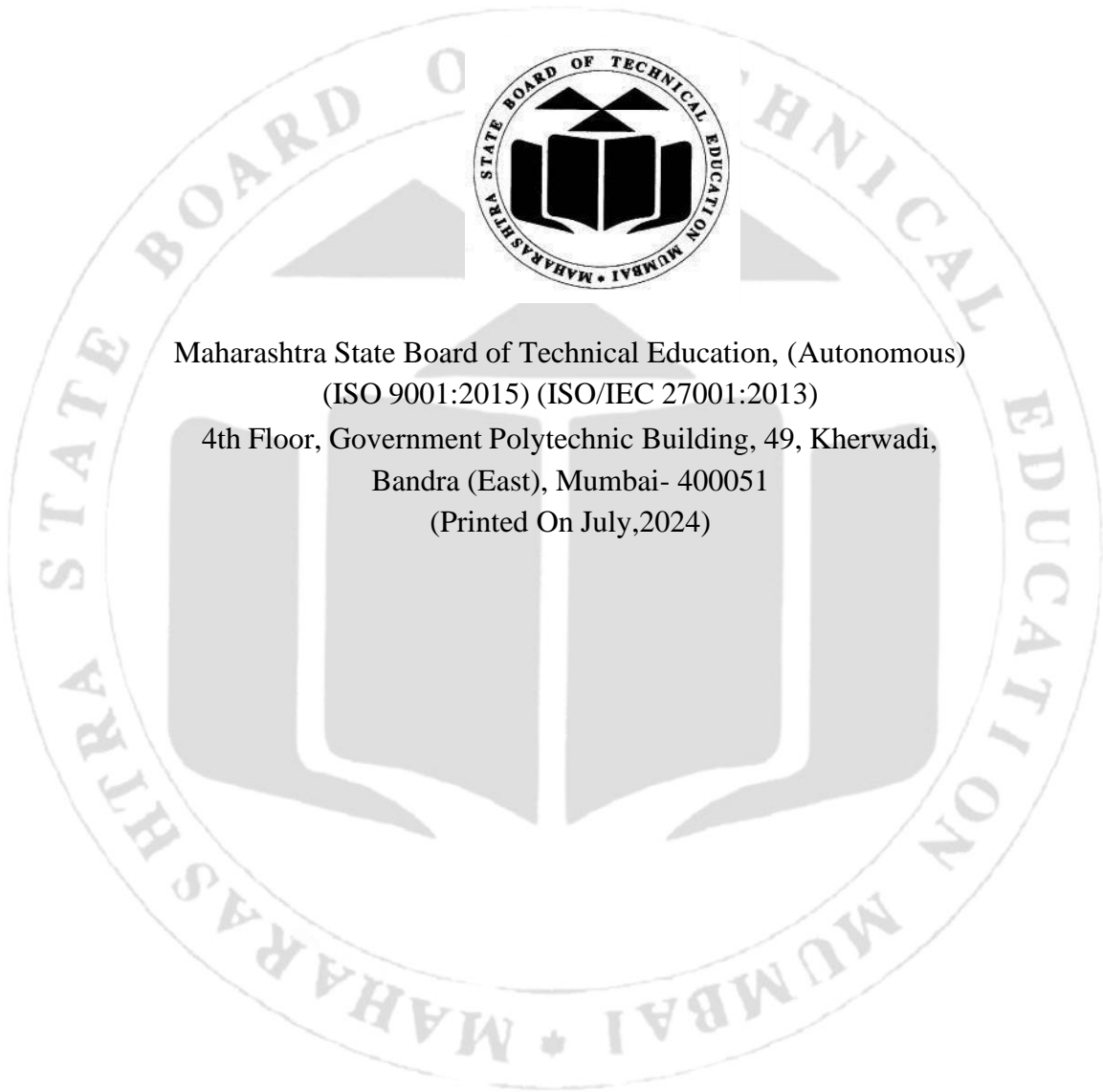
Exam. Seat No:

Subject Teacher

Head of the Department

Principal





Maharashtra State Board of Technical Education, (Autonomous)
(ISO 9001:2015) (ISO/IEC 27001:2013)

4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai- 400051

(Printed On July,2024)

Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much-needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative 'K' Scheme curricula for engineering diploma programmes with outcome-based education as the focus and accordingly, relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work making each teacher, instructor and student to realize that every minute of the laboratory time need to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome- based curriculum, every practical has been designed to serve as a *'vehicle'* to develop this industry identified competency in every student. The practical skills are difficult to develop through 'Chalk and duster' activity in the classroom situation. Accordingly, the 'K' scheme laboratory manual development team designed the practical to *focus* on the *outcomes*, rather than the traditional age old practice of conducting practical to 'verify the theory' (which may become a byproduct along the way).

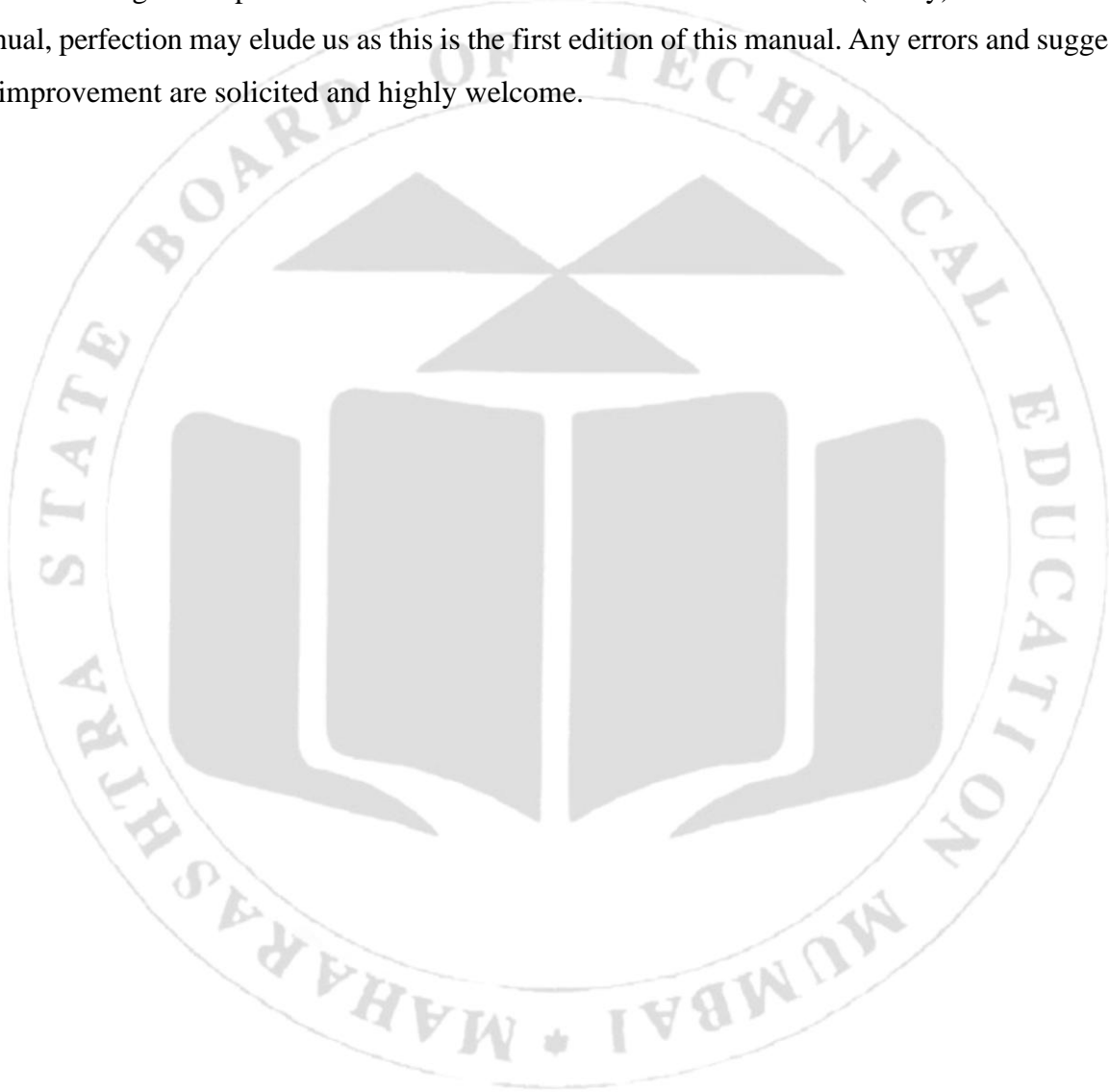
This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes that serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student-centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

Basic Python Programming(313011)

This course focuses on Electronics based industries needs to deal with creating circuits design, simulation, signal processing and control systems which can be developed using Python. This course deals with the basics of python to enhance the programming skills of diploma students. The course will enable students to write python programs as well as use different python libraries to solve given problems

Although best possible care has been taken to check for errors (if any) in this laboratory manual, perfection may elude us as this is the first edition of this manual. Any errors and suggestions for improvement are solicited and highly welcome.



Programmed Outcomes (POs) and Program Specific Outcomes (PSOs) to be achieved through Practical's of this course.

PO1. Basic and Discipline Specific Knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO2. Problem Analysis: Identify and analyses well-defined engineering problems using codified standard methods.

PO3. Design/ Development of Solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4. Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO5. Engineering Practices for Society, Sustainability and Environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO6. 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.

PO7. Life-Long Learning: Ability to analyze individual needs and engage in updating in the context of technological changes.

Program Specific Outcomes (PSOs):

PSO1. Electronics and Telecommunication systems: Maintain Various types of Electronics and Telecommunication systems.

PSO2. EDA Tools Usage: Use EDA tools to develop simple Electronics and Telecommunication engineering related circuit.

Practical- Course Outcome Matrix

Course Outcomes (COs)						
CO1 - Develop script to demonstrate use of basic building blocks of python.						
CO2 - Implement conditional and looping statements for given problem statement.						
CO3 - Perform operations on sequence structures in python.						
CO4 - Implement basics of object oriented programming concepts.						
CO5 - Create modules and packages for given purpose.						
Sr. No	Title of the Practical	CO1	CO2	CO3	CO4	CO5
1	a) Install and configure Python IDE. b) Write Python program to display message on screen.	√				
2	*a) Write simple Python program to calculate equivalent registers connected in series and parallel. Accept values of R1, R2 and R3 from the user. *b) Write simple Python program to calculate value of voltage by applying Ohm's law. Accept value of Current(I) and Resistance(R) from the user.	√				
3	Write program to check whether entered frequency is radio frequency or audio frequency.		√			
4	*a)Write program to display various radio frequency bands using if..elseif ladder. *b)Write program to display resistor color code using switch statement.		√			
5	*a)Write a simple Python program to demonstrate use of control loops: i) while ii) do while *b) Create a simple program, to demonstrate use of: for loop in Python (e.g.: various pattern building, printing multiplication table, checking palindrome number etc.)		√			
6	*Write Python program to perform following operations on List: a) Create b) Access c) Update d) Delete elements from list			√		

Basic Python Programming(313011)

Sr. No.	Title of the Practical	CO1	CO2	CO3	CO4	CO5
7	Develop Python program to perform following operations on Tuples: a) Create b) Access c) Update d) Delete Tuple elements			√		
8	Write Python program to perform following operations on Set: a) Create b) Access c) Update d) Delete Access Set elements			√		
9	*Create a program to perform following operations on Dictionaries in Python: a) Create b) Access c) Update d) Delete e) Looping through Dictionary			√		
10	a) *Create python program to demonstrate use of math built-in function. b) *Create python program to demonstrate use of string built-in function.				√	
11	Write python programs to define function with arguments. a) Calculate factorial of a number b) Swapping of two variables				√	
12	Write programs to define function with default arguments.				√	
13	*Create a program to demonstrate use of: Built-in module (e.g.: numeric, mathematical functional and programming module) in Python.					√
14	Write program to create a userdefined module (e.g.:building calculator) in python.					√
15	*Develop Python program to demonstrate use of NumPy package for creating, accessing and performing different array operations.					√
16	Write program to demonstrate the use of user defined packages in Python.					√

Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.



Guidelines to Teachers

1. For incidental writing on the day of each practical session every student should maintain a dated logbook for the whole semester, apart from this laboratory manual, which she/he has to submit for assessment to the teacher in the next practical session.
2. Teachers should give opportunity to students for hands-on after the demonstration.
3. Assess the skill achievement of the students and COs of each unit.
4. Explain prior concepts to the students before starting of each experiment.
5. List of few sample questions for reference are given. Teachers must design more such questions so as to ensure the achievement of identified CO.
6. Teacher should ensure that the practical skill and competencies are developed in the students after the completion of the practical exercise.
7. Teacher may provide additional knowledge and skills to the students even though it's not covered in the manual but are expected from the students by the industries.
8. Teacher may suggest the students to refer additional related literature of the technical papers/ Reference books/ Seminar proceedings, etc.
9. Teacher shall assess the performance of students continuously as per norms prescribed by MSBTE.
10. During assessment teacher is expected to ask questions to the students to tap their Achievements grading related knowledge and skills so that student can prepare while submitting record of the practical focus should be given on development of enlisted skills rather than theoretical knowledge.

Instructions for Students

1. Understand the purpose of practical and its implementation.
2. Student shall develop practical skills as expected by the industries.
3. Listen carefully to the instructions given by the teacher about importance of relevant program outcomes, relevant course outcomes, practical significance, competency and practical skills, practical outcome and the theoretical background during the practical session.
4. Write the answers of the questions allotted by the teacher during practical session.
5. Student should develop the habit of group discussion related to the practical, so that exchange of knowledge/skills could take place.
6. Student shall attempt to develop related hands-on-skills to gain confidence.
7. Student shall refer technical magazines, websites related to the scope of the course.
8. Student should develop habit to submit the practical, exercise continuously and progressively on the scheduled dates and should get the assessment done.
9. Student should be well prepared while submitting the write up of the exercise.
10. Student should not hesitate to ask any difficulty faced during conduct of practical.

Content Page
List of Practical's and Progressive Assessment Sheet

Sr. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks (25)	Dated sign.of teacher	Remarks (if any)
1.	a) Install and configure Python IDE. b) Write Python program to display message on screen.	1					
2.	*a) Write simple Python program to calculate equivalent registers connected in series and parallel. Accept values of R1, R2 and R3 from the user. *b) Write simple Python program to calculate value of voltage by applying Ohm's law. Accept value of Current(I) and Resistance(R) from the user.	13					
3.	Write program to check whether entered frequency is radio frequency or audio frequency.	20					
4.	*a) Write program to display various radio frequency bands using if..elseif ladder. *b) Write program to display resistor color code using switch statement.	27					
5.	*a) Write a simple Python program to demonstrate use of control loops: i) while ii) do while *b) Create a simple program, to demonstrate use of: for loop in Python (e.g.: various pattern building, printing multiplication table, checking palindrome number etc.)	35					
6	*Write Python program to perform following operations on List: a) Create b) Access c) Update d) Delete elements from list.	44					
7.	Develop Python program to perform following operations on Tuples: a) Create b) Access c) Update d) Delete Tuple elements	53					

Basic Python Programming(313011)

Sr. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks (25)	Dated sign. of teacher	Remarks (if any)
8.	Write Python program to perform following operations on Set: a) Create b) Access c) Update d) Delete Access Set elements	62					
9.	*Create a program to perform following operations on Dictionaries in Python: a) Create b) Access c) Update d) Delete e) Looping through Dictionary	72					
10.	*a) Create Python program to demonstrate use of math built-in function. *b) Create Python program to demonstrate use of string built-in function.	79					
11.	Write Python programs to define function with arguments. a) Calculate factorial of a number b) Swapping of two variables	85					
12.	Write programs to define function with default arguments.	90					
13.	*Create a program to demonstrate use of: Built-in module (e.g.: numeric, mathematical functional and programming module) in Python.	94					
14.	Write program to create a user-defined module (e.g.: building calculator) in python.	99					
15.	*Develop Python program to demonstrate use of NumPy package for creating, accessing and performing different array operations.	104					
16.	Write program to demonstrate the use of user defined packages in Python.	109					
Total Marks							
Total Marks (Scaled to 25 Marks)							

Note: Marks to be transferred to Performa of (CIAAN-K format).

Practical No. 1: a) Install and configure Python IDE.

b) Write Python program to display message on screen.

I. Practical Significance

Python is a high-level, general-purpose, interpreted, interactive, object-oriented dynamic programming language. Student will able to select and install appropriate installer for Python in windows and package manager for Linux in order to setup Python environment for running programs.

II. Industry /Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO1 - Develop script to demonstrate use of basic building blocks of python.

IV. Laboratory Learning Outcome

LLO 1.1 Install Python Integrated Development Environment.

V. Relevant Affective Domain related Outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

Python was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV Show called 'Monty Python's Flying Circus' and not after Python-the snake.

Installing Python in Windows:

- Open any internet browser. Type <http://www.Python.org/downloads/> in address bar and press Enter.
- Home page of Python will have displayed as shown in Fig. 1

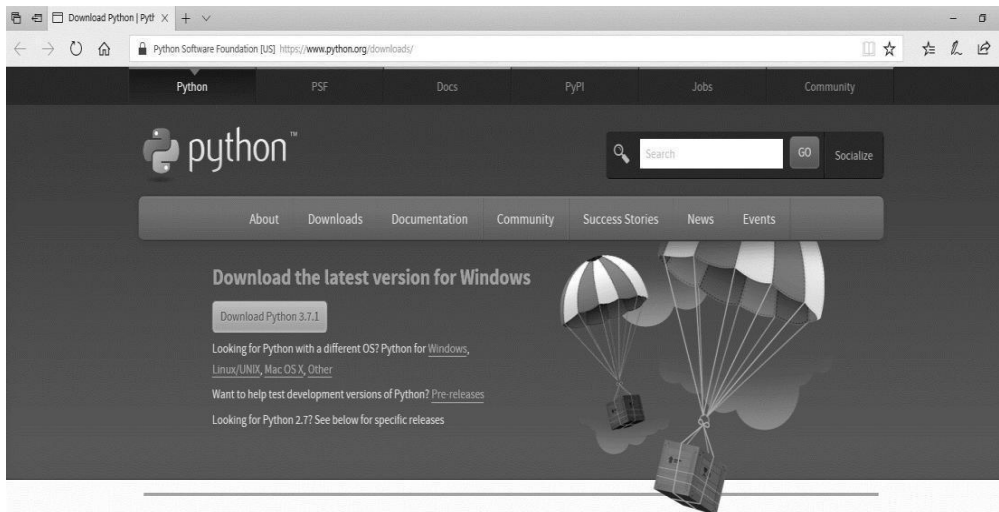


Fig. 1: Home Page

- Click on download the latest version for windows, which shows latest version as shown in Fig. 2

Looking for a specific release?
Python releases by version number:

Release version	Release date		Click for more
Python 3.7.1	2018-10-20	Download	Release Notes
Python 3.6.7	2018-10-20	Download	Release Notes
Python 3.5.6	2018-08-02	Download	Release Notes
Python 3.4.9	2018-08-02	Download	Release Notes
Python 3.7.0	2018-06-27	Download	Release Notes
Python 3.6.6	2018-06-27	Download	Release Notes
Python 2.7.15	2018-05-01	Download	Release Notes

[View older releases](#)

Fig.2: Python release versions

- Open the Python 3.7.1 version pack and double click on it to start installation and installation windows will be open as shown in Fig. 3.



Fig. 3: Installation Type

- Click on next install now for installation and then Setup progress windows will be opened as shown in Fig. 4.

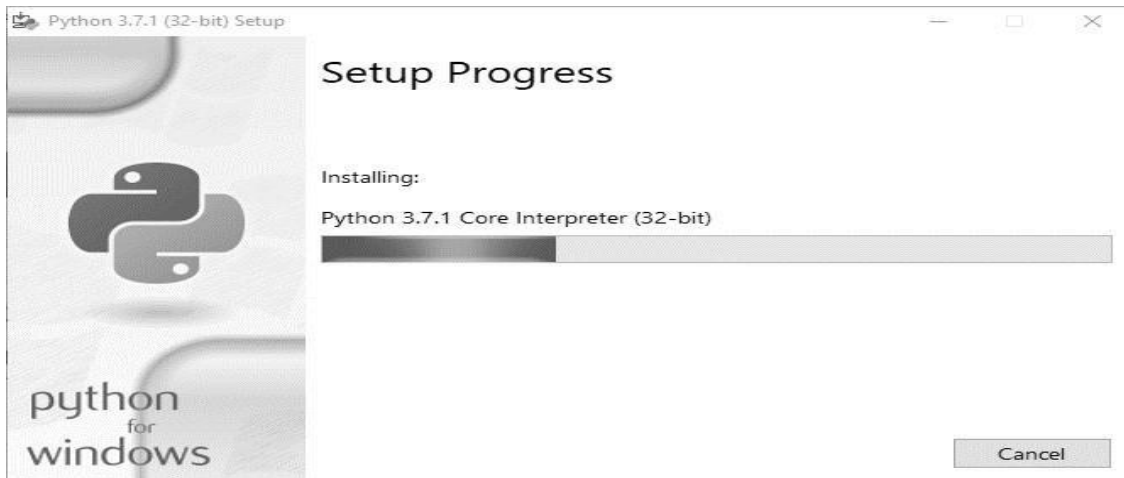


Fig. 4: Setup Progress

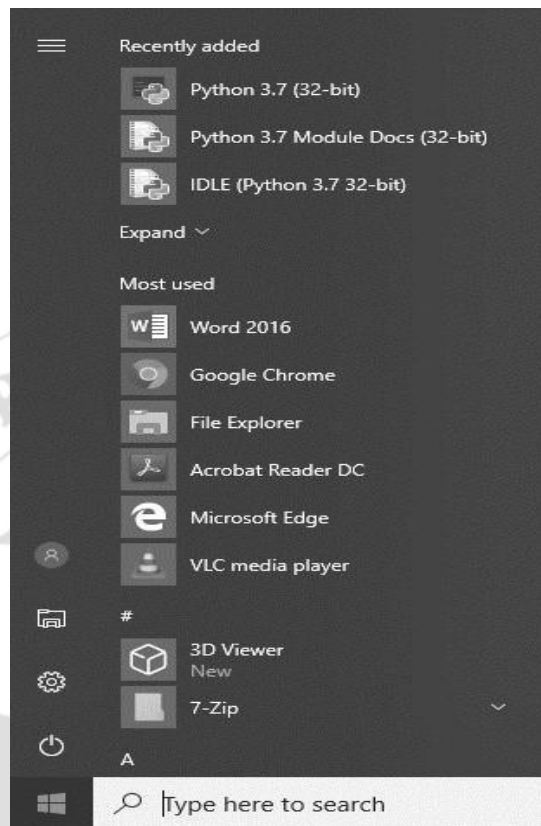
- After complete the installation, Click on close button in the windows as shown in Fig. 5.



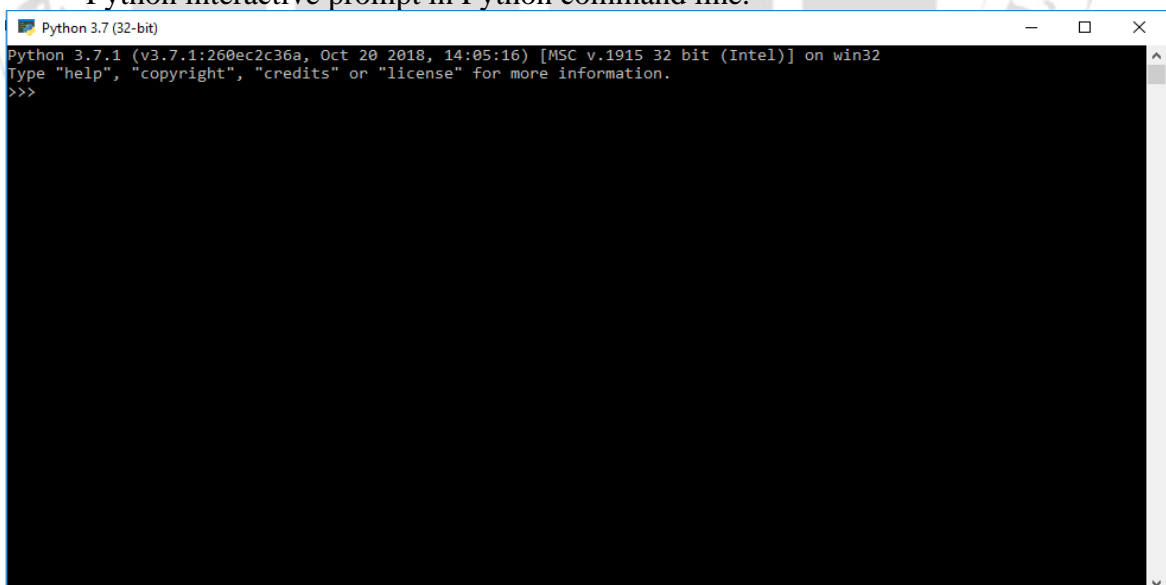
Fig. 5: Setup Completion Starting Python in different modes:

1) Starting Python (Command Line)

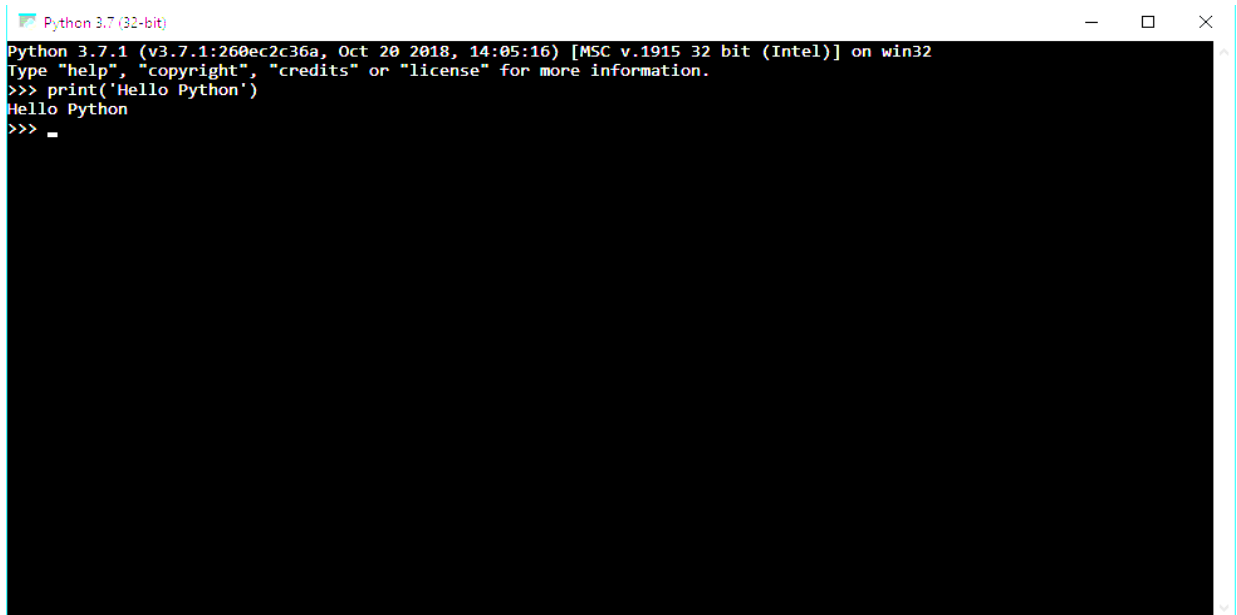
- Press start button



- Click on all programs and then click on Python 3.7 (32 bit). You will see the Python interactive prompt in Python command line.



- Python command prompt contains an opening message >>> called command prompt. The cursor at command prompt waits for to enter Python command. A complete command is called a statement. For example check first command to print message.

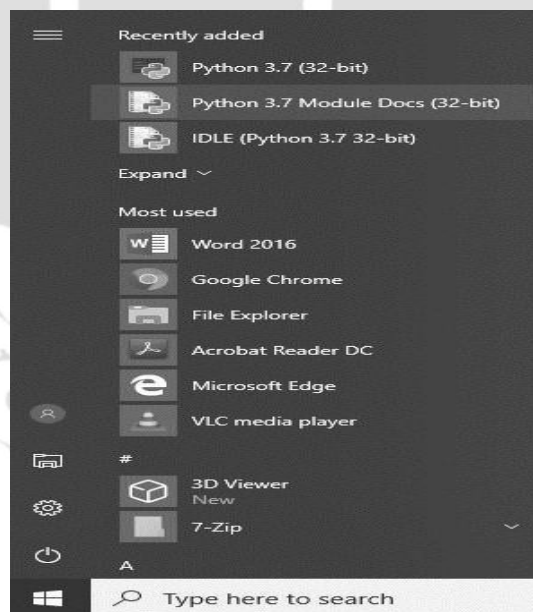


```
Python 3.7.1 (32-bit)
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello Python')
Hello Python
>>> _
```

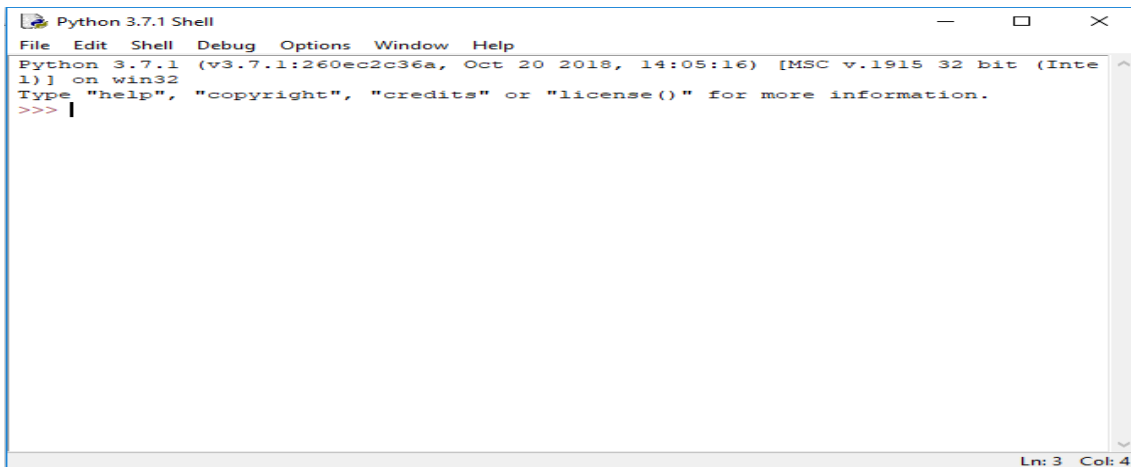
exit from the command line of Python, press Ctrl+z followed by Enter or Enter exit() or quit() and Enter.

2) Starting Python IDLE

- Press start button and click on IDLE (Python 3.7 32 bit) options.

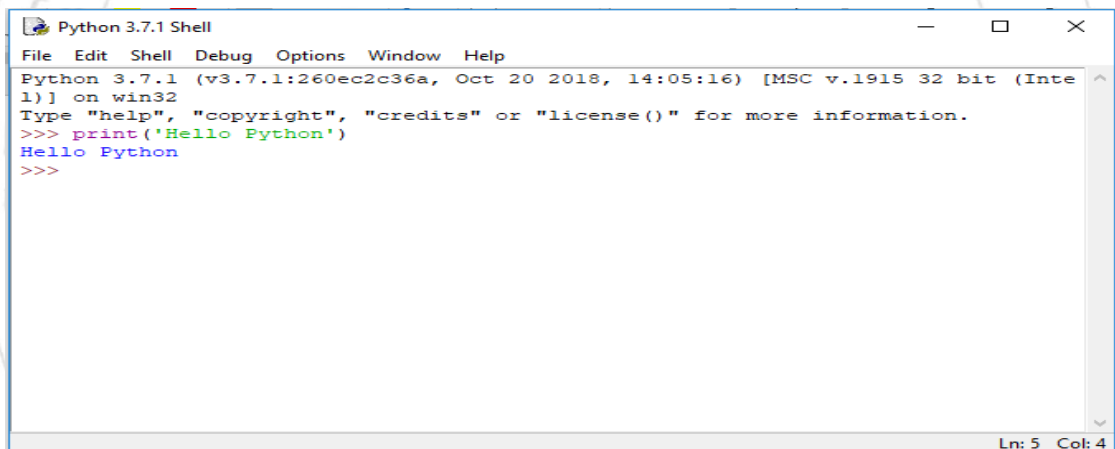


- You will see the Python interactive prompt i.e. interactive shell.



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

- Python interactive shell prompt contains opening message >>>, called shell prompt. A cursor is waiting for the command. A complete command is called a statement. When you write a command and press enter, the Python interpreter will immediately display the result.



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello Python')
Hello Python
>>>
```

VII. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write steps for installing Python on window.
2. State IDLE in Python.
3. List key features of Python.
4. List different modes of Programming in Python.
5. Describe procedure to execute program using Interactive Mode.
6. State the steps involved in executing the program using Script Mode.
7. State the procedure to make file executable.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

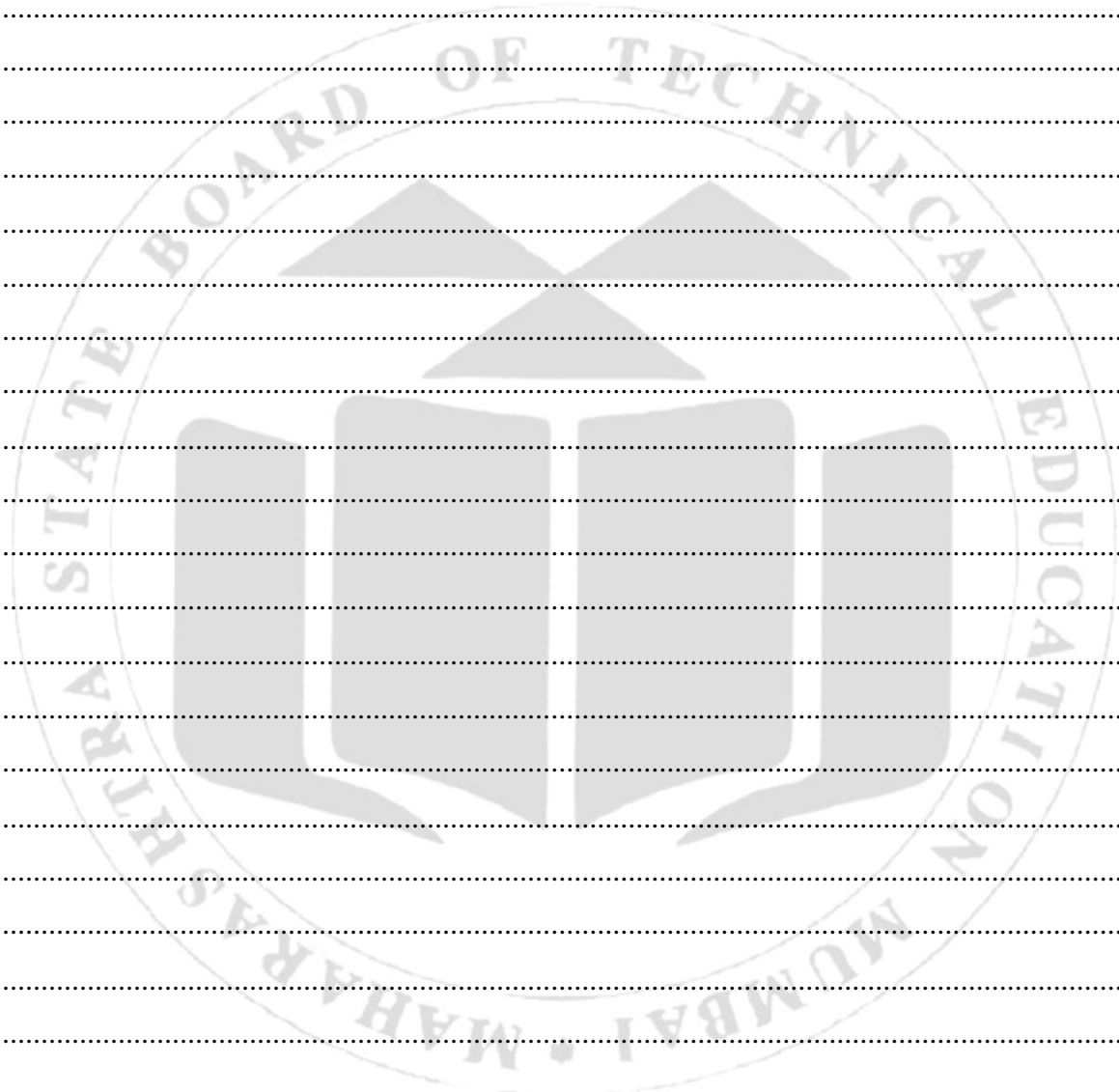
.....

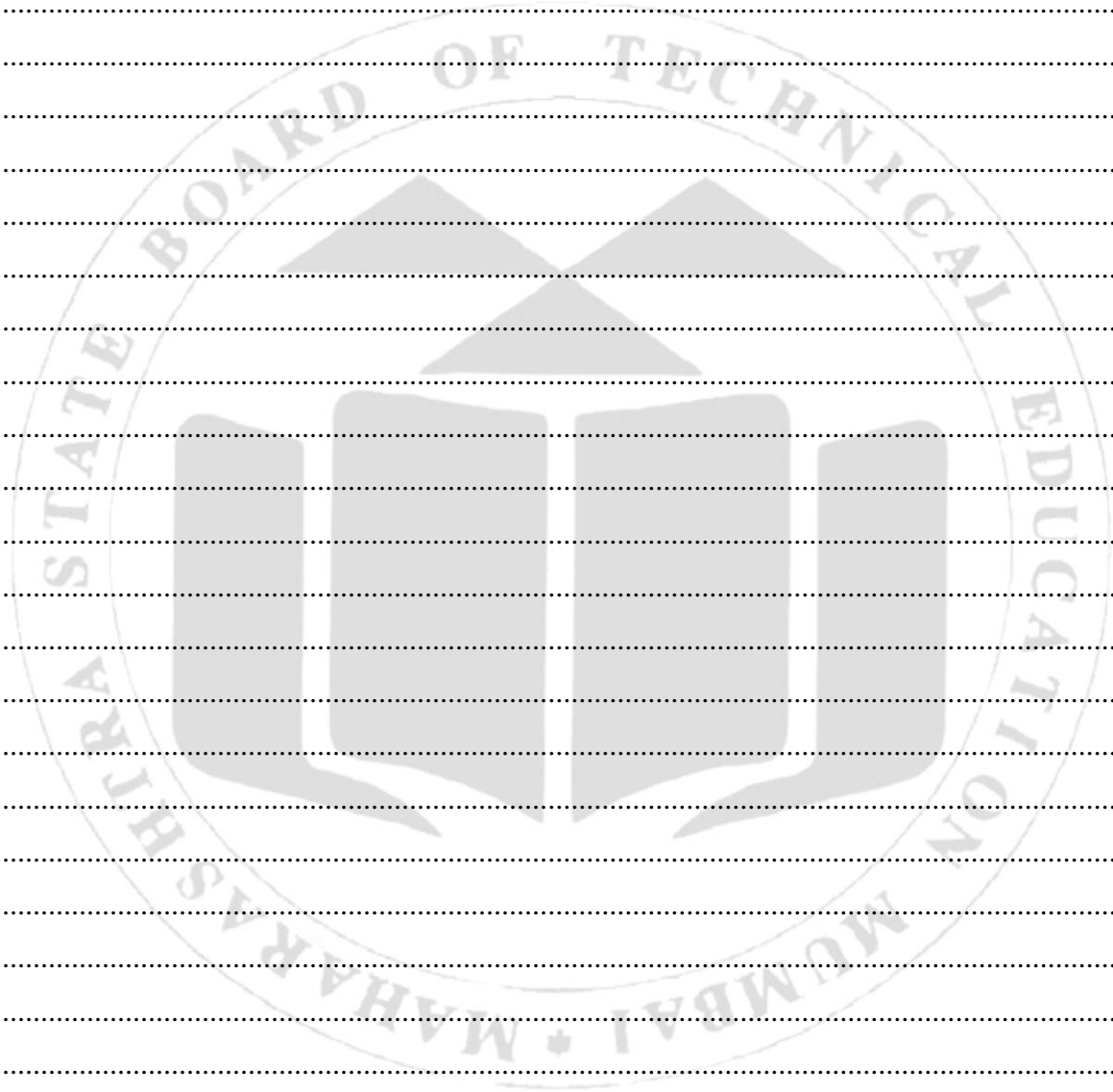
.....

.....

.....

.....





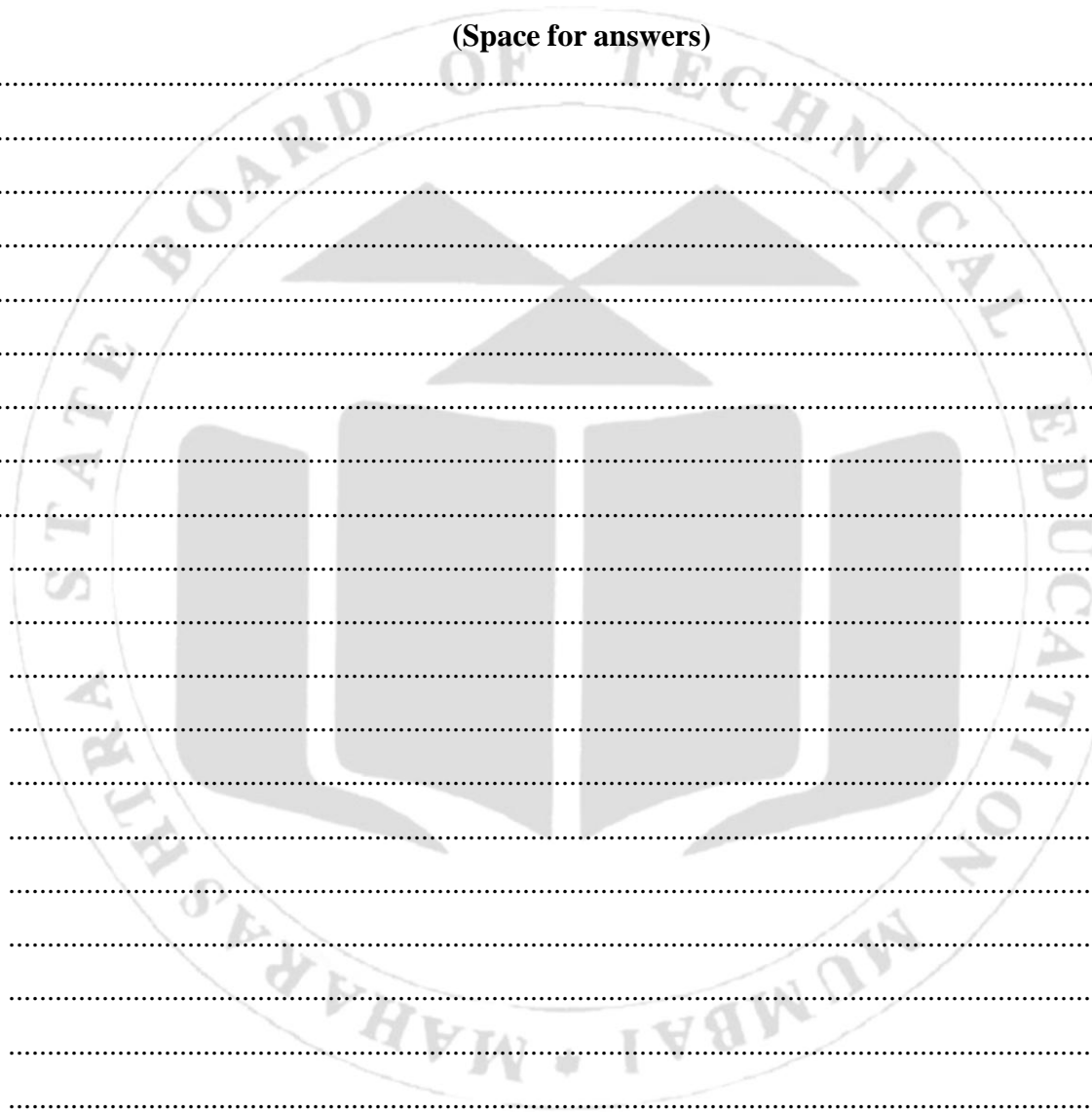
X. Exercise

Note: Faculty must ensure that every group of students use different input value.

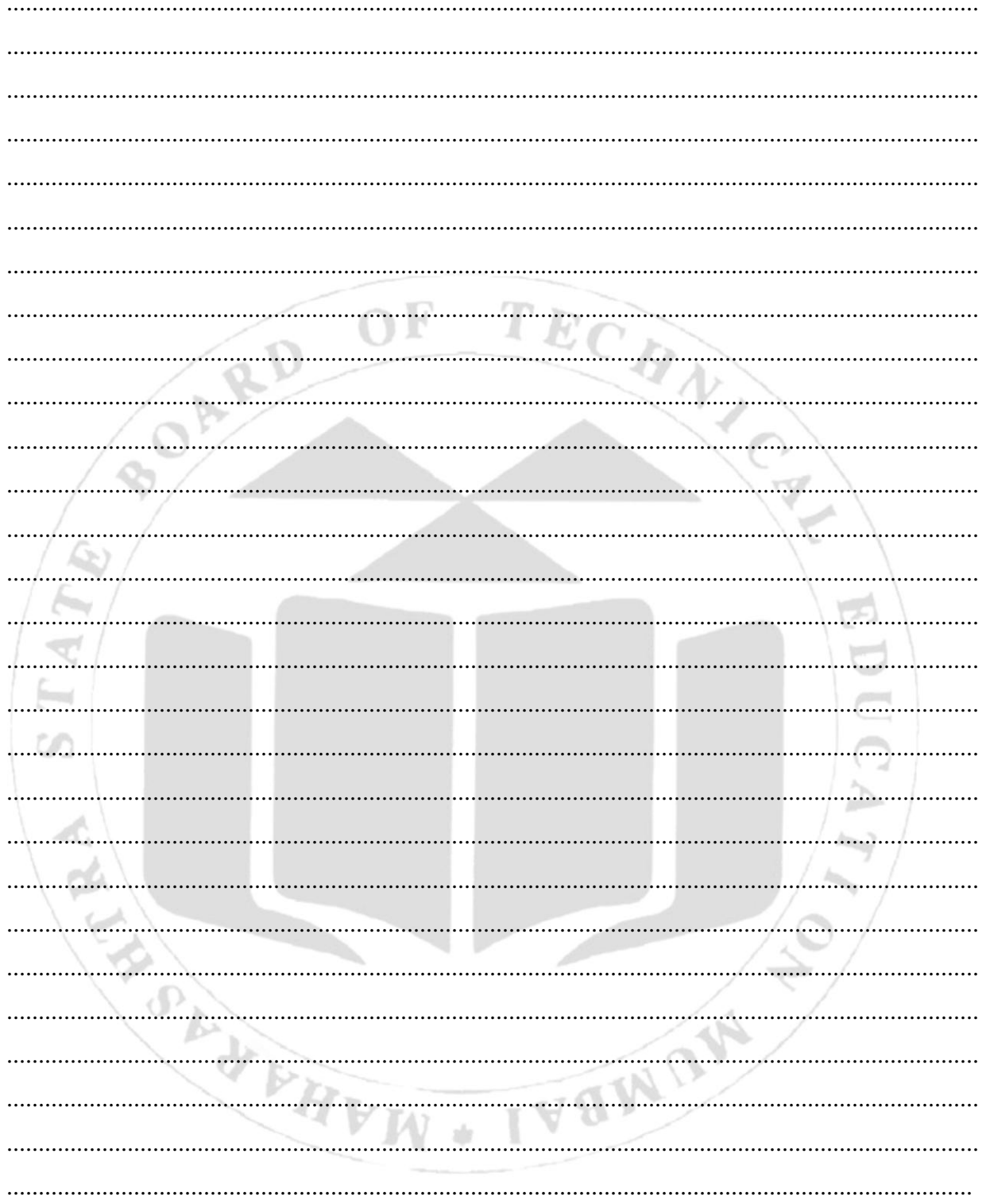
(Use blank space for answers or attach more pages if needed)

1. Print the version of Python.
2. Write steps to be followed to load Python interpreter in windows.
3. Write a Python program to display your name using Interactive Mode.
4. Write a Python program to display “MSBTE” using Script Mode.

(Space for answers)



A large watermark of the Maharashtra State Board of Technical Education logo is centered on the page. The logo is circular and contains the text 'MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI' around the perimeter. In the center, there is a stylized emblem featuring a book and a lamp. Below the watermark, the page is filled with horizontal dotted lines for writing answers.



XI. References / Suggestions for further Reading

1. <https://www.howtogeek.com/197947/how-to-install-Python-on-windows/>
2. <https://www.ics.uci.edu/~pattis/common/handouts/Pythoneclipsejava/Python.html>
2. <https://www.maketecheasier.com/set-up-Python-windows10/>
3. <https://www.youtube.com/watch?v=gV-yluunPjI>
4. https://www.youtube.com/watch?v=N9jTJPt_xu8

XII. Assessment Scheme

Performance Indicators		Weightage
Process related: 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to sample questions	20%
Total 25 Marks		100%

Marks obtained			Dated Signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No 2: *a) Write simple Python program to calculate equivalent registers connected in series and parallel. Accept values of R1, R2 and R3 from the user.

***b) Write simple Python program to calculate value of voltage by applying Ohm's law. Accept value of Current(I) and Resistance(R) from the user.**

I. Practical Significance

The practical significance of using operators in Python includes performing essential mathematical and logical computations, enabling complex decision-making in control flow, and facilitating data manipulation and low-level programming tasks. Additionally, membership and identity operators enhance efficiency in data structure operations. Overall, operators contribute to code readability and conciseness, making Python powerful for various applications.

II Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III Course Level Learning Outcome

CO1-Develop script to demonstrate use of basic building blocks of python.

IV Laboratory Learning Outcome

LLO 2.1 Use operators in Python.

V Relevant Affective domain related Outcome(s)

1. Follow safety practices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

In Python, operators are special symbols or keywords that carry out operations on values and python variables. They serve as a basis for expressions, which are used to modify data and execute computations. Python contains several operators, each with its unique purpose.

Types of Python Operators: Python language supports various types of operators, which are:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison (Relational) Operators
4. Logical Operators
5. Identity Operators
6. Membership Operators
7. Bitwise Operators

1. Arithmetic Operators:

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
	Multiplication	$x * y$
/	Division	x / y
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

2. Assignment Operators:

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
//=	$x //= 3$	$x = x // 3$
**=	$x ** = 3$	$x = x ** 3$
&=	$x \& = 3$	$x = x \& 3$
=	$x = 3$	$x = x 3$
^=	$x \wedge = 3$	$x = x \wedge 3$
>>=	$x >> = 3$	$x = x >> 3$
<<=	$x << = 3$	$x = x << 3$
:=	<code>print(x := 3)</code>	$x = 3$ <code>print(x)</code>

3. Comparison Operators:

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	$x == y$
!=	Not equal	$x != y$
>	Greater than	$x > y$
<	Less than	$x < y$
>=	Greater than or equal to	$x >= y$
<=	Less than or equal to	$x <= y$

4. Logical Operators:

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	$x < 5$ and $x < 10$
or	Returns True if one of the statements is true	$x < 5$ or $x < 4$
not	Reverse the result, returns False if the result is true	not($x < 5$ and $x < 10$)

5. Identity Operators:

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

6. Membership Operators:

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

7. Bitwise Operators:

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	$x \& y$
	OR	Sets each bit to 1 if one of two bits is 1	$x y$
^	XOR	Sets each bit to 1 if only one of two bits is 1	$x \wedge y$
~	NOT	Inverts all the bits	$\sim x$
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	$x \ll 2$
>>	Signed rightshift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	$x \gg 2$

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

- 1) Write simple Python program to calculate equivalent registers connected in series and parallel. Accept values of R1, R2 and R3 from the user.
- 2) Write simple Python program to calculate value of voltage by applying Ohm’s law. Accept value of Current(I) and Resistance(R) from the user.
- 3) Write a program to find the square root of a number.
- 4) Write a program to find the area of Rectangle.
- 5) Write a program to calculate area and perimeter of the square.
- 6) Write a program to calculate surface volume and area of a cylinder.
- 7) Write a program to swap the value of two variables.

[Space for Answer]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References / Suggestions for further Reading

1. https://www.w3schools.com/Python/Python_operators.asp
2. <https://www.geeksforgeeks.org/basic-operators-Python/>
3. <https://www.guru99.com/Python-operators-complete-tutorial.html>
4. https://www.tutorialspoint.com/Python/Python_basic_operators
5. <https://www.youtube.com/watch?v=v5MR5JnKcZI>
6. https://www.youtube.com/watch?v=fN8yDMdl_aw

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 3: Write program to check whether entered frequency is radio frequency or audio frequency.

I. Practical significance

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE.

II. Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO 2- Implement conditional and looping statements for given problem statement.

IV. Laboratory Learning Outcome

LLO 3.1 Implement two-way branching statement.

V. Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

Simple If Statement

A simple `if` statement is used to execute a block of code only if a specified condition is true. If the condition is false, the block of code inside the `if` statement is skipped.

Syntax:

if condition:

 code to execute if condition is true

****Example**:**

```
age = 18
```

```
if age >= 18:
```

```
    print("You are eligible to vote.")
```

In this example, the message "You are eligible to vote." is printed only if the `age` is 18 or older.

If-Else Statement

An `if-else` statement is used to execute one block of code if the condition is true, and another block of code if the condition is false.

Syntax:

if condition:

```
# code to execute if condition is true
else:
    # code to execute if condition is false
```

****Example**:**

```
age = 16
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

In this example, the message "You are not eligible to vote." is printed because the `age` is less than 18.

Nested If Statement

A nested `if` statement is an `if` statement inside another `if` statement. This allows for more complex decision-making and multi-level condition checking.

Syntax:

```
if condition1:
    # code to execute if condition1 is true
    if condition2:
        # code to execute if both condition1 and condition2 are true
```

****Example**:**

```
age = 20
has_id = True
if age >= 18:
    if has_id:
        print("You are eligible to enter the club.")
    else:
        print("You need an ID to enter the club.")
else:
    print("You are not old enough to enter the club.")
```

In this example, the message "You are eligible to enter the club." is printed because both conditions (`age >= 18` and `has_id`) are true. If `has_id` were `False`, the message "You need an ID to enter the club." would be printed instead.

Summary:

Simple If: Executes a block of code if the condition is true.

If-Else: Executes one block of code if the condition is true, and another block if it is false.

Nested If: An `if` statement inside another `if` statement, allowing for multiple levels of condition checking.

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write a Python program that checks if the voltage of a battery is greater than 3.7V and prints "Battery is charged" if true.
2. Write a Python program that checks if a resistor value is within a tolerance range (e.g., 5% tolerance for a 100Ω resistor) and prints whether it is within tolerance or not.
3. Write a Python program that checks if a resistor value is within a tolerance range (e.g., 5% tolerance for a 100Ω resistor) and prints whether it is within tolerance or not.
4. Write a Python program that checks if a resistor value is within a tolerance range (e.g., 5% tolerance for a 100Ω resistor) and prints whether it is within tolerance or not.
5. Write a Python program that checks three sensor readings (voltage, current, and temperature) and prints an alert if any of them are out of their specified ranges (e.g., voltage: 3.3V ± 0.3V, current: 10mA ± 1mA, temperature: 25°C ± 5°C). The program should indicate which parameter(s) are out of range.

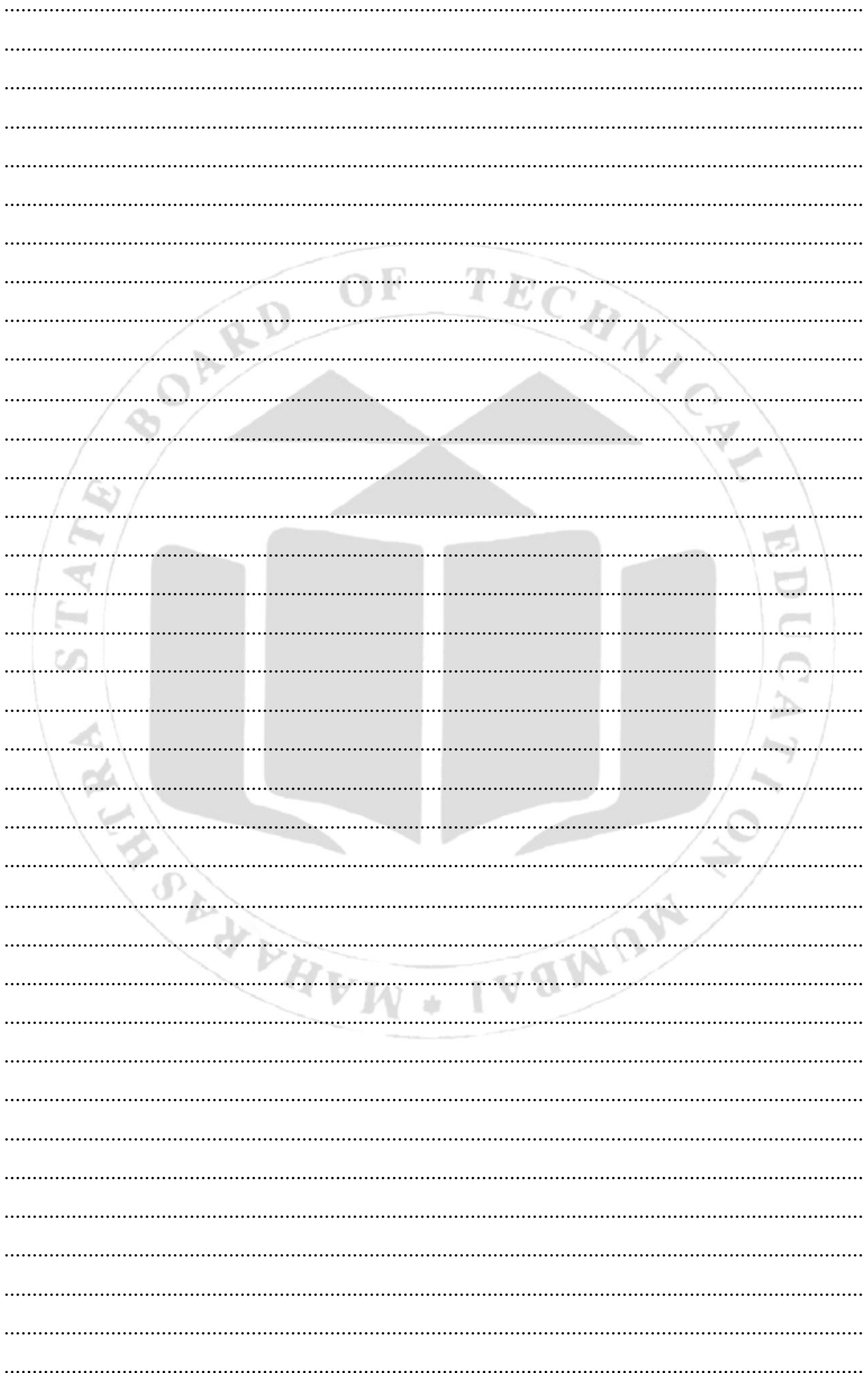
(Space for answers)

.....

.....

.....

.....



.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

- 1) Write program to check whether entered frequency is radio frequency or audio frequency.
- 2) Write a Python function to convert a given binary number to its equivalent gray code.
- 3) Write a program to find out absolute value of an input number.
- 4) Write a program to check the largest number among the three numbers.
- 5) Write a program to check if the input year is a leap year or not.
- 6) Write a program to check if a Number is Positive, Negative or Zero.
- 7) Write a program that takes the marks of 5 subjects and displays the grade.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

X References / Suggestions for further Reading

1. https://www.tutorialspoint.com/Python/Python_if_else
2. <https://www.programiz.com/Python-programming/if-elif-else>
3. <https://www.geeksforgeeks.org/decision-making-Python-else-nested-elif/>
4. <https://www.guru99.com/if-loop-Python-conditional-structures.html>
5. <https://www.youtube.com/watch?v=umfR4uPJPnw>
6. <https://www.youtube.com/watch?v=JKhnWZGHM54>

XI Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No 4: *a) Write program to display various radio frequency bands using if..elseif ladder.

***b) Write program to display resistor color code using switch statement.**

I. Practical significance

Multi-way branching statements in Python, such as if-elif-else, handle complex decision-making efficiently by outlining conditions and actions clearly. They improve code readability and are crucial in applications like grading systems, user role management, and state transitions. This ensures robust, flexible, and user-friendly applications.

II. Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning outcome

CO 2- Implement conditional and looping statements for given problem statement.

IV. Laboratory Learning outcome

LLO 4.1 Implement multi-way branching statement.

V. Relevant Affective Domain related Outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

Python if-elif-else Ladder

An if-elif-else ladder is an order of if statements connected by elif statements. This enables you to check for numerous conditions and run separate code blocks based on which condition is met.

Syntax of Python if-elif-else Ladder

```
if condition1: # code block 1  
elif condition2:  
    # code block 2  
elif condition3:  
    # code block 3  
else:
```

default code block

Example of Python if-elif-else ladder

```
voltage = 230
if voltage < 1.5:
    level = "Low"
elif 1.5 <= voltage < 3.3:
    level = "Medium"
elif 3.3 <= voltage < 5.0:
    level = "High"
elif 5.0 <= voltage < 12.0:
    level = "Very High"
else:
    level = "Dangerously High"
print(f"The voltage level is: {level}")
```

By understanding and effectively using the if-elif-else ladder, you can write Python programs that are capable of making nuanced decisions based on multiple conditions, thus making your software more robust and versatile.

Python does not have a built-in switch-case statement like some other programming languages (e.g., C, Java). However, you can achieve similar functionality using dictionaries or functions. Here are a few methods to mimic switch-case behavior in Python.

Python Match Case Statement Syntax

The match case statement in Python is initialized with the match keyword followed by the parameter to be matched. Then various cases are defined using the case keyword and the pattern to match the parameter. The “_” is the wildcard character that runs when all the cases fail to match the parameter value.

match parameter:

```
case pattern1:
    # code for pattern 1
case pattern2:
    # code for pattern 2
.
.
case patterN:
    # code for pattern N
case _:
    # default code block
```

Basic Python Programming (313011)

The following code has a function named week day(). It receives an integer argument, matches it with all possible weekday number values, and returns the corresponding name of day.

```
def weekday(n):
    match n:
        case 0: return "Monday"
        case 1: return "Tuesday"
        case 2: return "Wednesday"
        case 3: return "Thursday"
        case 4: return "Friday"
        case 5: return "Saturday"
        case 6: return "Sunday"
        case _: return "Invalid day number"
print (weekday(3))
print (weekday(6))
print (weekday(7))
```

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write a Python program that takes the color of a resistor as input and prints its corresponding resistance value based on the following color code:

Black: 0 ohms
Brown: 1 ohm
Red: 2 ohms
Orange: 3 ohms
Yellow: 4 ohms

Basic Python Programming (313011)

Green: 5 ohms

Blue: 6 ohms

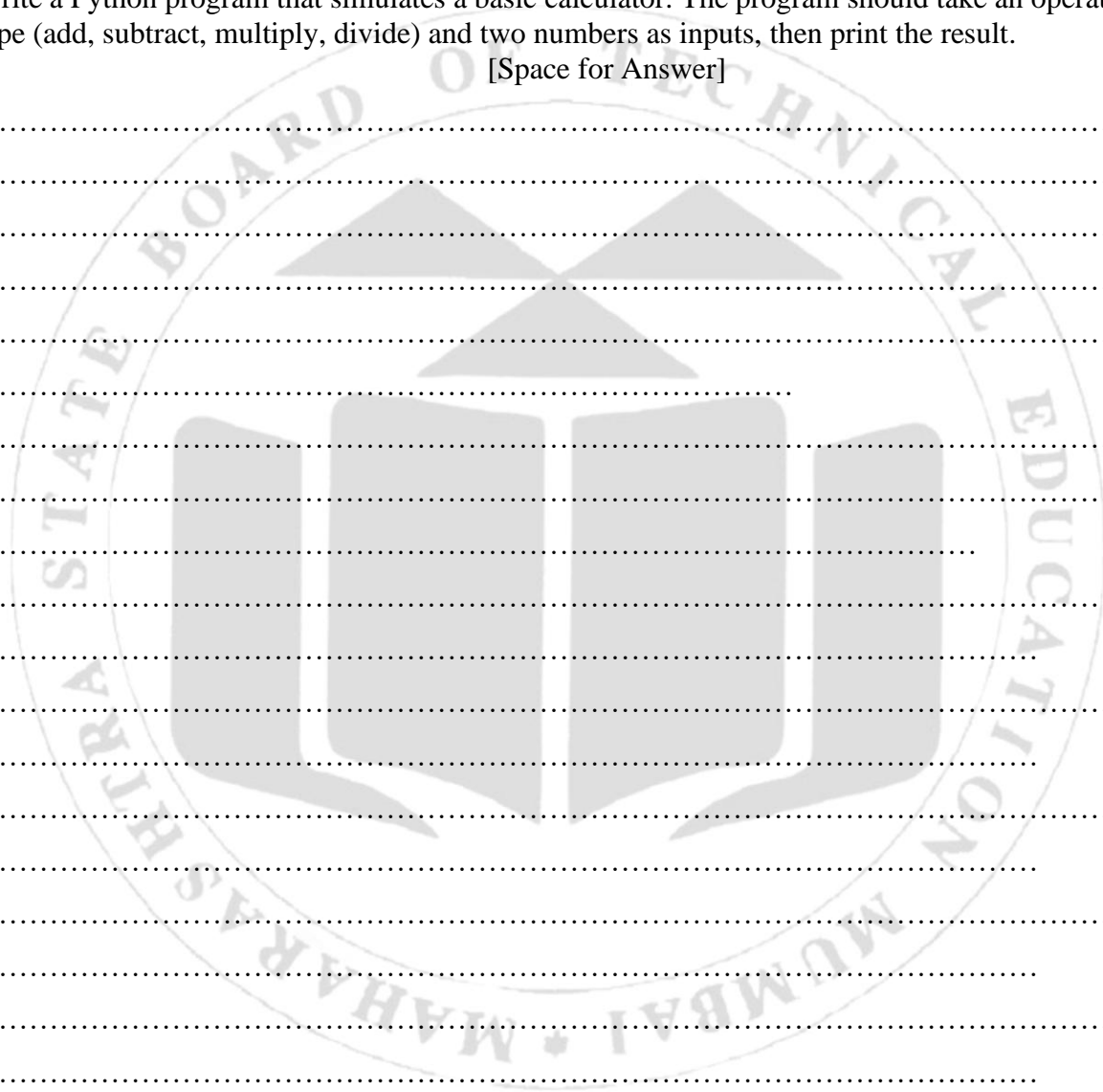
Violet: 7 ohms

Gray: 8 ohms

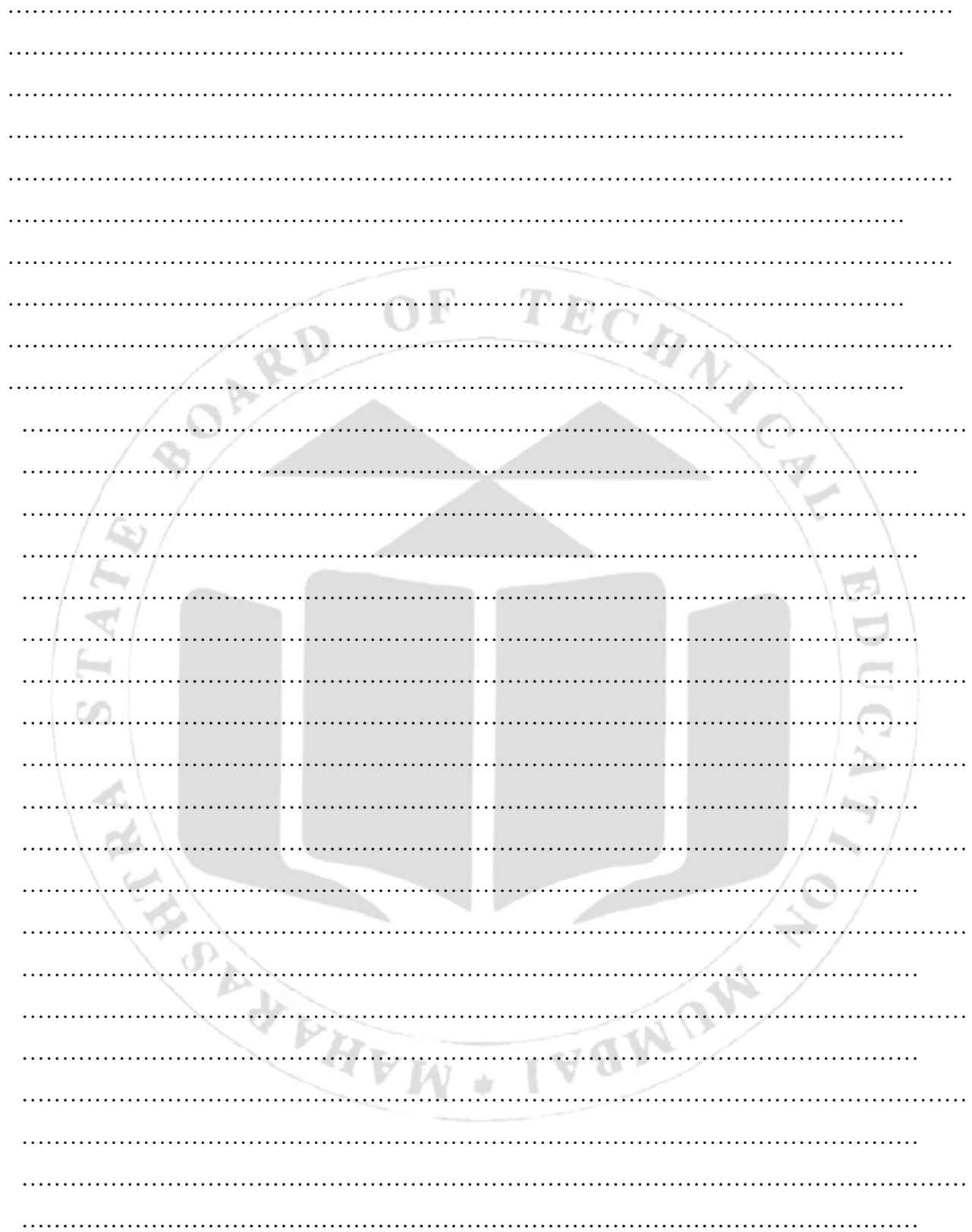
White: 9 ohms

2. Write a Python program that simulates the behavior of basic logic gates (AND, OR, NOT). The program should take the type of gate and the input values as arguments and print the result.
3. Write a Python program that simulates a basic calculator. The program should take an operation type (add, subtract, multiply, divide) and two numbers as inputs, then print the result.

[Space for Answer]



A large, faint watermark of the Maharashtra State Board of Technical Education logo is centered on the page. The logo features a shield with a book and a lamp, surrounded by the text 'MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI'. Below the watermark, the page contains numerous horizontal dotted lines for writing the answer.



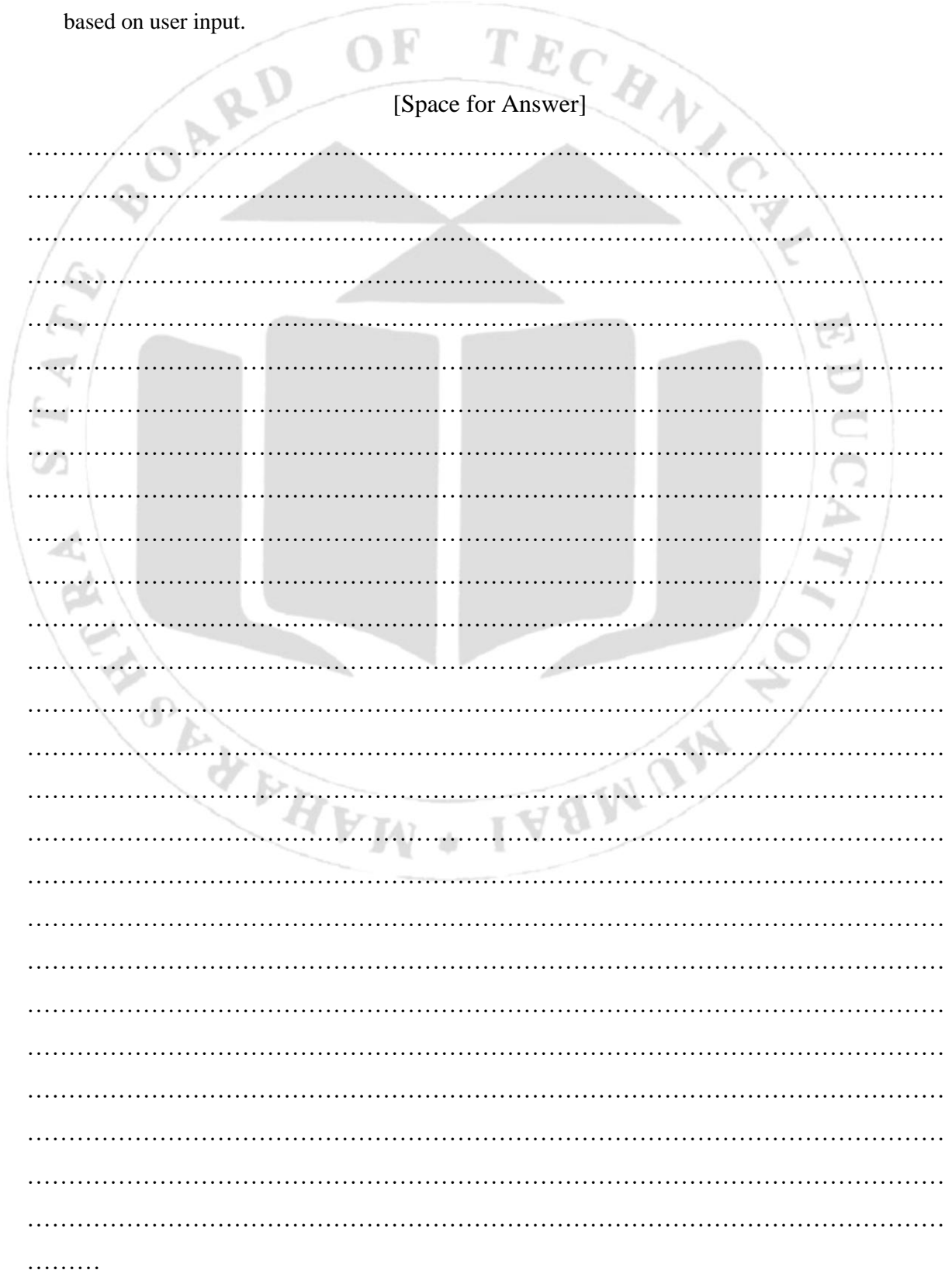
X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write program to display various radio frequency bands using if..elseif ladder.
2. Write program to display resistor color code using switch statement.
3. Write a Python program that converts temperatures between Celsius, Fahrenheit, and Kelvin based on user input.

[Space for Answer]



.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References / Suggestions for further reading

1. https://www.tutorialspoint.com/Python/Python_basic_syntax.htm
2. https://www.youtube.com/watch?v=skIy7ZhT_tk
3. <https://www.youtube.com/watch?v=3xp-ixFbDuE>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

- Practical No.5 *a) Write a simple Python program to demonstrate use of control loops:**
- i) while**
 - ii) do while**
- *b) Create a simple program, to demonstrate use of: for loop in Python (e.g.: various pattern building, printing multiplication table, checking palindrome number etc.)**

I. Practical significance

Control loops are essential for solving iterative problems by automating repetitive tasks and processing data collections efficiently. They are crucial in implementing algorithms like sorting and searching and handling dynamic input gracefully. In concurrent processing, loops manage multiple tasks simultaneously, enhancing performance and scalability in software applications.

II. Industry / Employer Expected outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning outcome

CO2 - Implement conditional and looping statements for given problem statement.

IV. Laboratory Learning outcome

LLO 5.1 Implement control loops for solving iterative problems.

V. Relevant Affective Domain related Outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

Python control loops, specifically `for` loops and `while` loops, are essential tools for solving iterative problems. They allow you to execute a block of code repeatedly based on specific conditions or for a certain number of iterations. Here's an explanation of how they work and their significance in solving iterative problems:

1. for Loops:

- for loops are used when you know the number of iterations or when iterating over a sequence (e.g., lists, tuples, strings).
- They iterate over each item in a sequence, executing a block of code for each iteration.

➤ **Syntax:**

for item in sequence:

 # Code block to execute for each item

Example:

```
numbers = [1, 2, 3, 4, 5]
```

```
for num in numbers:
```

```
    print(num)
```

2. while Loops:

A while loop is used to repeatedly execute a block of code as long as a specified condition is true. It continues iterating until the condition becomes false.

Syntax:

```
while condition:
```

```
    # Code block to execute while condition is true
```

Example:

```
num = 1
```

```
while num <= 5:
```

```
    print(num)
```

```
    num += 1
```

Loop Control Statements:

Python provides loop control statements to modify the behavior of loops:

1. **break:** Terminates the loop prematurely, exiting the loop entirely.

```
for i in range(5):
```

```
    if i == 3:
```

```
        break
```

```
    print(i)
```

2. **continue:** Skips the rest of the current iteration and moves to the next iteration.

```
for i in range(5):
```

```
    if i == 3:
```

```
print(i)
```

3. **pass:** Placeholder that does nothing. It is used when a statement is syntactically required but no action is needed.

```
for i in range(5):
```

```
    pass # Placeholder for future code
```

Nested Loops:

You can nest loops within each other to perform more complex iterations, such as iterating over nested lists or performing matrix operations.

Example:

```
for i in range(3):
```

```
    for j in range(2):
```

```
        print(i, j)
```

VII. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Pyhton IDE		

VIII. Resources used (Additional)

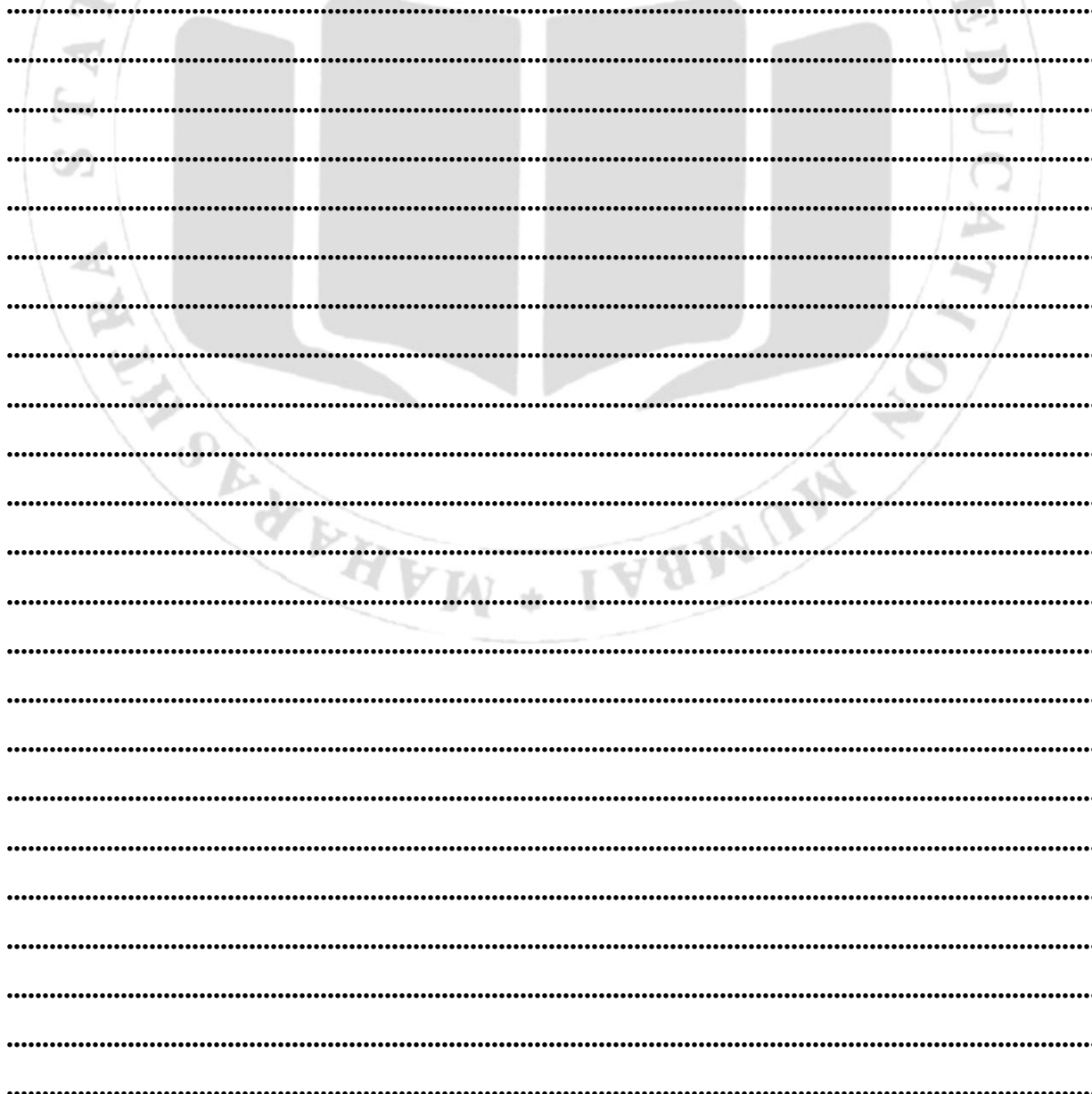
Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related questions

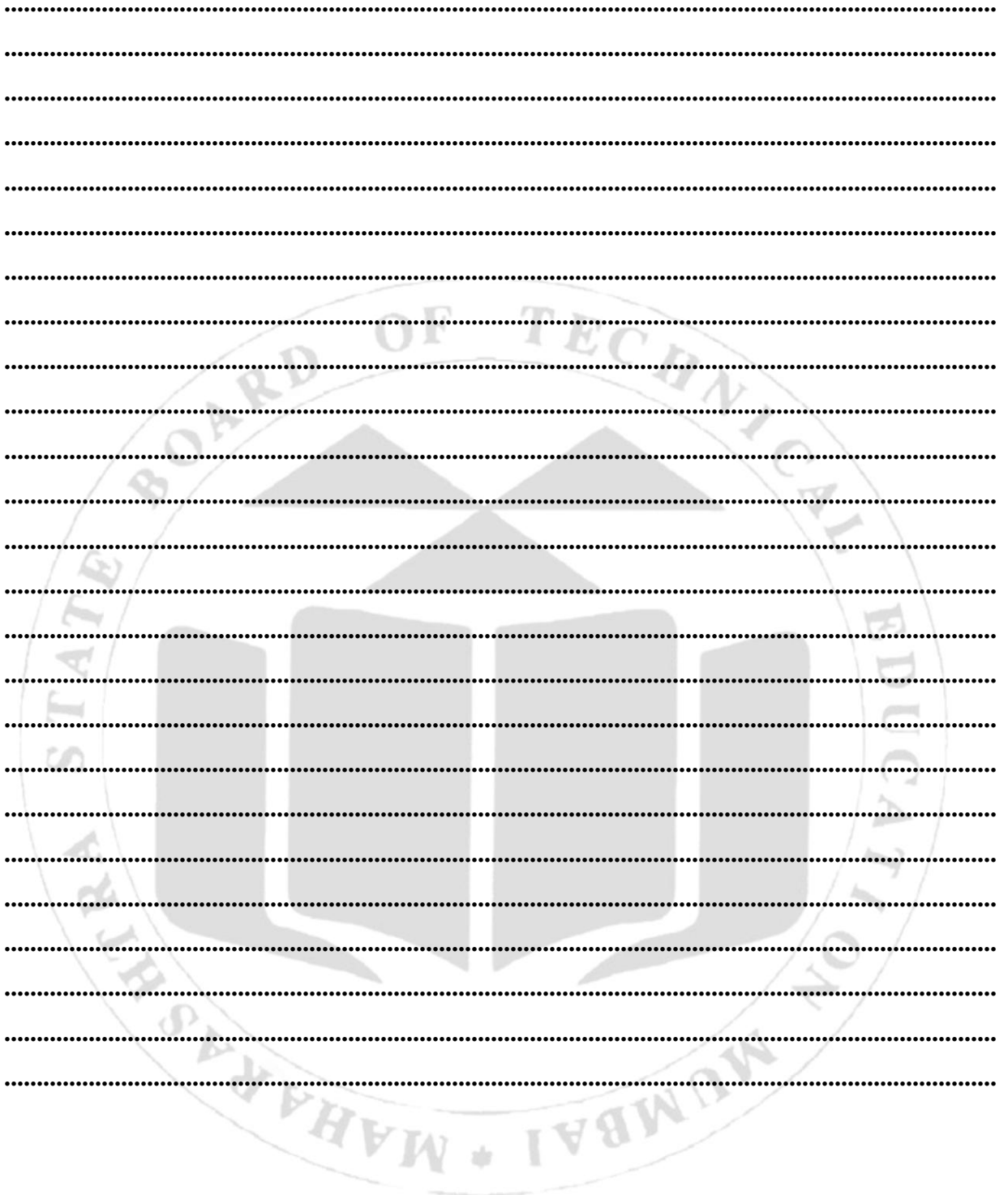
Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write a program to convert a binary number (input as a string) to its decimal equivalent using looping constructs.
2. Implement a program to convert a decimal number (input as an integer) to its binary equivalent using looping constructs.
3. Implement a program to count the number of ones in a binary number (input as a string) using looping constructs.
4. Implement a program to calculate the output voltage of a voltage divider circuit using given resistor values and an input voltage. Use looping constructs to iterate through different resistor combinations and calculate the output voltage.

(Space for answers)



A series of horizontal dotted lines provided for writing the answers to the questions above.



X. Exercise:

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python program to print all even numbers between 1 to 100 using while loop.
2. Write a Python program to find the sum and reverse of n-digit numbers using for loop.
3. Write a Python program to print Fibonacci series on any number entered by the user.
4. Write a Python program to calculate factorial of a number.
5. Write a Python to print the LCM and GCD of two non-negative integer numbers.
6. Write a Python program which will print all the Armstrong numbers between 1 to 500.
7. Write a Python program that takes a number and checks whether it is a palindrome or not.
8. Print the following patterns using loop:

a) *
**

b) *

*

c) 1010101
10101
101
1

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References/Suggestions for further reading

1. https://www.tutorialspoint.com/Python/Python_loops.htm
2. <https://www.youtube.com/watch?v=zFvoXxeosI>
3. <https://www.youtube.com/watch?v=IS7AAV177kA>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.6 *Write Python program to perform following operations on List:

- a) Create
- b) Access
- c) Update
- d) Delete elements from list.

I. Practical significance

Lists in Python are crucial for storing and managing ordered collections of items, providing flexibility with various data types. They support dynamic sizing, allowing for efficient addition and removal of elements. Lists are integral to data processing tasks such as sorting, filtering, and iterating over elements. Their versatility and built-in methods enhance code readability and efficiency in handling complex data operations.

II. Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO3-Perform operations on sequence structures in python.

IV. Laboratory Learning Outcome

LLO 6.1 Perform basic operations on the Lists.

V. Relevant Affective Domain related Outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

In Python, a list is a fundamental data structure that allows you to store and manipulate collections of items. Lists are ordered, mutable (modifiable), and can contain elements of different data types. They are defined by enclosing elements within square brackets [] and separating them with commas.

Here's a detailed explanation of lists in Python with examples:

1. Creating Lists:

You can create a list by simply enclosing items within square brackets. Items can be of any data type, including integers, floats, strings, or even other lists.

```
my_list = [1, 2, 3, 4, 5]
mixed_list = [1, "apple", 3.14, True]
nested_list = [[1, 2], [3, 4], [5, 6]]
empty_list = []
```

2. Accessing Elements:

You can access elements of a list using indexing. Indexing starts from 0 for the first element and goes up to `len(list) - 1` for the last element.

```
my_list = [10, 20, 30, 40, 50]
print(my_list[0]) # Output: 10 (Accessing first element)
print(my_list[-1]) # Output: 50 (Accessing last element using negative indexing)
```

3. Slicing Lists:

You can also access a portion of a list using slicing. Slicing allows you to create a new list containing a subset of the original list.

```
my_list = [10, 20, 30, 40, 50]
print(my_list[1:4]) # Output: [20, 30, 40] (Slice from index 1 to 3)
print(my_list[:3]) # Output: [10, 20, 30] (Slice from start to index 2)
print(my_list[2:]) # Output: [30, 40, 50] (Slice from index 2 to end)
```

4. Modifying Lists:

Lists are mutable, meaning you can change the elements after creating the list. You can modify elements, append new elements, insert elements, or remove elements.

```
my_list = [10, 20, 30, 40, 50]
my_list[1] = 25 # Modify element at index 1
my_list.append(60) # Append 60 to the end of the list
my_list.insert(2, 35) # Insert 35 at index 2
my_list.remove(40) # Remove the element 40
del my_list[0] # Delete element at index 0
```

5. List Operations:

Lists support various operations such as concatenation (+), repetition (*), and checking membership (in).

Basic Python Programming (313011)

```
list1 = [1, 2, 3]
```

```
list2 = [4, 5, 6]
```

```
concatenated_list = list1 + list2 # Concatenate lists
```

```
repeated_list = list1 * 3 # Repeat list 3 times
```

```
is_present = 2 in list1 # Check if 2 is present in list1 (returns True)
```

6. List Methods:

Python provides built-in methods to perform common operations on lists, such as sorting, reversing, counting, and finding elements.

```
my_list = [4, 2, 1, 3]
```

```
my_list.sort() # Sort the list in ascending order
```

```
my_list.reverse() # Reverse the elements of the list
```

```
count_1 = my_list.count(1) # Count occurrences of 1 in the list
```

```
index_3 = my_list.index(3) # Get index of first occurrence of 3
```

7. Nested Lists:

Lists can contain other lists as elements, enabling you to create nested data structures for more complex data representations.

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
print(matrix[1][2]) # Output: 6 (Accessing element in row 1, column 2)
```

Lists are versatile data structures in Python that play a vital role in various programming tasks, including data processing, manipulation, and storage. Their flexibility and wide range of built-in operations make them indispensable for many programming tasks.

VII. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

- 1) Write a Python program that takes a list of numbers as input and prints the sum of all the elements in the list.
- 2) Create a Python program that filters out even numbers from a list of integers and prints the filtered list.
- 3) Write a Python program that reverses a given list of characters and prints the reversed list.
- 4) Write a Python program that duplicates a given list of strings and prints the duplicated list.
- 5) Develop a Python program that takes a list of strings as input and prints the length of the longest string in the list.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....
.....
.....
.....
.....
.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

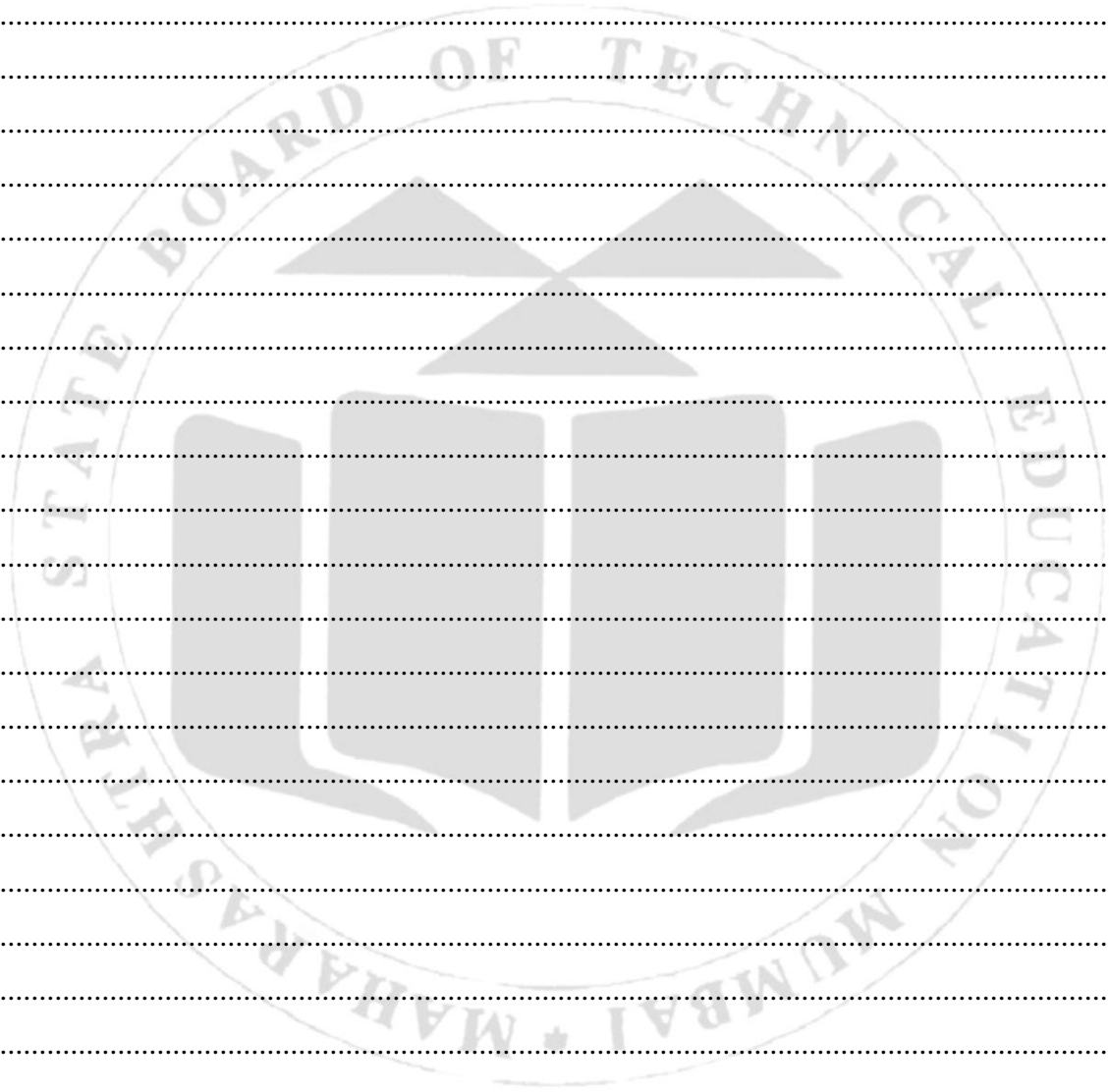
(Use blank space for answers or attach more pages if needed)

- 1) Create a Python program that sorts a list of strings in ascending order and prints the sorted list.
- 2) Write a Python program that takes two lists as input and concatenates them into a single list. Then, print the concatenated list.
- 3) Develop a Python program that searches for a specific element in a list of strings. Print "Element found" if the element exists; otherwise, print "Element not found".
- 4) Develop a Python program that creates a nested list representing a 3x3 matrix of integers. Then, print the matrix.
- 5) Create a Python program that takes a list of integers as input and prints the sublist containing elements from index 2 to index 5.

(Space for answers)

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....



XI. References/Suggestions for further reading

1. https://www.tutorialspoint.com/Python/Python_loops.htm
2. <https://www.youtube.com/watch?v=zFvoXxeosI>
3. <https://www.youtube.com/watch?v=IS7AAV177kA>

XII . Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.7 Develop Python program to perform following operations on Tuples:

- a) Create
- b) Access
- c) Update
- d) Delete Tuple elements

I. Practical significance

Tuples in Python provide immutable data structures, ensuring data integrity and security. They are efficient for storing fixed collections and support multiple assignments for convenient data manipulation. Additionally, tuples can be used as dictionary keys, facilitating efficient data organization and lookup operations.

II. Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO3-Perform operations on sequence structures in python.

IV. Laboratory Learning Outcome

LLO 7.1 Execute various tuple operations.

V. Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

In Python, a tuple is an immutable collection of elements, meaning once created, its contents cannot be modified. Tuples are defined by enclosing elements within parentheses () and separating them with commas. They can contain elements of different data types, including integers, floats, strings, or even other tuples. Here's a detailed explanation with examples:

1. Creating Tuples:

You can create a tuple by simply enclosing elements within parentheses. Tuples can have zero or more elements.

```
empty_tuple = ()
single_element_tuple = (5,)
multiple_elements_tuple = (1, 2, 3, 'a', 'b', 'c')
nested_tuple = ((1, 2), ('a', 'b'))
```

2. Accessing Elements:

You can access elements of a tuple using indexing, similar to lists. Indexing starts from 0 for the first element and goes up to `len(tuple) - 1` for the last element.

```
my_tuple = ('apple', 'banana', 'cherry', 'date')
print(my_tuple[0]) # Output: 'apple' (Accessing first element)
print(my_tuple[-1]) # Output: 'date' (Accessing last element using negative indexing)
```

3. Slicing Tuples:

You can also access a portion of a tuple using slicing, which creates a new tuple containing a subset of the original tuple.

```
my_tuple = ('apple', 'banana', 'cherry', 'date')
print(my_tuple[1:3]) # Output: ('banana', 'cherry') (Slice from index 1 to 2)
print(my_tuple[:2]) # Output: ('apple', 'banana') (Slice from start to index 1)
print(my_tuple[2:]) # Output: ('cherry', 'date') (Slice from index 2 to end)
```

4. Immutable Nature:

Unlike lists, tuples are immutable, meaning their elements cannot be changed or modified after creation. Once a tuple is created, its contents remain fixed.

```
my_tuple = (1, 2, 3)
# Attempting to modify a tuple will result in an error
# my_tuple[0] = 10 # This will raise a TypeError: 'tuple' object does not support item
assignment
```

5. Tuple Packing and Unpacking:

Tuples support multiple assignments, allowing convenient packing and unpacking of data, which simplifies operations like function returns and concurrent processing.

```
# Tuple Packing
my_tuple = 10, 20, 30
# Tuple Unpacking
x, y, z = my_tuple
print(x) # Output: 10
print(y) # Output: 20
print(z) # Output: 30
```

6. Tuples as Dictionary Keys:

Tuples are hashable and can be used as keys in dictionaries, unlike lists, which makes them suitable for dictionary keys when representing composite keys or immutable data.

```
my_dict = {('John', 25): 'Engineer', ('Alice', 30): 'Manager'}
print(my_dict[('John', 25)]) # Output: 'Engineer'
print(my_dict[('Alice', 30)]) # Output: 'Manager'
```

Tuples are versatile data structures in Python, useful for representing fixed collections of elements and ensuring data integrity in scenarios where immutability is required.

VII. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

- 1) Write a Python program that takes a tuple and an additional element as input, then creates a new tuple with the added element. Print the original tuple and the modified tuple.
- 2) Develop a Python program that takes a tuple and an index as input, then replaces the element at the specified index with a new value. Print the original tuple and the modified tuple.
- 3) Create a Python program that takes a tuple and an index as input, then creates a new tuple with the element at the specified index removed. Print the original tuple and the modified tuple.
- 4) Write a Python program that takes a tuple, an index, and a new value as input, then creates a new tuple with the element at the specified index updated with the new value. Print the original tuple and the modified tuple.
- 5) Write a Python program that takes two tuples as input and concatenates them into a single tuple. Then, print the concatenated tuple.
- 6) Write a Python program that takes two tuples of integers as input and calculates the element-wise sum of corresponding elements in the tuples. Then, print the resulting tuple.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....



X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

- 1) Develop a Python program that takes a tuple of (x, y) coordinates as input and unpacks it into separate variables. Then, calculate the distance between the two points using the distance formula.
- 2) Create a Python program that sorts a tuple of strings in alphabetical order and prints the sorted tuple.
- 3) Develop a Python program that searches for a specific element in a tuple of strings. Print "Element found" if the element exists; otherwise, print "Element not found".
- 4) Develop a Python program that checks if a given element exists in a tuple of integers. Print "Element found" if the element exists; otherwise, print "Element not found".
- 5) Write a Python program that compares two tuples element-wise and prints "Equal" if all corresponding elements are equal; otherwise, print "Not equal".
- 6) Create a Python program that takes a tuple of integers as input and repeats each element three times in a new tuple. Then, print the resulting tuple.
- 7) Write a Python program that takes a tuple of strings as input and prints the length of the longest string in the tuple.
- 8) Create a Python program that takes a tuple of strings as input and prints the second element of the tuple.

(Space for answers)

XI. References/Suggestions for further reading

1. https://www.tutorialspoint.com/Python/Python_loops.htm
2. <https://www.youtube.com/watch?v=zFvoXxeosI>
3. <https://www.youtube.com/watch?v=IS7AAV177kA>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.8 Write Python program to perform following operations on Set:

- a) Create
- b) Access
- c) Update
- d) Delete Access Set elements

I. Practical significance

Sets in Python provide an efficient way to store and manipulate unique elements, automatically eliminating duplicates. They support fast membership testing, making them ideal for tasks like filtering out repeated items and performing set operations like union, intersection, and difference. This makes sets particularly useful for handling collections of data where uniqueness and efficient lookups are crucial.

II. Industry / Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO3- Perform operations on sequence structures in python.

IV. Laboratory Learning Outcome

LLO 8.1 Implement various set operations.

V. Relevant Affective Domain related outcome(s)

1. Follow safety practices.
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices.

VI. Relevant Theoretical Background

A set is an unordered collection of unique elements. This means that no element can appear more than once in a set. Sets are mutable, meaning you can add or remove elements from them after they are created. They are particularly useful for membership testing, removing duplicates from a sequence, and performing mathematical set operations like union, intersection, difference, and symmetric difference.

1. Creating a Set

You can create a set by placing all the items (elements) inside curly braces {}, separated by commas, or by using the built-in set() function.

```
# Creating a set using curly braces
```

```
my_set = {1, 2, 3, 4, 5}
```

```
print("Set:", my_set)
```

```
# Creating a set using the set() function
```

```
my_set = set([1, 2, 3, 4, 5])
```

```
print("Set:", my_set)
```

```
# Creating an empty set
```

```
empty_set = set()
```

```
print("Empty Set:", empty_set)
```

2. Adding Elements to a Set

You can add elements to a set using the add() method or update() method.

```
my_set = {1, 2, 3}
```

```
print("Original Set:", my_set)
```

```
# Adding a single element
```

```
my_set.add(4)
```

```
print("Set after adding 4:", my_set)
```

```
# Adding multiple elements
```

```
my_set.update([5, 6])
```

```
print("Set after adding 5 and 6:", my_set)
```

3. Removing Elements from a Set

You can remove elements using the remove() method, discard() method, or pop() method. The remove() method will raise a KeyError if the element is not found, while the discard() method will not.

```
my_set = {1, 2, 3, 4, 5}
```

```
print("Original Set:", my_set)
```

```
# Removing an element
my_set.remove(4)
print("Set after removing 4:", my_set)

# Discarding an element
my_set.discard(3)
print("Set after discarding 3:", my_set)

# Removing an element using pop (removes an arbitrary element)
popped_element = my_set.pop()
print("Popped Element:", popped_element)
print("Set after popping an element:", my_set)
```

4. Set Operations

Sets support various mathematical operations.

Union: Combines elements from both sets (all unique elements).

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
print("Union:", union_set)
```

Intersection: Returns elements that are common in both sets.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersection_set = set1.intersection(set2)
print("Intersection:", intersection_set)
```

Difference: Returns elements that are in the first set but not in the second.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
difference_set = set1.difference(set2)
print("Difference:", difference_set)
```

Symmetric Difference: Returns elements that are in either of the sets, but not in both.

```
set1 = {1, 2, 3}
```

Basic Python Programming (313011)

```
set2 = {3, 4, 5}
sym_diff_set = set1.symmetric_difference(set2)
print("Symmetric Difference:", sym_diff_set)
```

Membership Testing : You can test if an element is a member of a set using the in keyword.

```
my_set = {1, 2, 3}
print(1 in my_set) # Output: True
print(4 in my_set) # Output: False
```

Sets are a powerful feature in Python that help in handling collections of unique elements and performing efficient membership tests and set operations.

VII. Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Create a Python program that takes two sets of numbers as input and returns a new set that is the union of the two sets.
2. Write a Python program that takes two sets of numbers as input and returns a new set that contains elements that are in the first set but not in the second set.
3. Develop a Python program that takes two sets of numbers as input and returns a new set that contains elements that are in either of the sets but not in both.
4. Create a Python program that checks if a given element is present in a set. Print "Element found" if the element exists in the set; otherwise, print "Element not found".
5. Write a Python program that checks if one set is a subset of another set. Print "Subset" if it is a subset; otherwise, print

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python program that takes a sentence and returns a set of unique words.
2. Write a Python program that checks if a given element is present in a set.
3. Write a Python program that takes multiple sets and print their intersection.
4. Write a Python program that generates a set of squares of all even numbers between 1 and 10.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

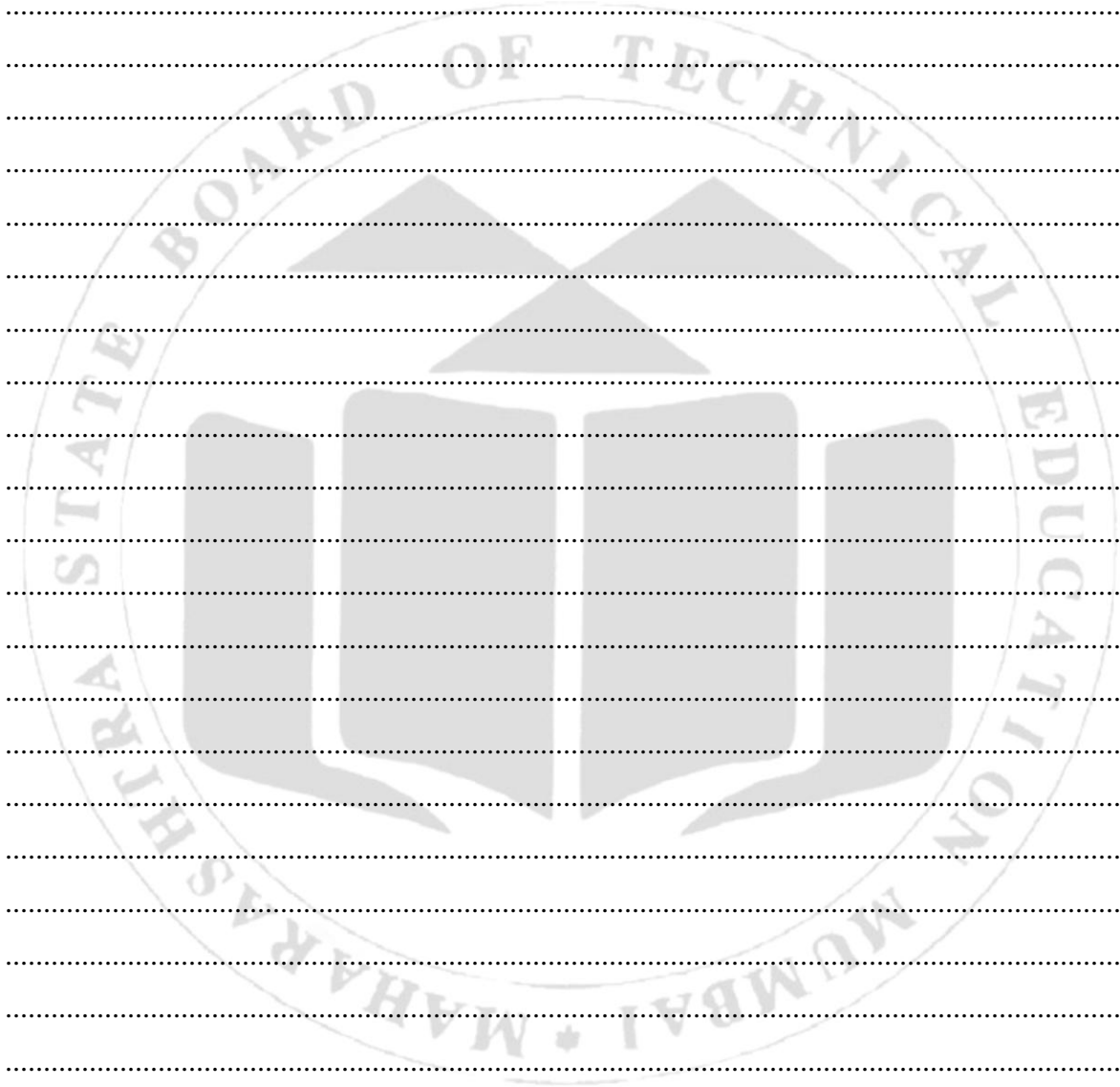
.....

.....

.....

.....

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....



.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References/Suggestions for further reading

1. https://www.tutorialspoint.com/Python/Python_loops.htm
2. <https://www.youtube.com/watch?v=zFvoXxeoosI>
3. <https://www.youtube.com/watch?v=IS7AAV177kA>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.9: *Create a program to perform following operations on Dictionaries in Python:

- a) Create
- b) Access
- c) Update
- d) Delete
- e) Looping through Dictionary

I. Practical Significance

Python dictionaries allow us to associate a value to a unique key, and then to quickly access this value. Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered, changeable and does not allow duplicates.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO3 - Perform operations on sequence structures in python.

IV. Laboratory Learning Outcome

LLO 9.1 Execute various operations on Dictionaries.

V. Relevant Affective Domain related outcome(s)

- a) Follow safety practices.
- b) Demonstrate working as a leader / a team member.
- c) Follow ethical practices.

VI. Relevant Theoretical Background

Python dictionary is a container of key-value pairs. It is mutable and can contain mixed types. A dictionary is an unordered collection. Python dictionaries are called associative arrays or hash tables in other languages. The keys in a dictionary must be immutable objects like strings or numbers. They must also be unique within a dictionary.

a) Creating the Dictionary

The most common method is to use curly braces {} separating the keys and values by colons. For example, the following example creates a dictionary with three items:

The syntax to define the dictionary is given below:

```
dict = {"name": "John", "age": 30, "city": "New York"}
```

b) Accessing Values in Dictionary:

To access dictionary elements, use square brackets along with the key to obtain its value. To extract values, use the following format:

```
dictionary[key]
```

Example:

```
dict = {"name": "John", "age": 30, "city": "New York"}  
name = dict["name"] # Get value assigned to the "name" key  
print(name)
```

Output:

John

c) **Updating Dictionary:** You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry.

Example 1:

```
dict = {"name": "John", "age": 30, "city": "New York"}  
print(dict) # Before modification  
dict["age"] = 35  
print(dict) # After modification
```

Output Before: {'name': 'John', 'age': 30, 'city': 'New York'}

Output After: {'name': 'John', 'age': 35, 'city': 'New York'}

Example 2:

```
dict = {"name": "John", "age": 30, "city": "New York"}  
dict["gender"] = "male" # Add new key-value pair  
print(dict)
```

Output:

```
{"name": "John", "age": 30, "city": "New York", "gender": "male"}
```


d) Creating the Dictionary

The most common method is to use curly braces {} separating the keys and values by colons. For example, the following example creates a dictionary with three items:

The syntax to define the dictionary is given below:

e) Creating the Dictionary

The most common method is to use curly braces {} separating the keys and values by colons. For example, the following example creates a dictionary with three items:

The syntax to define the dictionary is given below:

```
dict = {"name": "John", "age": 30, "city": "New York"}
```

f) Accessing Values in Dictionary:

To access dictionary elements, use square brackets along with the key to obtain its value. To extract values, use the following format:

```
dictionary[key]
```

Example:

```
dict = {"name": "John", "age": 30, "city": "New York"}  
name = dict["name"] # Get value assigned to the "name" key  
print(name)
```

Output:

John

g) Updating Dictionary: You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry.

Example 1:

```
dict = {"name": "John", "age": 30, "city": "New York"}  
print(dict) # Before modification  
dict["age"] = 35  
print(dict) # After modification
```

Output Before: {'name': 'John', 'age': 30, 'city': 'New York'}

Output After: {'name': 'John', 'age': 35, 'city': 'New York'}

Example 2:

```
dict = {"name": "John", "age": 30, "city": "New York"}  
dict["gender"] = "male" # Add new key-value pair  
print(dict)
```

Output:

```
{"name": "John", "age": 30, "city": "New York", "gender": "male" }
```

h) Delete Dictionary Elements:

To remove individual dictionary elements or clear the entire contents of a dictionary or to delete entire dictionary in a single operation or to explicitly remove an entire dictionary, just use the del statement.

Example:

```
dict = {"name": "John", "age": 30, "city": "New York"}
del dict['name']; # remove entry with key 'name'
```

Output:

```
{"age": 30, "city": "New York" }
```

```
dict.clear(); # remove all entries in dict
```

```
del dict ; # delete entire dictionary
```

i) Looping through Dictionary:

A dictionary can be iterated using the for loop. If you want to get both keys and the values in the output. You just have to add the keys and values as the argument of the print statement in comma separation. After each iteration of the for loop, you will get both the keys its relevant values in the output.

Example

```
dict = {"name": "John", "age": 30, "city": "New York"}
for key, value in dict.items():
    print(key, ' - ', value)
```

Output:

```
name - John age - 30 city - New York
```

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

```
dict = {1:"Photodiode", 2:"Zener diode", 3:" Schottky diode",4: "P-N junction diode" }
del dict[1];
for key, val in dict.items():
    print(key)
for key, val in dict.items():
    print(key,values)
del dictionary:
for key, val in dict.items():
    print(key)
```

2. Write the output of the following program.

```
dict1 = {'Amplifiers': 'BJT', 'Transistor': 'PNP'}
dict2 = {'FET':10, 'MOSFET': 5}
dict1.update(dict2)
print(dict1)
```

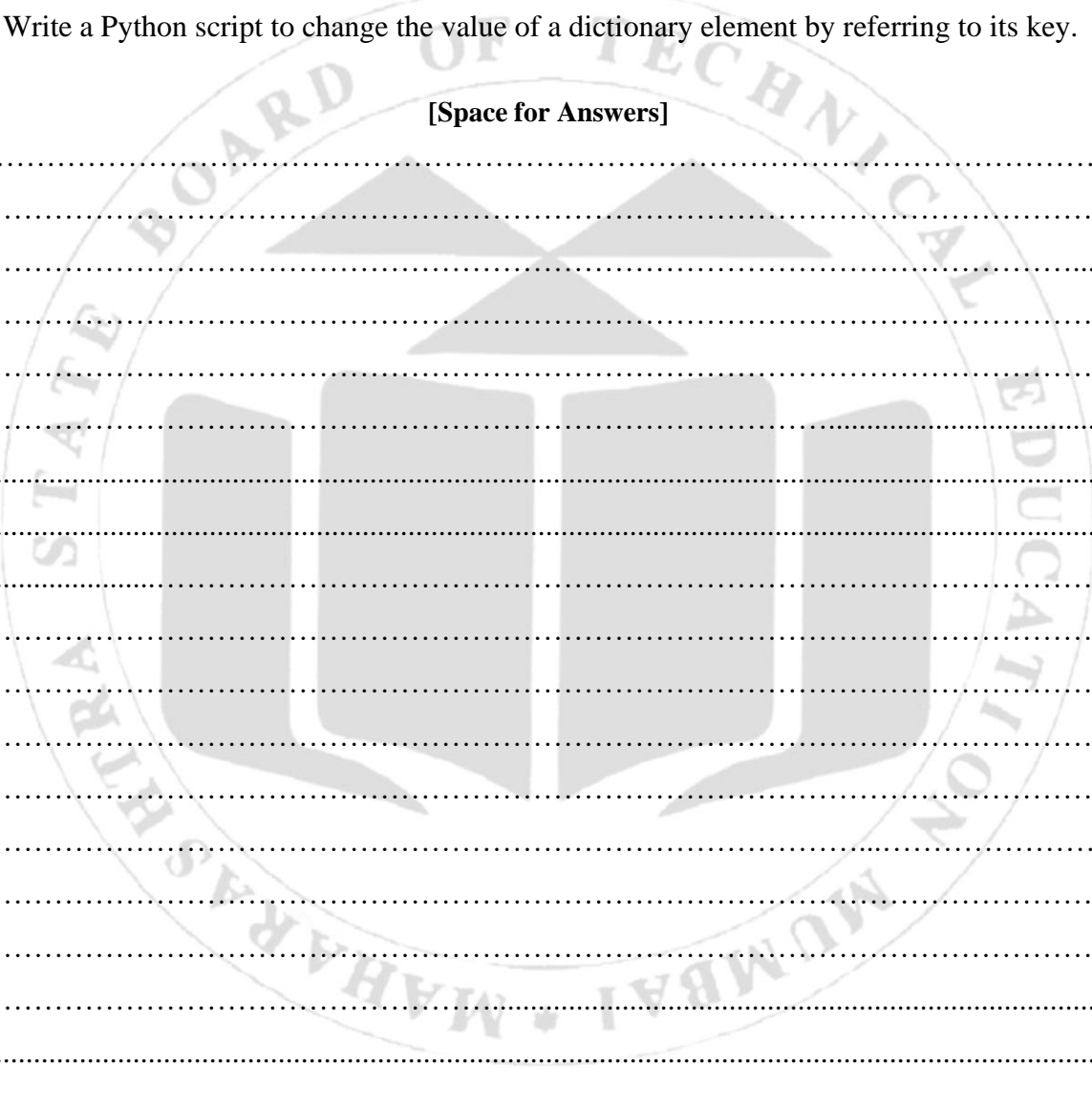
X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python script to iterate over dictionaries using for loops.
2. Write a Python script to change the value of a dictionary element by referring to its key.

[Space for Answers]



A large watermark of the Maharashtra State Board of Technical Education logo is centered on the page. The logo is circular and contains the text 'MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION' and 'MUMBAI'. In the center of the logo is a stylized emblem featuring a book and a lamp. Below the watermark, the page is filled with horizontal dotted lines for writing answers.

XI. References/Suggestions for further reading

1. <https://www.w3resource.com>
2. https://www.tutorialspoint.com/Python/Python_dictionary.htm
3. <https://www.javatpoint.com/Python-dictionary>
4. https://www.youtube.com/watch?v=fAw8pM_dQP4&t=1598s
5. <https://www.youtube.com/watch?v=daefaLgNkw0>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related: 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.10: *a) Create python program to demonstrate use of math built-in function.

***b) Create python program to demonstrate use of string built-in function.**

I. Practical Significance

Python includes multiple math-related and string-related functions. The math built-in module is used to perform various mathematical calculations, such as numeric, trigonometric, logarithmic, and exponential calculations. The Python string module provides several constants that are useful for checking to see if a character, slice, or string contains letters, digits, symbols, etc. To use these modules, we need to import the module into our code.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO4 – Implement basics of object oriented programming concepts.

IV. Laboratory Learning Outcome

LLO 10.1 Use built-in mathematical functions and string functions in python.

V. Relevant Affective Domain related outcome(s)

- a) Follow safety practices.
- b) Demonstrate working as a leader / a team member.
- c) Follow ethical practices.

VI. Relevant Theoretical Background

a) List of Math's built-in Functions and its attributes:

Function	Description
ceil(x)	Return the Ceiling value. It is the smallest integer, greater or equal.
copysign(x, y)	Returns x with the sign of y.
cos(x)	Return the cosine of x in radians.
exp(x)	Returns e**x.

Basic Python Programming(313011)

factorial(x)	Returns the factorial of x.
floor(x)	Returns the largest integer less than or equal to x.
fmod(x, y)	Returns the remainder when x is divided by y.
gcd(x, y)	Returns the Greatest Common Divisor of x and y.
log(x[, base])	Returns the Log of x, where base is given. The default base is e.
log2(x)	Returns the Log of x, where base is 2.
log10(x)	Returns the Log of x, where base is 10.
pow(x, y)	Return the x to the power y value.
remainder(x, y)	Find remainder after dividing x by y.
radians(x)	Convert angle x from degrees to radian.
sqrt(x)	Finds the square root of x.
sin(x)	Return the sine of x in radians.
tan(x)	Return the tangent of x in radians.

b) String built-in functions:

Python includes the following built-in methods to manipulate strings:

Function	Description
capitalize()	Converts the first character to upper case
center()	Pad the string with the specified character.
count()	Returns the number of occurrences of a substring in the string.
encode()	Encodes strings with the specified encoded scheme
find()	Returns the lowest index of the substring if it is found
format()	Formats the string for printing it to console
index()	Returns the position of the first occurrence of a substring in a string
isalnum()	Checks whether all the characters in a given string is alphanumeric or not
isalpha()	Returns "True" if all characters in the string are alphabets
isdecimal()	Returns true if all characters in a string are decimal
isdigit()	Returns "True" if all characters in the string are digits
isidentifier()	Check whether a string is a valid identifier or not
islower()	Checks if all characters in the string are lowercase
isnumeric()	Returns "True" if all characters in the string are numeric characters
isdigit()	Returns True if all characters in the string are digits
lower()	Converts a string into lower case
replace()	Returns a string where a specified value is replaced with a specified
upper()	Converts a string into upper case
title()	Converts the first character of each word to upper case

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

```
import math

x = math.ceil(1.4)
y = math.floor(1.4)

print(x)
print(y)
```

2. Write the output of the following program.

```
for i in range (3,12,2):
    print("a".upper())
```

3. The function pow(x,y,z) is evaluated as:

- a) $(x**y)**z$
- b) $(x**y) / z$
- c) $(x**y) \% z$

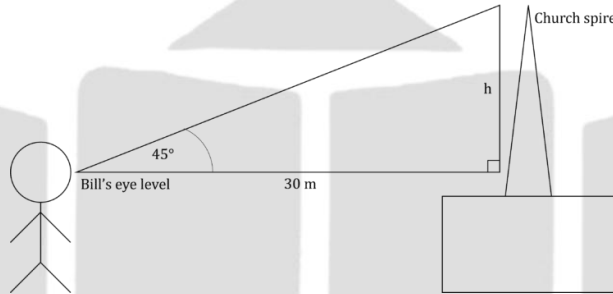
.....
.....
.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

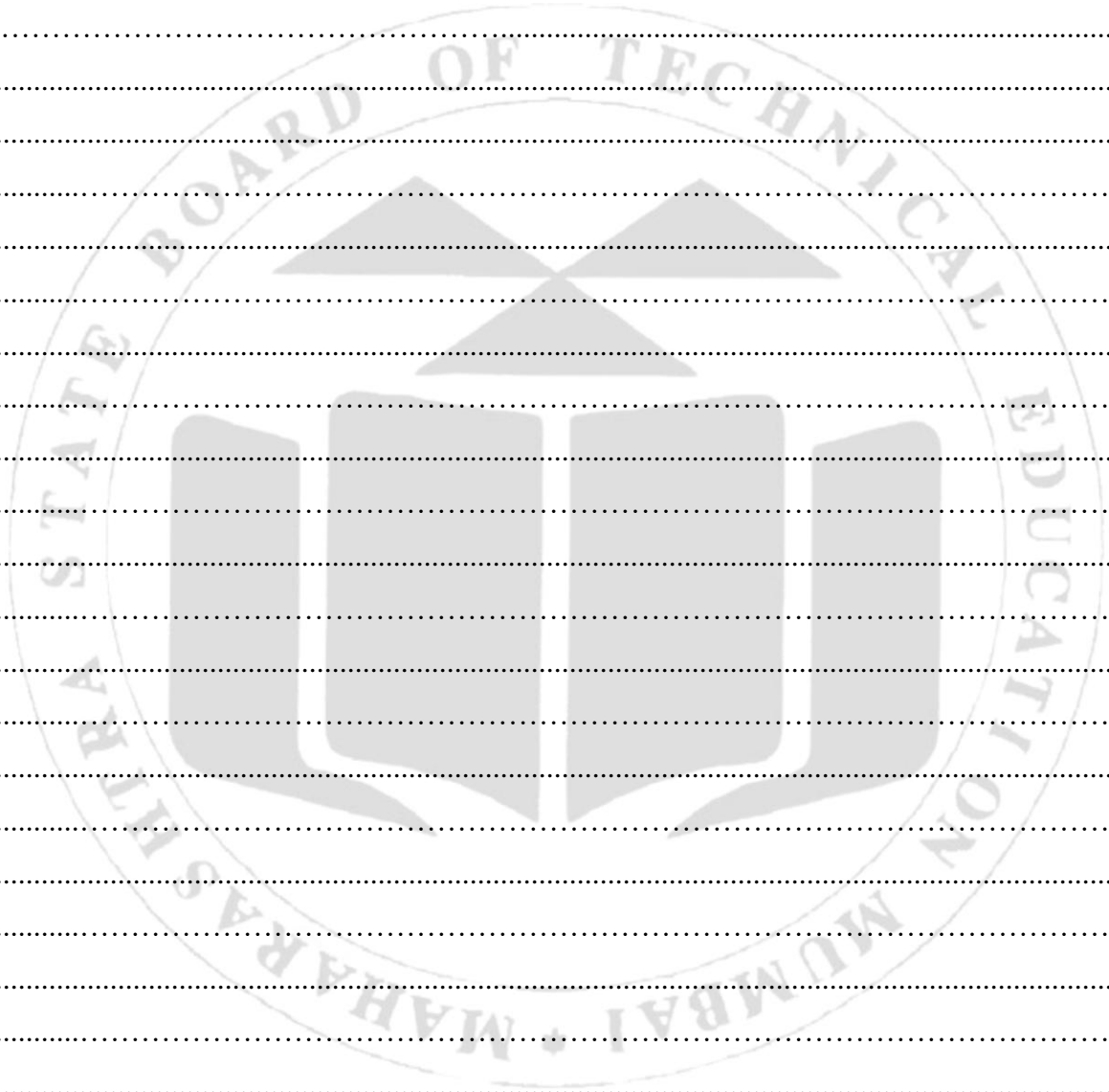
1. Write a Python script to calculate factorial of a number using math built-in function.
2. Bill is 30 m away from a church. The angle of elevation when he looks at the top of the church's spire is 45° . If Bill's eye level is 1.5 m above the ground, how tall is the church spire. (Use math built-in function)



3. Write a Python script that accepts a string in lowercase letters and convert into upper case letters using string module.
4. Write a Python script that uses find() function to search for the “language” string from the original string "Python is a powerful language" and returns its index in the Python string.

[Space for Answers]

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....



XI. References/Suggestions for further reading

1. https://www.w3schools.com/python/python_ref_string.asp
2. <https://www.w3resource.com/python-exercises/math/>
3. <https://www.toppr.com/guides/computer-science/introduction-to-python/functions/functions-from-math-module/>
4. <https://www.youtube.com/watch?v=vrCpsUMh368>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.11 Write python programs to define function with arguments.

a) Calculate factorial of a number

b) Swapping of two variables

I. Practical Significance

Functions help to decompose a large program into small segments which makes program easy to understand, maintain and debug. If repeated code occurs in a program. Function can be used to include those codes and execute when needed by calling that function. This practical will make learner reuse code instead of rewriting the same code multiple times.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO4 - Implement basics of object oriented programming concepts.

IV. Laboratory Learning Outcome

LLO 11.1 Create user defined functions in Python.

V. Relevant Affective Domain related outcome(s)

- a) Follow safety practices.
- b) Demonstrate working as a leader / a team member.
- c) Follow ethical practices.

VI. Relevant Theoretical Background

Functions are useful in programming because they eliminate needless and excessive copying and pasting of code throughout a program. Functions also help organize your code.

How to Define a Function in Python:

Syntax:

```
def function_name(parameters):  
    function body
```

Example:

```
def hello_world_func():  
    print("hello world")
```

Python Function Arguments:

The data/information in the functions can be passed as the arguments or parameters. The arguments are specified in the parentheses and are separated by commas.

Example:

```
def add_numbers(a, b): # Defining a function  
    sum = a + b  
    print('Sum:', sum)  
  
add_numbers(2, 3) # Calling a function
```

return Statement: A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller.

Example:

```
def add_numbers (a,b): # function definition  
    result = a+b  
    return result  
  
addition = add_numbers (3,6) # function call  
print ('Sum of two numbers:', addition)
```

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

```
def find_square(num):
    result = num * num
    return result
```

```
square = find_square(3)
```

```
print('Square:', square)
```

2. Write error and give justification for the same.

```
def Interest(P,R,T=7):
```

```
    I = (P*R*T)/100
    print(I)
```

```
Interest(20000,.08,15) #Line 1
```

```
Interest(T=10,20000,.075) #Line 2
```

```
Interest(50000,.07) #Line 3
```

```
Interest(P=10000,R=.06,Time=8) #Line 4
```

```
Interest(80000,T=10) #Line 5
```

.....
.....
.....

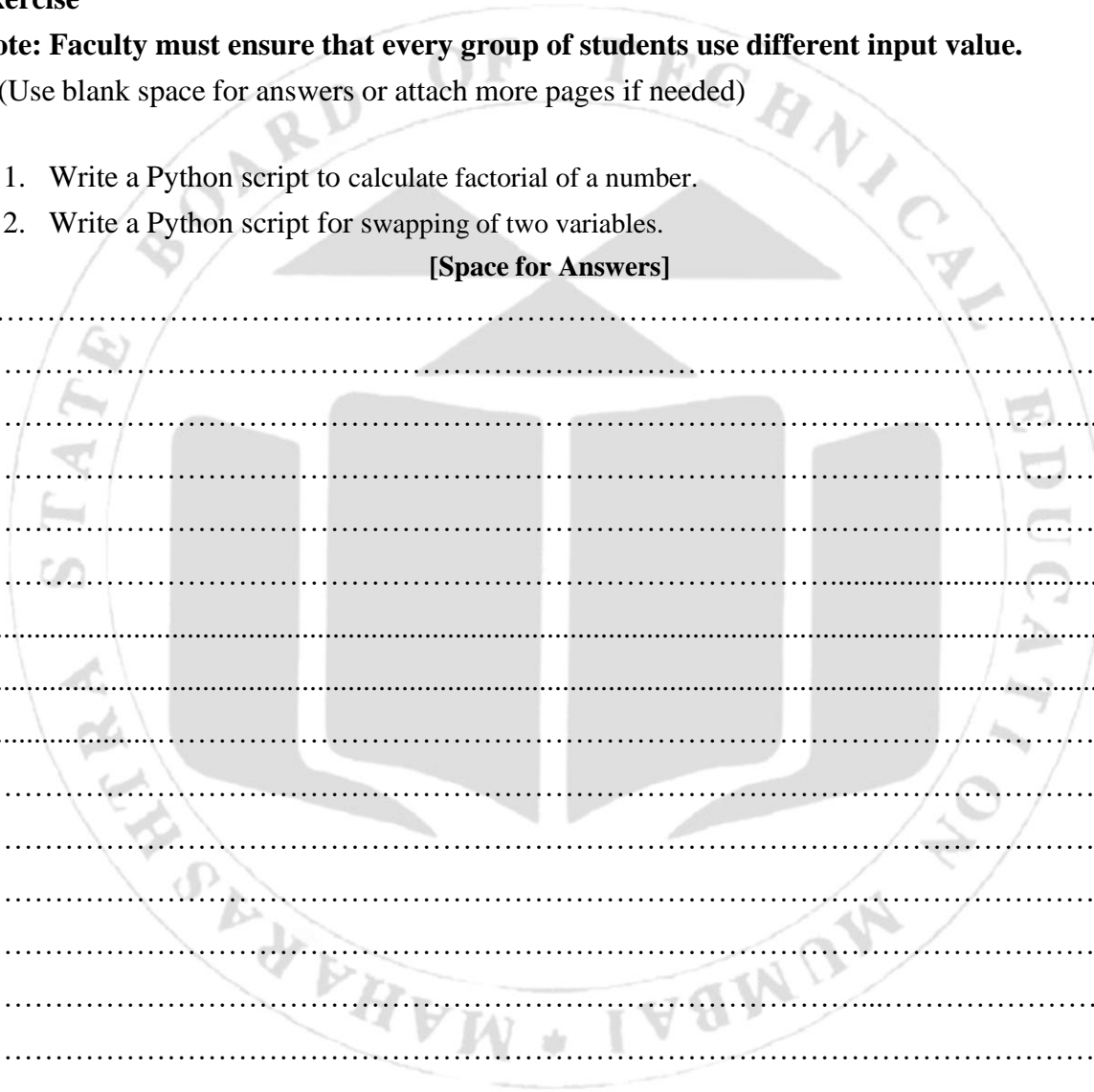
X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python script to calculate factorial of a number.
2. Write a Python script for swapping of two variables.

[Space for Answers]



.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XI. References/Suggestions for further reading

1. https://www.w3schools.com/python/python_functions.asp
2. <https://www.geeksforgeeks.org/>
3. <https://www.freecodecamp.org/>
4. https://www.youtube.com/watch?v=u-OmVr_fT4s&t=1s

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.12 Write programs to define function with default arguments.

I. Practical Significance

Default function parameters allow named parameters to be initialized with default values if no value or undefined is passed. Default arguments in Python represent the **function arguments** that will be used if no arguments are passed to the function call.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO4 - Implement basics of object oriented programming concepts.

IV. Laboratory Learning Outcome

LLO 12.1 Implement function with default arguments.

V. Relevant Affective Domain related outcome(s)

- d) Follow safety practices.
- e) Demonstrate working as a leader / a team member.
- f) Follow ethical practices.

VI. Relevant Theoretical Background

Default arguments in a function are parameters with a predefined value set by the function definition. When calling a function, if no argument is provided for a parameter with a default value, the default value will be used. In this way, the function is more flexible and does not require each argument to be specified.

Syntax:

```
def function_name(parameter1=default_value1, parameter2=default_value2,
...):# Function body
# The code that uses the parameters
# A parameter's default value will be used if no value is provided
```

Example:

```
def function(a = 24, b,
c):print('Values
are:') print(a)
print
(b)
print
(c)
function(10, 4)
```

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

```
def student(firstname, lastname='Mark', standard='Fifth'):
    print(firstname, lastname, 'studies in', standard, 'Standard')
```

```
student("John")
student("Jane", "Smith")
student("Bob", "Johnson", "Sixth")
```

2. Write the errors and give justification for the same.

```
X = 100
def Change(P=10,Q=25):
    global X
    if P%6==0:
        X+=100
    else:
        X+=50
    Sum=P+Q+X
    print(P,'#',Q,'$',Sum)
Change()
Change(18,50)
Change(30,100)
```

.....
.....
.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python script that accepts price and discount rates as arguments in a function which returns the new price based on the discount rate. The default discount rate is 3%.
2. Write a Python script to calculate Gross Salary of Employee by taking salary as a function that accepts Basic Salary,TA,DA,HRA as arguments(use default value of TA =4% and DA=8%).

[Space for Answers]

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XI. References/Suggestions for further reading

1. https://www.w3schools.com/python/python_functions.asp
2. <https://www.geeksforgeeks.org/>
3. <https://www.freecodecamp.org/>
4. https://www.youtube.com/watch?v=u-OmVr_fT4s&t=1s

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.13: *Create a program to demonstrate use of: Built-in module (e.g. numeric, mathematical functional and programming module) in Python.

I. Practical Significance

Python's built-in modules for file handling and input/output (I/O) operations offer a wide range of tools and functions for reading, writing, and manipulating files and data streams. These modules are essential for working with files, whether for data processing, text manipulation, or data storage.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO5 - Create modules and packages for given purpose.

IV. Laboratory Learning Outcome

LLO 13.1 Use built-in python mathematical modules.

V. Relevant Affective Domain related outcome(s)

- a) Follow safety practices.
- b) Demonstrate working as a leader / a team member.
- c) Follow ethical practices.

VI. Relevant Theoretical Background

- There are several built-in modules in Python, which you can import whenever you like. A module is a collection of Python objects such as functions, classes, and so on. Python interpreter is bundled with a standard library consisting of large number of built-in modules.
- Built-in modules are generally written in C and bundled with Python interpreter in precompiled form. A built-in module may be a Python script (with .py extension) containing useful utilities.
- A module may contain one or more functions, classes, variables, constants, or any other Python resources.

I. Numeric and Mathematical Modules:

This module provides numeric and math-related functions and data types.

Following are the modules which are classified as numeric and mathematical modules.

- (i) numbers (Numeric abstract base classes).
- (ii) math (Mathematical functions).
- (iii) cmath (Mathematical functions for complex numbers).
- (iv) decimal (Decimal fixed point and floating point arithmetic).
- (v) fractions (Rational numbers).
- (vi) random (Generate pseudo-random numbers).
- (vii) statistics (Mathematical statistics functions).

II. Functional Programming Modules:

• These modules provide functions and classes that support a functional programming style and general operations on callables.

Following are modules which comes under functional programming modules.

- (i) itertools (functions creating iterators for efficient looping).
- (ii) functools (higher-order functions and operations on callable objects).
- (iii) operator (standard operators as functions).

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

```
import math
def circle_area(r):
    return math.pi*r**2
radius = 3
```

```
print("Area =", circle_area(radius))
```

2. Write the output of the following program.

```
import fractions
for deci in ['0.6', '2.5', '2.3', '4e-1']:
    fract = fractions.Fraction(deci)
    print(fract)
```

.....

.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python script to calculate the DC output voltage (V_{dc}) of Half wave rectifier using Formula: $V_{dc} = (V_m / \pi)$ where V_m is the peak value of output voltage using math module.
2. Write a Python script to calculate the Area of cross section of the wire, $A = \pi * d^2 / 4 =$ _____ m^2 where d is the diameter of wire.

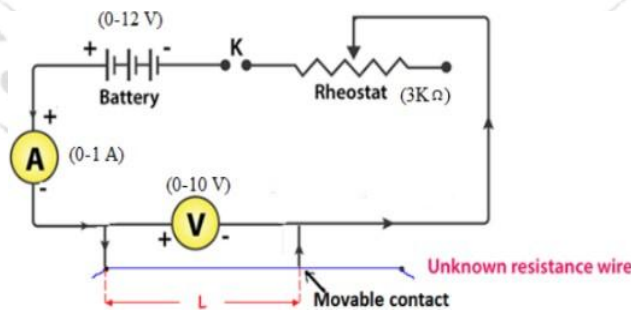
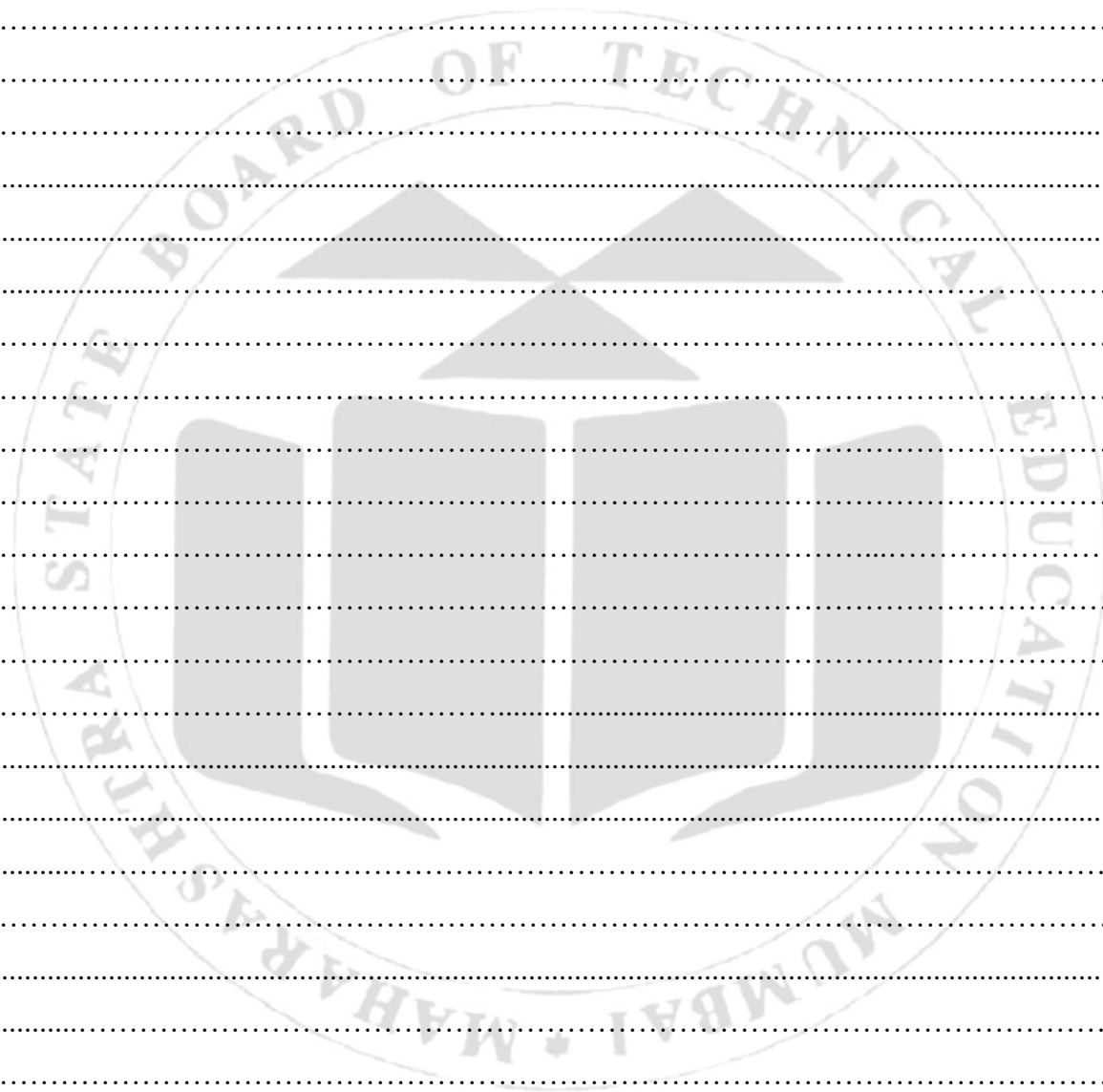


Figure 1.1: Determination of resistivity

3. Write a Python Script by using statistics module to calculate mean (average value), median (middle value), mode (most often value), standard deviation (spread of values).

[Space for Answers]



XI. References/Suggestions for further reading

1. <https://www.javatpoint.com/python-math-module>
2. <https://docs.python.org/3/library/math.html>
3. https://www.w3schools.com/python/module_math.asp
4. <https://www.youtube.com/watch?v=2H00FMtANI>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.14 Write program to create a user-defined module (e.g.: building calculator)in python.

I. Practical Significance

Python modules assist us in organizing and grouping content through the use of files and folders. Python modules come into play in this modular programming method, where we have separated the code into discrete components. User-defined modules in Python are those modules that are created by the users to make their projects easy and simple.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO5 - Create modules and packages for given purpose.

IV. Laboratory Learning Outcome

LLO 14.1 Write user-defined module in python.

V. Relevant Affective Domain related outcome(s)

- d) Follow safety practices
- e) Demonstrate working as a leader / a team member.
- f) Follow ethical practices

VI. Relevant Theoretical Background

Modules in Python are simply files with the “.py” extension that contain Python code such as Python functions, methods, classes, or variables that may be imported into another Python program. A Python module is a file that contains Python commands and definitions. A Python module can be imported by another Python script using the **import** statement. Once a module is imported, its contents can be accessed using the dot notation (e.g., *module_name.function_name*). This allows the functionality defined in the module to be used in other parts of the program.

Basic of Python Programming (313011)

For example, let's say you want to create a module called *my_module.py* that contains a function called *square()* that calculates the square of a number. You could create this module by creating a new file called *my_module.py* and adding the following code to it:

```
def square(x):
```

```
    return x*x
```

Once you have created the module file, you can use it in another Python script by importing it using the *import* statement:

```
import my_module
```

```
result = my_module.square(5)
```

```
print(result)
```

Output:

```
25
```

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

Save this code in the file mymodule.py

```
person1 = {  
    "name": "John",  
    "age": 36,  
    "country": "Norway"  
}  
  
import mymodule  
  
a = mymodule.person1["age"]  
print(a)
```

2. Tick the correct statement which correctly describes a module.

- a) Denoted by triple quotes for providing the specification of certain program elements
- b) Design and implementation of specific functionality to be incorporated into a program
- c) Defines the specification of how it is to be used
- d) Any program that reuses code

3. Tick the statement which is not an advantage of using modules.

- a) Provides a means of reuse of program code
- b) Provides a means of dividing up tasks
- c) Provides a means of reducing the size of the program
- d) Provides a means of testing individual parts of the program

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

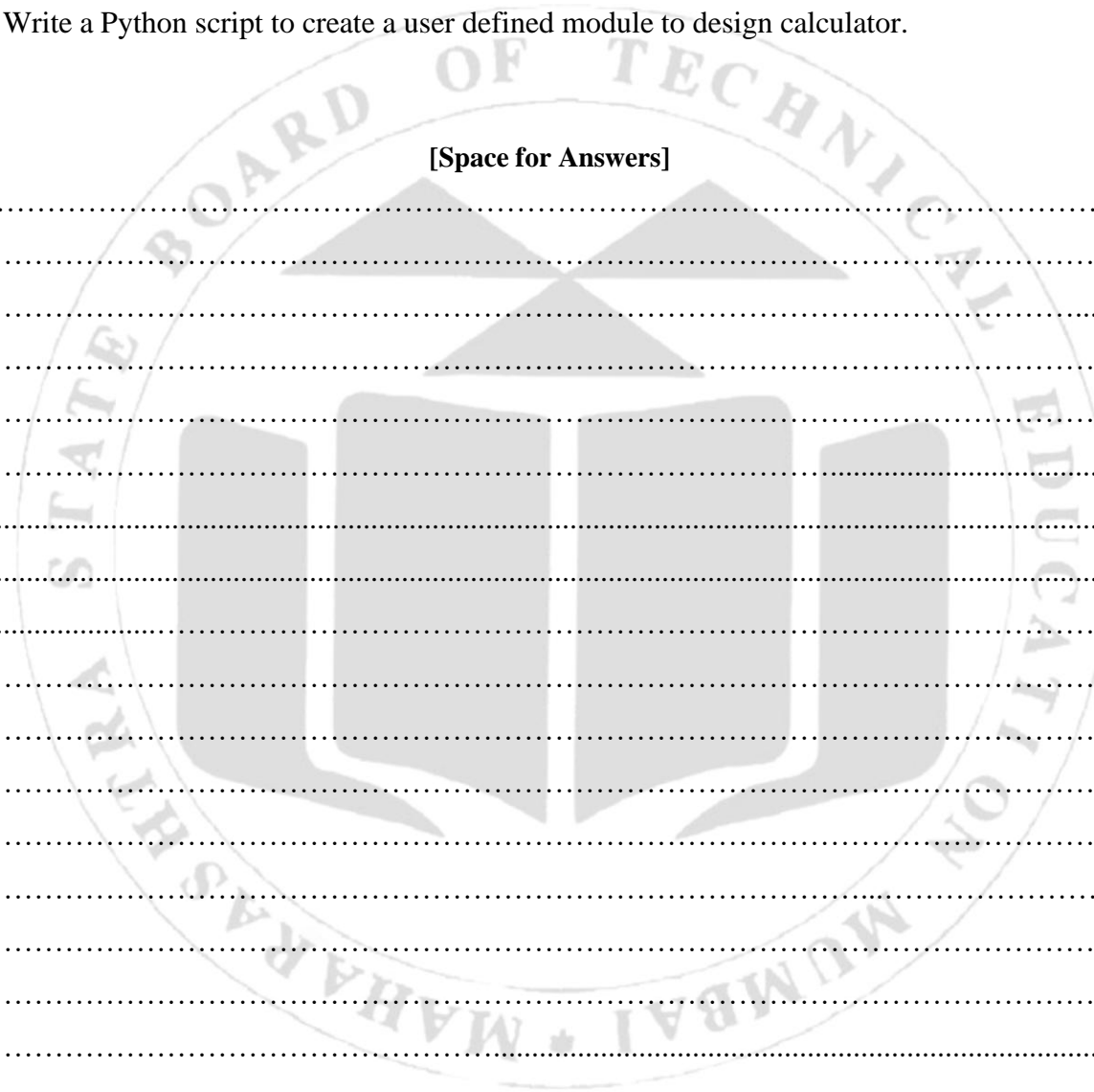
X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python script to create a user defined module that will ask personal details of an employee and will display the details.
2. Write a Python script to create a user defined module to design calculator.

[Space for Answers]



A large, faint watermark of the Maharashtra State Board of Technical Education logo is centered on the page. The logo is circular and contains the text 'MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION' and 'MUMBAI * MAHARASHTRA'. In the center of the logo is a stylized emblem featuring a book and a lamp. Below the watermark, the page is filled with horizontal dotted lines for writing answers.

XI. References/Suggestions for further reading

1. <https://www.toppr.com/guides/python-guide/references/methods-and-functions/python-modules/>
2. <https://www.programiz.com/python-programming/modules>
3. https://www.tutorialspoint.com/python/python_modules.htm
4. https://www.youtube.com/watch?v=M_Pa1fZWY8Y

XII. Assessment Scheme

Performance Indicators		Weight age
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.15: *Develop Python program to demonstrate use of NumPy package for creating, accessing and performing different array operations.

I. Practical Significance

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:

Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO5 - Create modules and packages for given purpose.

IV. Laboratory Learning Outcome

LLO 15.1 Use python built-in packages.

V. Relevant Affective Domain related outcome(s)

- a) Follow safety practices.
- b) Demonstrate working as a leader / a team member.
- c) Follow ethical practices.

VI. Relevant Theoretical Background

NumPy stands for "Numerical Python". It provides a high-performance multidimensional array object, and tools for working with these arrays.

An array is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers and represented by a single variable. NumPy's array class is called ndarray. It is also known by the alias array.

In NumPy arrays, the individual data items are called elements. All elements of an array should be of the same type. Arrays can be made up of any number of dimensions.

In NumPy, dimensions are called axes. Each dimension of an array has a length which is the total number of elements in that direction.

The size of an array is the total number of elements contained in an array in all the dimension. The size of NumPy arrays are fixed; once created it cannot be changed again.

Numpy arrays are great alternatives to Python Lists. Some of the key advantages of Numpy arrays are that they are fast, easy to work with, and give users the opportunity to perform calculations across entire arrays.

a) **Creating Numpy:** To create a NumPy use ndarray object by using the array() function.

Install Numpy by using a python library called pip by using the command :

pip install numpy .

To use NumPy you need to import Numpy:

import numpy as np # alias np

b) **Accessing and performing different array operations:**

Example 1: For NumPy with array object.

```
import numpy as np
a=np.array([1,2,3]) # one dimensional array
print(a)
```

Output:

```
[1 2 3]
```

Example 2:

```
arr=np.array([[1,2,3],[4,5,6]]) # two dimensional array
print(arr)
```

Output:

```
[[1 2 3]
```

```
[4 5 6]]
```

Example 3:

```
arr=np.array([[1,2,3],[4,5,6]])
a=arr.reshape(3,2)
print(a)
array([[1, 2],
```


[3, 4],
[5, 6]])

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write the output of the following program.

```
import numpy as np
arr = np.zeros((2, 3), dtype=float) # 2 rows and 3 columns
print (arr)
```

2. Write the output of the following program.

```
import numpy as np
arr = np.random.random((3, 3)) # 3 rows and 3 columns
print (arr)
```

3. Insert the correct method for creating a NumPy array.

```
arr = np.____([1, 2, 3, 4, 5])
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a Python NumPy program to create a 3x3 identity matrix.
2. Write a Python NumPy program to create a 4x4 matrix with values ranging from 2 to 10.

[Space for Answers]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References/Suggestions for further reading

1. <https://www.geeksforgeeks.org/python-numpy/>.
2. <https://www.scaler.com/topics/numpy/numpy-getting-started/>.
3. <https://www.javatpoint.com/numpy-array>.
4. <https://www.youtube.com/watch?v=QUT1VHiLmmI>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.16 Write program to demonstrate the use of user defined packages in Python.

I. Practical Significance

Packages make it significantly easier to reuse and maintain the code within a project and across different projects. Packages are used to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories: Built-in Packages (packages from the Java API) User-defined Packages (create your own packages). User-defined packages are the packages that are defined by the user.

II. Industry/Employer Expected Outcome

The aim of this course is to attain the following industry/employer expected outcome through various teaching learning experiences:
Develop programs using python to solve wide-reaching electronics engineering related problems.

III. Course Level Learning Outcome

CO5 - Create modules and packages for given purpose.

IV. Laboratory Learning Outcome

LLO 16.1 Implement user defined packages in python.

V. Relevant Affective Domain related outcome(s)

- a) Follow safety practices.
- b) Demonstrate working as a leader / a team member.
- c) Follow ethical practices.

VI. Relevant Theoretical Background

In Python, A user-defined package is a package that you create yourself, as opposed to one of the built-in packages that are included with Python by default. Creating a user-defined package is a way to organize your code and make it more modular and reusable.

User-defined packages are those packages that are designed or created by the developer to categorize classes and packages. It can be imported into other classes and used the same as we use built-in packages.

1. First, we create a directory and give it a package name, preferably related to its operation.

Directory name=Cars

2. Then we put the classes and the required functions in it.

Filename=Maruti.py

```
class Maruti:
```

```
def __init__(self):
```

```
    self.models=['800','Alto','WagonR']
```

```
def PModel(self):
```

```
    print("Models of Maruti")
```

```
    for model in self.models:
```

```
print('\t%s ' % model)

Filename=Mahindra.p
class Mahindra

def __init__(self):
self.models=['Scorpio','Bolero','Xylo']

def PModel(self):
print("Models of Mahendra")
for model in self.models:
print('\t%s ' % model)
```

3. Finally we create an `__init__.py` file inside the directory, to let Python know that the directory is a package.

```
Filename=__init__.py
from Maruti import Maruti
from Mahindra import Mahindra
```

4. To access package car, create `sample.py` file and access classes from directory car

```
Filename=sample.py
from Maruti import Maruti
from Mahindra import Mahindra
```

```
ModelMaruti=Maruti()
ModelMaruti.PModel()
```

```
ModelMahindra=Mahindra()
ModelMahindra.PModel()
```

Output:

```
Models of Maruti
800
Alto
WagonR
Models of Mahindra
Scorpio
Bolero
Xylo
```

VII. Resources Required

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System	Computer (i3-i5 preferable RAM>2GB)	As per Batch Size	For ALL Experiments
2.	Operating System	Windows/Linux		
3.	Development Software	Python IDE		

VIII. Resources used (Additional)

Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

IX. Practical related Questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Fill in the blanks:

Suppose you have a package named “**pkg**” with two modules **mod1** and **mod2**. mod1 has a function **func1()** and mod2 has a function **func2()**.

Complete the following Python code to import and use these functions:

```

_____
mod1.func1()
from pkg.mod2 _____
baz()
    
```

for first blank

- a) import pkg
- b) from pkg import mod1
- c) from pkg.mod1 import func1

for second blank

- a) import func2
- b) import func2 as fn
- c) import fn

2. Fill in the blanks:

Using the following package **pkg** following structure:

```
pkg/  
├── __init__.py  
├── sub_pkg1/  
│   ├── __init__.py  
│   ├── mod1.py  
│   └── mod2.py  
└── sub_pkg2/  
    ├── __init__.py  
    ├── mod3.py  
    └── mod4.py
```

import **mod1** from the subpackage **sub_pkg1** by using the following statement:

import _____

- a) mod1
- b) mod1 from pkg.sub_pkg1
- c) sub_pkg1.mod1 from pkg
- d) pkg. sub_pkg1.mod1

.....
.....
.....

X. Exercise

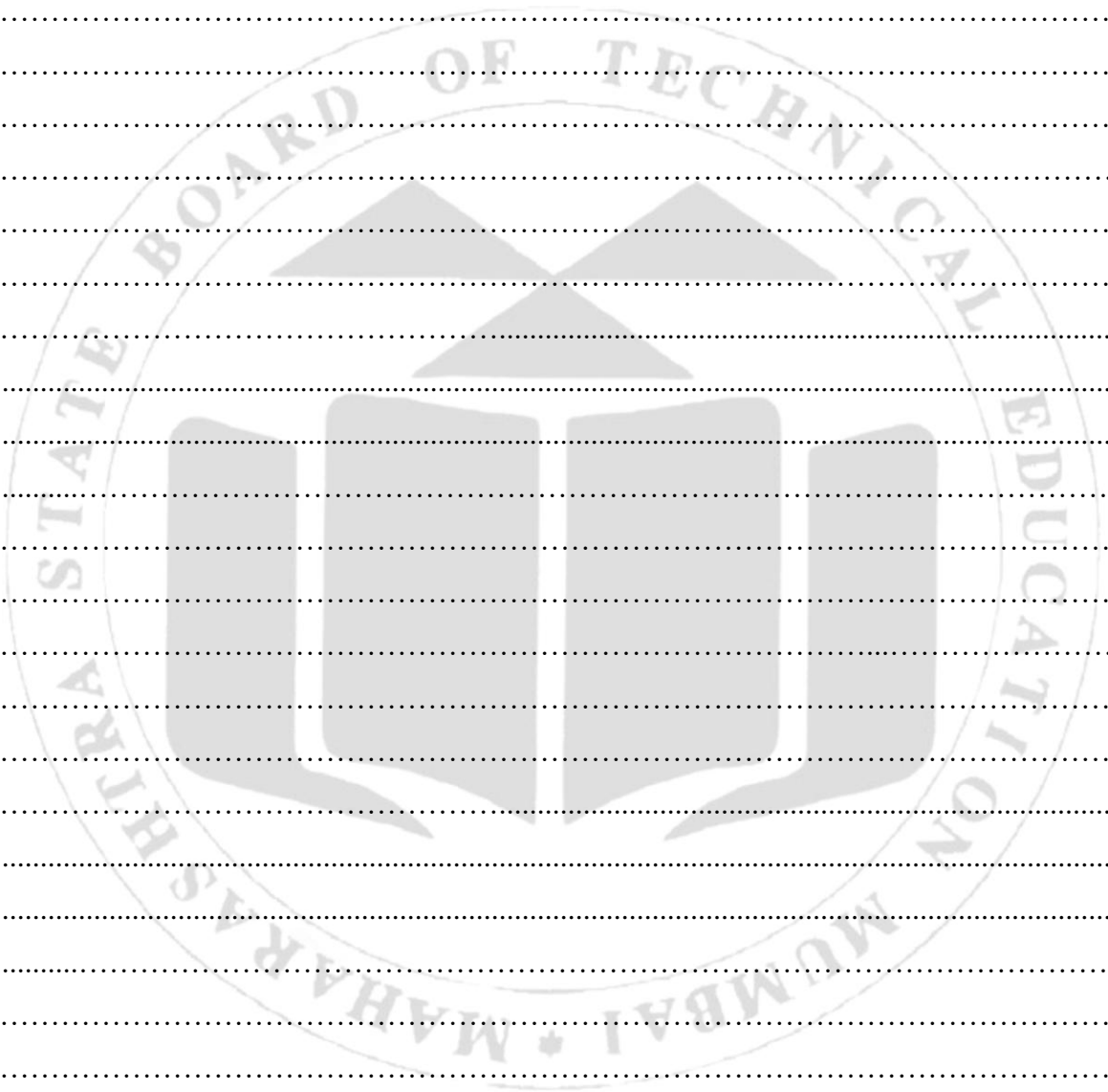
Note: Faculty must ensure that every group of students use different input value.

(Use blank space for answers or attach more pages if needed)

1. Write a python script to create a python package with the name “**calculator**” with modules addition, subtraction, multiplication, and division each module contain separate function for performing specific mathematical operations and import modules from the “**calculator**” packages and display the output.
2. Write a python script to create a python package with the name “**mypackage**” and two modules areafunctions.py and mathfunctions.py and display the output.

[Space for Answers]

.....
.....
.....
.....



XI. References/Suggestions for further reading

1. https://www.tutorialspoint.com/python/python_packages.htm
2. https://www.tutorialspoint.com/python/python_packages.htm
3. <https://www.prepbytes.com/blog/python/python-packages/>
4. <https://www.youtube.com/watch?v=g3t5XNVqDx0>

XII. Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		60%
1.	Debugging ability	20%
2.	Correctness of Program codes	30%
3.	Quality of output achieved (LLO mapped)	10%
Product Related: 10 Marks		40%
1.	Completion and submission of practical in time	20%
2.	Answer to exercise questions	20%
Total (25 Marks)		100 %

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	