SCHEME : K

Name : _____

Roll No. : _____ Year : 20___ 20___

Exam Seat No. : _____

LABORATORY MANUAL FOR

DATA STRUCTURE USING PYTHON

(313306)

COMPUTER 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.

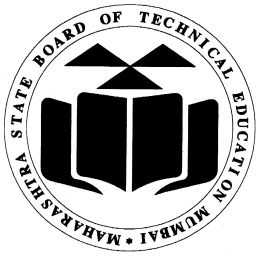
A Practical Manual
for

Data Structure using Python (313306)

Semester– (III)

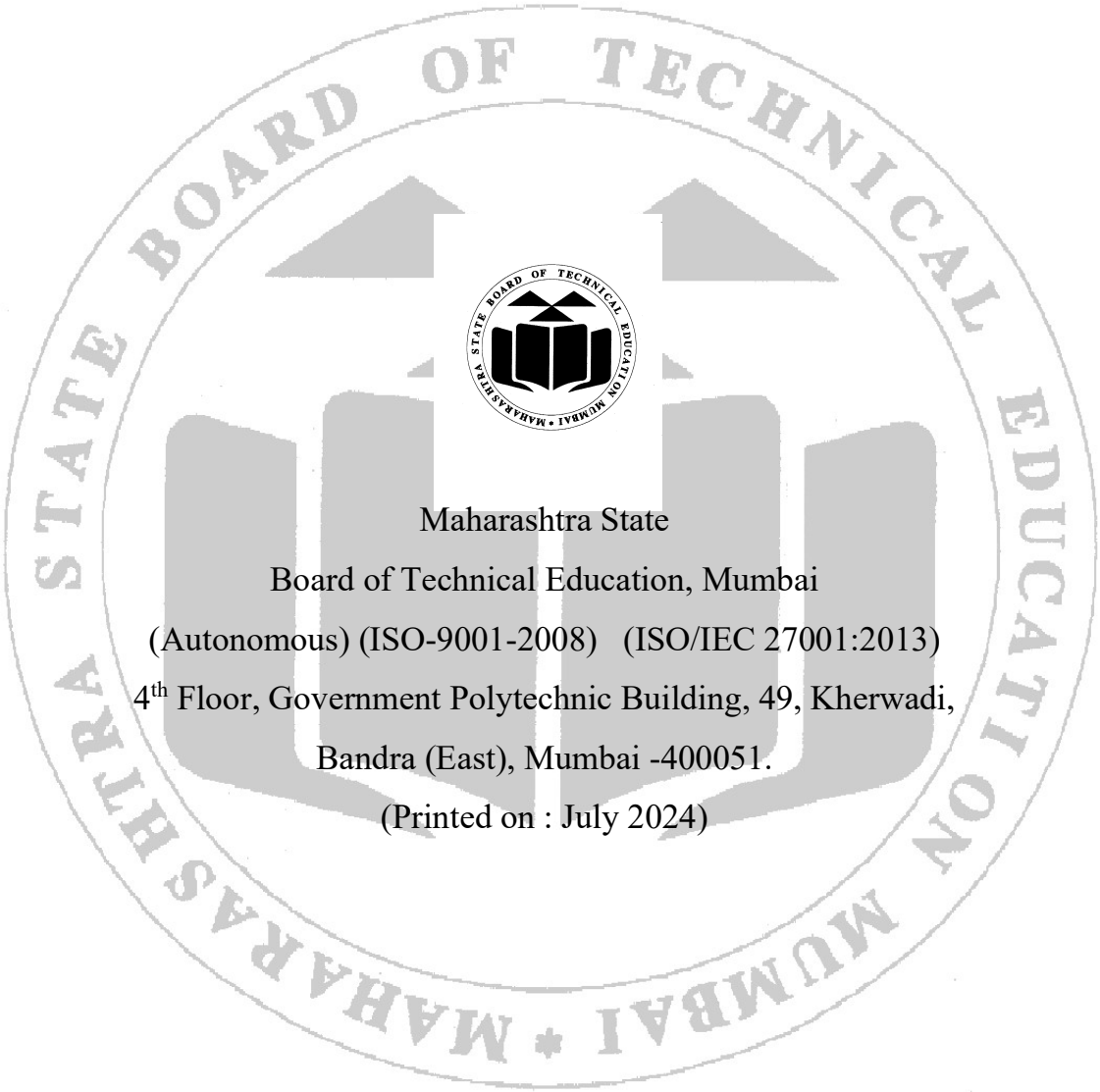
"K-SCHEME"

(AI/AN/DS)

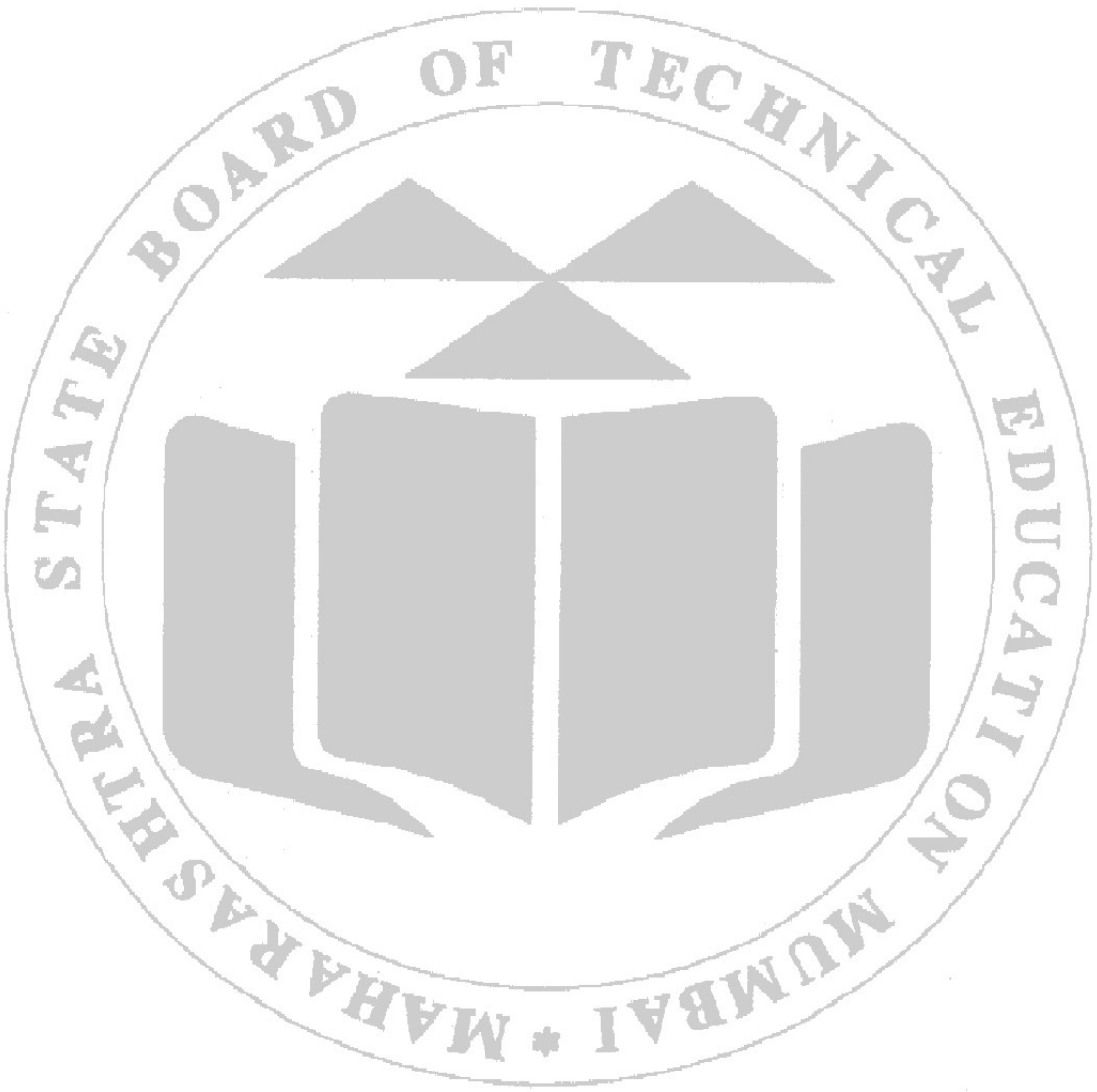


Maharashtra State
Board of Technical Education, Mumbai

(Autonomous) (ISO-9001-2008) (ISO/IEC 27001:2013)



Maharashtra State
Board of Technical Education, Mumbai
(Autonomous) (ISO-9001-2008) (ISO/IEC 27001:2013)
4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai -400051.
(Printed on : July 2024)


MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI

Maharashtra State
Board of Technical Education

Certificate

This is to certify that Mr. / Ms ………………………………………..…….

Roll No……………………….of Third Semester of Diploma in ……………………………..…..…..………..of the institute ……………………………………………………………..…………

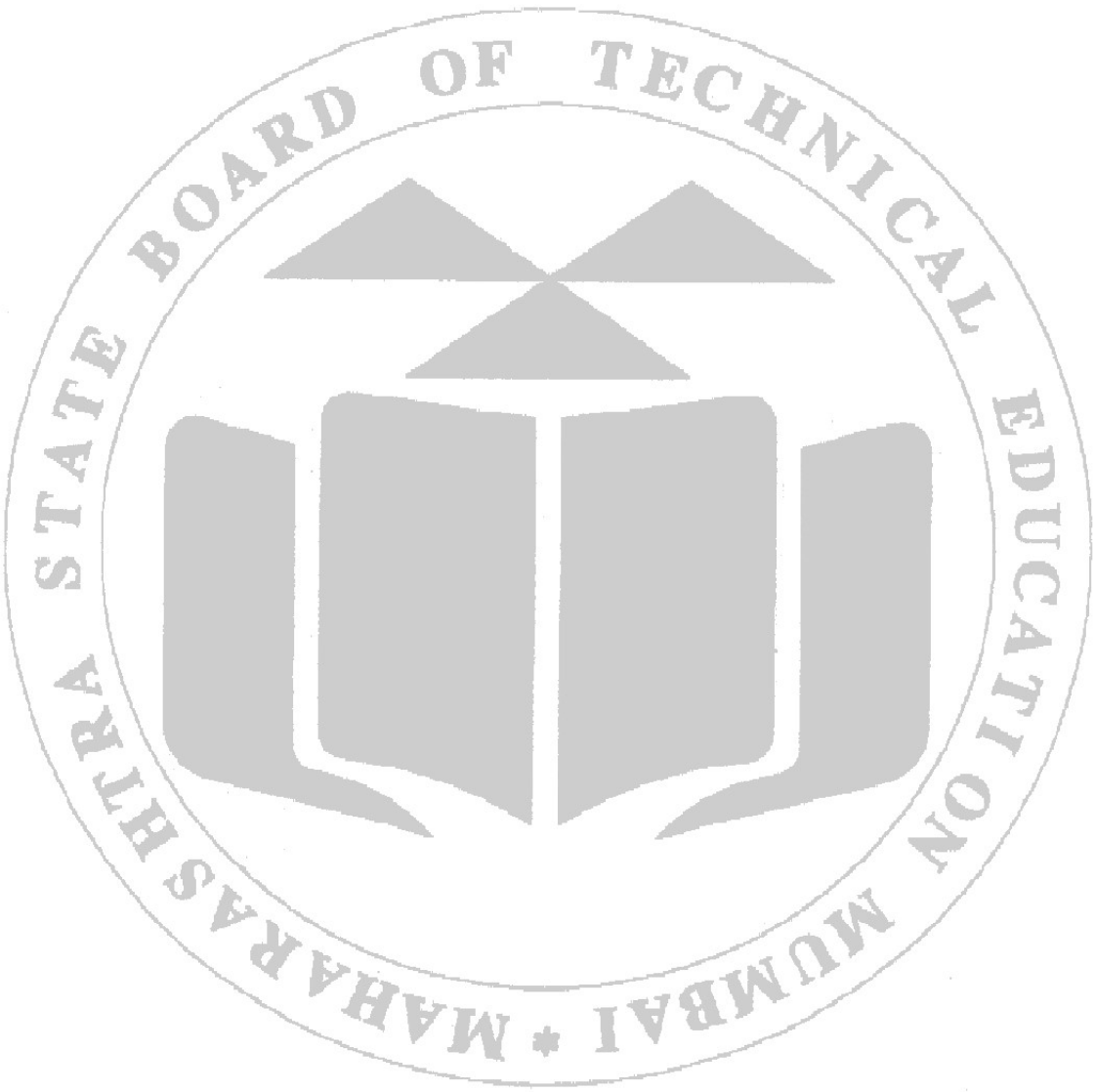
(Inst. Code………………..) has completed the term work satisfactorily in course Data Structure using Python **(313306)** for the academic year 20…….to 20…..... as prescribed in the curriculum.

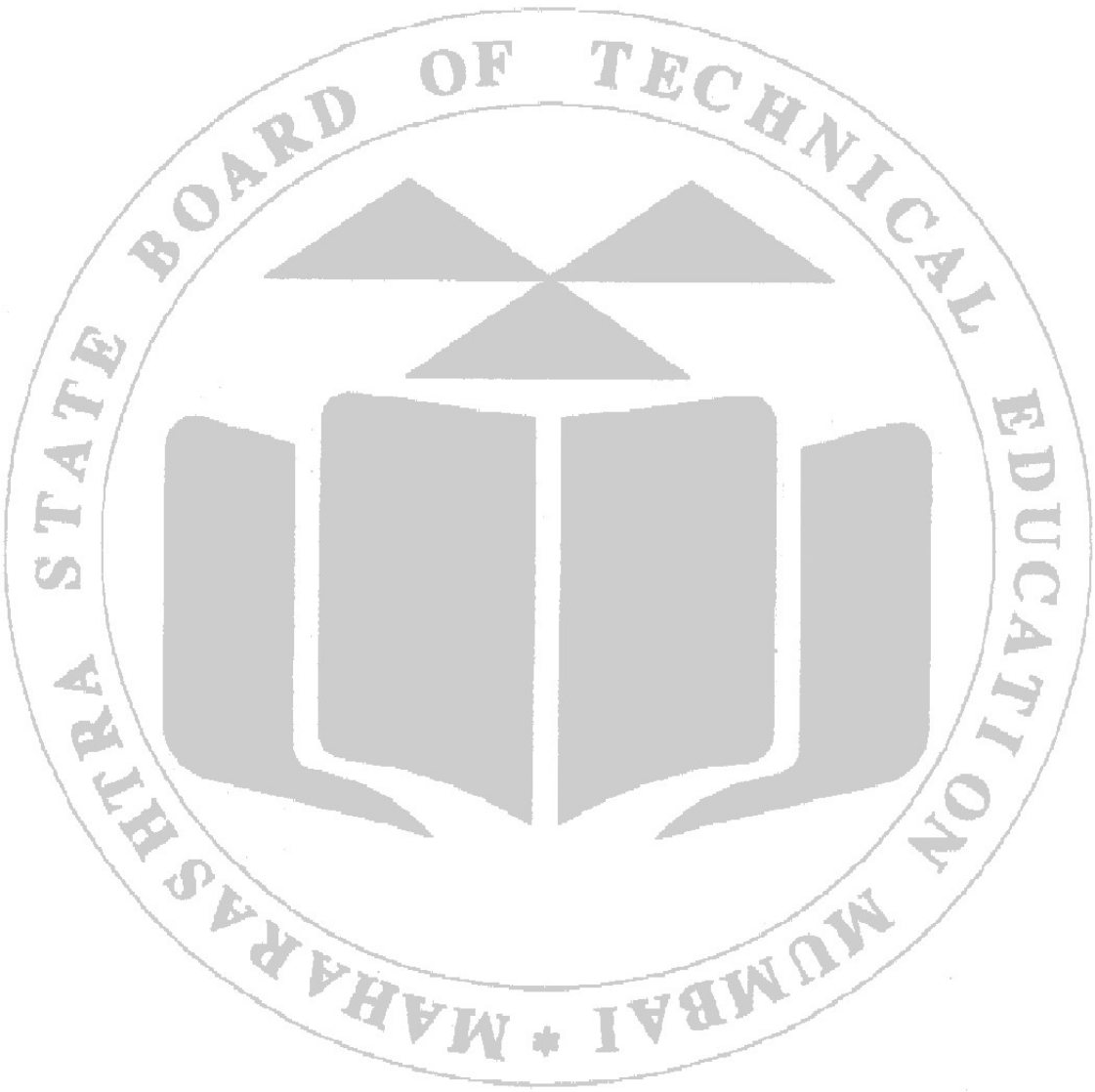
Place ………………. Enrollment No……………………

Date:…..................... Exam Seat No. …………………......

Course Teacher **Head of the Department** **Principal**

Seal of the Institute


MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI


MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION MUMBAI

Preface

The objective of all engineering laboratories or field experience in the technical education system is to help students acquire the critical competencies and skills that businesses demand. In light of this, MSBTE developed the cutting-edge “K” Scheme curricula for engineering diploma programs, emphasizing outcome-based learning and the National Education Policy 2020 (NEP2020). As a result, a sizable portion of the program is dedicated to practical work. This demonstrates how crucial laboratory work is in helping teachers, instructors, and students understand that every minute of lab time must be used efficiently to create these outcomes rather than being spent on pointless tasks. Consequently, each practical has been created to operate as a “vehicle” to advance this industry in order to ensure the successful implementation of this outcome-based curriculum. It is challenging to teach practical skills using only the “chalk and duster” activity. Because of this, the “K” scheme laboratory manual creation team focused on the outcomes when designing the practical rather than following the long-standing custom of doing the practical to “verify the theory” (which may turn out to be a by-product along the way).

This lab manual is intended to support all parties involved, particularly the students, instructors, and teachers, in helping the students achieve the pre-established objectives. It is required of every student to read through the relevant practical process in its entirety and comprehend the bare minimum of theoretical background related to the practical at least one day in advance of the practical. As a crucial starting point for carrying out the practical, each exercise in this manual starts with establishing the competency, industry-relevant skills, course outcomes, and practical outcomes. The skills, the students will acquire from the process outlined there, together with the necessary safety measures to be followed, will subsequently be made clear to them. These will enable them to apply the knowledge and abilities to solve real-world problems in their professional lives.

Data structures are an essential part of computer science and software development. They provide a way to organize and store data in a way that is efficient and easy to manipulate. Without data structures, it would be difficult to create efficient and effective software. One of the most important reasons for using data structures is to improve the performance of a program. In addition, data structures are also important in the field of artificial intelligence and machine learning. They are used to store and organize data in ways that make it easy for algorithms to process and learn from. After going through these learning experiences students will be able to identify the problem, analyze different algorithms and use them to solve the problem.

The team responsible for developing the Practical manual would like to express its gratitude to MSBTE for taking the lead in developing and implementing the curriculum. Additionally, the team recognizes the valuable contributions made by individual course experts who have been directly or indirectly involved in the development of the “K” scheme curriculum and the laboratory manual. It is impossible to claim perfection in this laboratory manual, even though every effort has been made to verify it for errors, especially because this is the first edition. Any such mistakes and recommendations for enhancements are quite appreciated and can be brought to our attention.

Lab Manual Development Team

Programme Outcomes (POs) to be achieved through Practical of this Course

Following POs are expected to be achieved through the practical's of the **Data Structure using Python** course.

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

PO 2. Problem analysis: Identify and analyze well-defined engineering problems using codified standard methods.

PO 3. 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.

PO 4. Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO 5. Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO 6. Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities in diverse and multidisciplinary fields.

PO 7. Life-long learning: Ability to analyze individual needs and engage in updating in the context of technological changes.

List of Industry Relevant Skills:

The following industry relevant skills of the competency 'Data Structure using Python 'are expected to be developed in you by undertaking the practical of this laboratory manual.

- Organize data in efficient manner.
- Use given data structure.
- Apply sorting techniques on databases.

Practical- Course Outcome (CO) matrix

CO1 - Develop Python program using basic syntactical constructs.
CO2 - Perform operations on sequence structures in Python.
CO3 - Implement Modules, Packages in Python for given problem.
CO4 - Design classes for given problem.
CO5 - Implement Linear Data Structure in Python.
CO6 - Develop Python program to implement tree data structure.

Sr. No.	Title of the Practical	CO 1.	CO 2.	CO 3.	CO 4.	CO 5.	CO 6.
1	*a) Install and configure Python IDE. b) Write Python program to display message on screen	√	-	-	-	-	-
2	*Write simple Python program using operators: a) Arithmetic Operators b) Logical Operators c) Relational Operators d) Conditional Operators e) Bitwise Operators f) Ternary Operators	√	-	-	-	-	-
3	*Write simple Python program to demonstrate use of conditional statements: a) if b) if…else c) Nested 'if' d) Switch case	√	-	-	-	-	-
4	Develop a simple Python program to demonstrate use of control loop: while	√	-	-	-	-	-
5	* 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 Lists: a) Create b) Access c) Update d) Delete elements in list	-	√	-	-	-	-
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	Apply math built-in function in Python.	-	√	-	-	-	-
11	*Develop a user defined Python function for given problem: Write a function with minimum 2 arguments.	-	√	-	-	-	-
12	Create a program to demonstrate use of built-in module (e.g. numeric, mathematical functional and programming module) in Python.	-	-	√	-	-	-
13	*Write a program to create a user- defined module (e.g. Building calculator) in Python.	-	-	√	-	-	-
14	*Develop a Python program to demonstrate use of NumPy packages for creating, accessing and performing different array operations.	-	-	-	√	-	-
15	Write a program to create user defined packages in Python.	-	-	-	√	-	-
16	Write a program in Python to demonstrate following operations: a) Method overloading b) Method overriding	-	-	-	√	-	-
17	*Develop program in Python to demonstrate following operations: a) Single Inheritance b) Multilevel Inheritance c) Multiple inheritance d) Hybrid Inheritance e) Hierarchical Inheritance	-	-	-	√	-	-
18	*Write a program in Python for handling array to demonstrate following operations: a) Array declaration b) Insertion c) Deletion d) Append e) Index f) Reverse	-	-	-	-	√	-

19	*Write a program in Python for linked list to demonstrate following operations a) Insertion b) Deletion c) Updating d) Merging to list	-	-	-	-	√	-
20	*Write a program in Python to demonstrate queues data structure operations. a) Enqueue b) Dequeue c) Display	-	-	-	-	√	-
21	*Use the searching techniques in data structures: a) Linear Search b) Binary Search	-	-	-	-	√	-
22	*Write a Python program to implement following sorting techniques: a) Bubble Sort b) Insertion Sort	-	-	-	-	√	-
23	* Write a program in Python to evaluate an expression.	-	-	-	-	√	-
24	Write a program in Python to Create binary tree from the given list using Binary tree module in python.	-	-	-	-	-	√

Guidelines to Teachers

1. There will be two sheets of blank pages after every practical for the student to report other matters (if any), which is not mentioned in the printed practical.
2. For difficult practical if required, teacher could provide the demonstration of the practical emphasizing of the skills which the student should achieve.
3. Teachers should give opportunity to students for hands-on after the demonstration.
4. Assess the skill achievement of the students and COs of each unit.
5. One or two questions ought to be added in each practical for different batches. For this teacher can maintain various practical related question banks for each course.
6. For effective implementation and attainment of practical outcomes, teacher ought to ensure that in the beginning itself of each practical, students must read through the complete write-up of that practical sheet.
7. During practical, ensure that each student gets chance and takes active part in taking observations/ readings and performing practical.
8. Teacher ought to assess the performance of students continuously according to the MSBTE guidelines.

Instructions for Students

1. For incidental writing on the day of each practical session every student should maintain a ***dated log book*** for the whole semester, apart from this laboratory manual which s/he has to ***submit for assessment to the teacher*** in the next practical session.
2. For effective implementation and attainment of practical outcomes, in the beginning itself of each practical, students need to read through the complete write-up including the practical related questions and assessment scheme of that practical sheet.
3. Student ought to refer the reference book, lab manuals etc. Student should not hesitate to ask any difficulties they face during the conduct of practical's.

Content Page

List of Practical and Progressive Assessment Sheet

S. No	Laboratory Practical Titles	Page No.	Date of performance	Date of submission	FA PR marks (25)	Dated sign. of teacher	Remarks (if any)
	in Python: a) Create b) Access c) Update d) Delete e) Looping through Dictionary						
10, 11	Apply math built-in function in Python. *Develop a user defined Python function for given problem: Write a function with minimum 2 arguments.	66					
12, 13	Create a program to demonstrate use of built-in module (e.g. numeric, mathematical functional and programming module) in Python. *Write a program to create a user-defined module (e.g. Building calculator) in Python.	74					
14, 15	*Develop a Python program to demonstrate use of NumPy packages for creating, accessing and performing different array operations. Write a program to create user defined packages in Python.	80					
16	Write a program in Python to demonstrate following operations: a) Method overloading b) Method overriding	86					
17	*Develop program in Python to demonstrate following operations: a) Single Inheritance b) Multilevel Inheritance c) Multiple inheritance d) Hybrid Inheritance e) Hierarchical Inheritance	92					
18	*Write a program in Python for handling array to demonstrate following operations: a) Array declaration b) Insertion c) Deletion d) Append e) Index f) Reverse	98					
19	*Write a program in Python for linked list to demonstrate following operations a) Insertion b) Deletion c) Updating d) Merging to list	104					

S. No	Laboratory Practical Titles	Page No.	Date of performance	Date of submission	FA PR marks (25)	Dated sign. of teacher	Remarks (if any)
20	*Write a program in Python to demonstrate queues data structure operations. a) Enqueue b) Dequeue c) Display	109					
21	*Use the searching techniques in data structures: a) Linear Search b) Binary Search	114					
22	*Write a Python program to implement following sorting techniques: a) Bubble Sort b) Insertion Sort	119					
23	*Write a program in Python to evaluate an expression.	124					
24	Write a program in Python to Create binary tree from the given list using Binary tree module in python.	128					
	Total						

Note: To be transferred to Proforma of CIAAN-2023.

Note : Out of above suggestive LLOs -

- '*' Marked Practical's (LLOs) Are mandatory.
- Minimum 80% of above list of lab experiment are to be performed.
- Judicial mix of LLOs are to be performed to achieve desired outcomes.

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.

Python is a high-level language which is trending in recent past. To make it more user-friendly developers of Python made sure that it can have two modes Viz. Interactive mode and Script Mode. The Script mode is also known as normal mode and uses scripted and finished .py files which are run on interpreter whereas interactive mode supports command line shells. Students will able to understand how to run programs using Interactive and Script mode.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern mechanical engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Selecting and installing Python in Windows using appropriate installer.
- Setting up Python environment for execution of Python programs
- Write a simple Python Program using Interactive mode.
- Write a simple Python Program using Script mode.

IV. Relevant Course Outcome (s)

CO1- Develop python program using basic syntactical constructs.

V. Practical Outcomes (PrOs)

- Setup a Python programming development environment
- Write simple Python program to display message on screen

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Python was created by Guidovan 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 Flying Circus' and not after Python –the snake.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$Python
Python2.4.3(#1,Nov112010,13:34:43)
[GCC4.1.220080704(RedHat 4.1.2-48)]onlinux2
Type "help","copyright","credits" or "license" for more information.
```

Type the following text at the Python prompt and press the Enter –

```
>>>print"Hello,Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");** however, in Python version 2.4.3, this produces the following result:

```
Hello,Python!
>>>
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file:

```
print"Hello,Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows:

```
$Pythontest.py
```

This produces the following result:

```
Hello,Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file:

```
#!/usr/bin/Python
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows:

```
$chmod+xtest.py        #This is to make file executable
$./test.py
```

This produces the following result –

```
Hello, Python!
```

Installing Python in Windows:

- Open any internet browser. Type http:///www.Python.org/downloads/ in address bar and Enter.
- Home page of Python will have displayed as shown in Fig.1.1

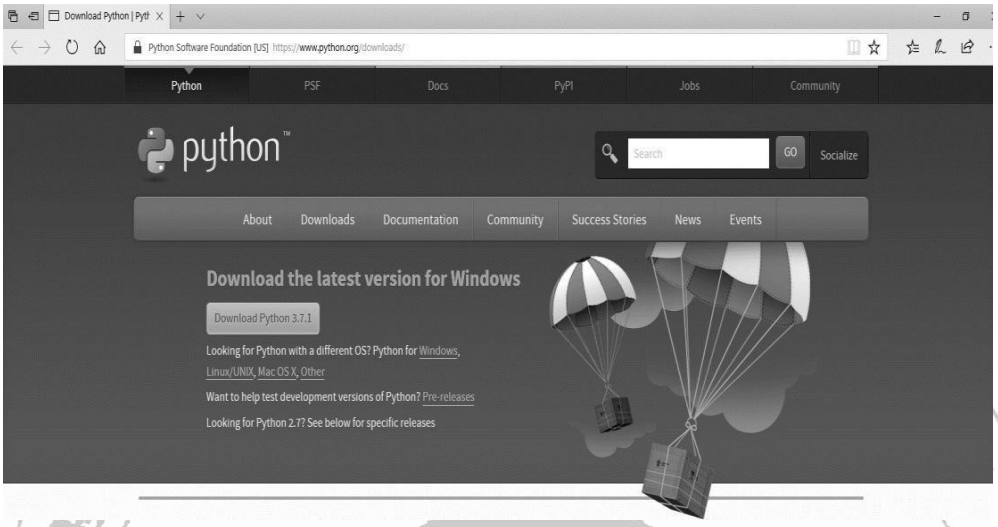


Fig.1.1: Home Page

- Click on download the latest version for windows, which shows latest version as shown in Fig.1.2

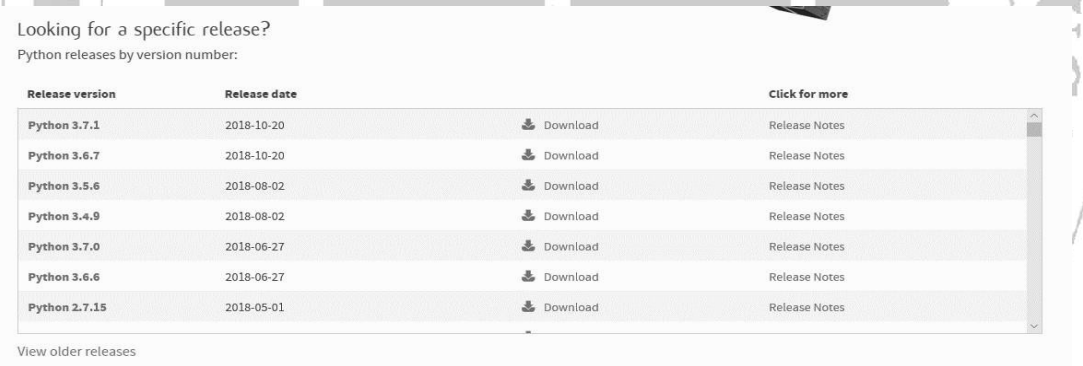


Fig.1.2: Python release versions

- Open the Python3.7.1 version packs and double click on it to start installation and installation windows will be open as shown in Fig.1.3.

Fig.1.3: Installation Type

- Click on next install now for installation and then Setup progress windows will be opened as shown in Fig.1.4.

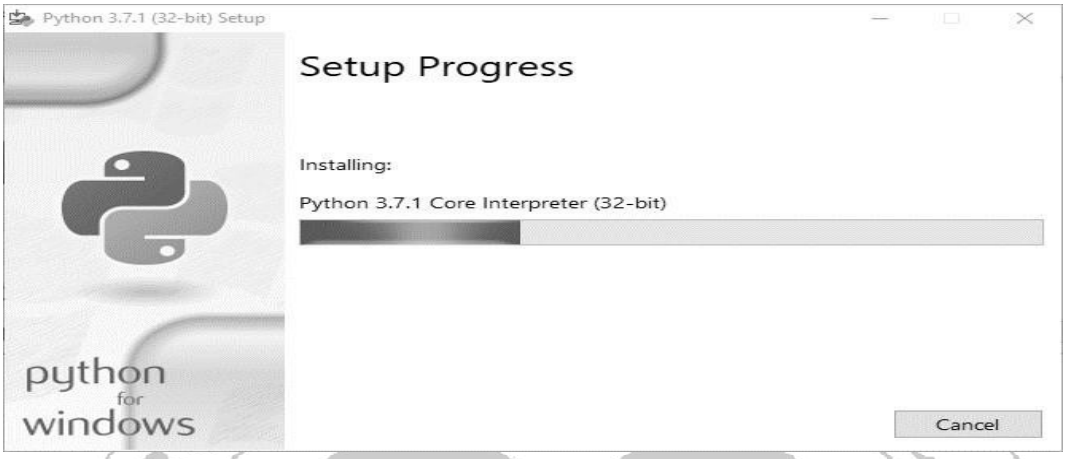


Fig.1.4: Setup Progress

- After complete the installation, click on close button in the windows as shown in Fig.1.5.

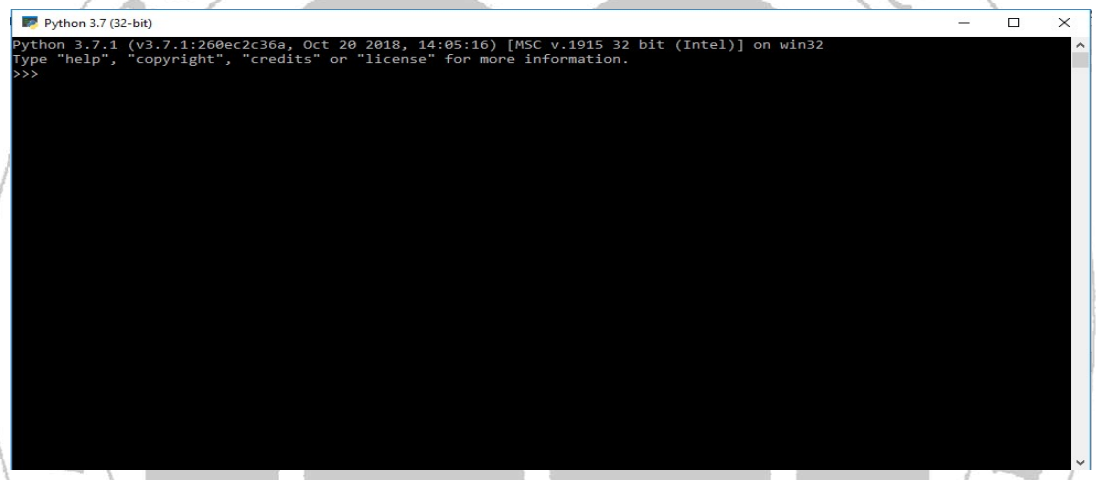
Fig.1.5: Setup Completion

Starting Python in different modes:

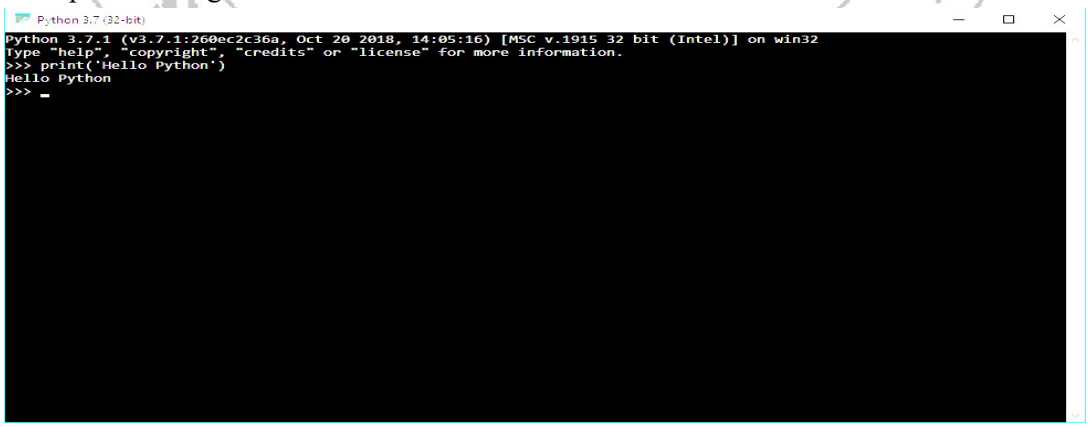
1) Starting Python (Command Line)

- Press start button

- Click on all programs and then click on Python3.7 (32bit). 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.

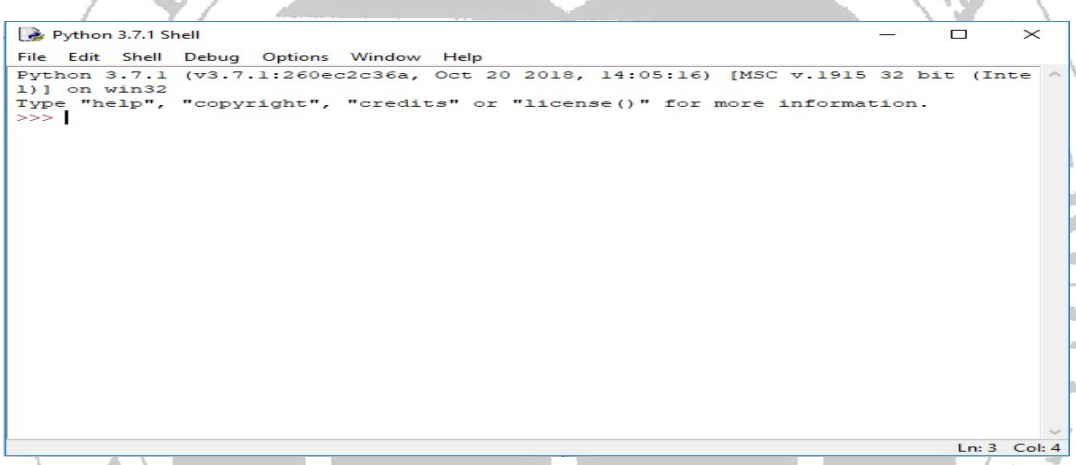


To exit from the command line of Python, press Ctrl+z followed by Enter or Enter exit() or quit() and Enter.

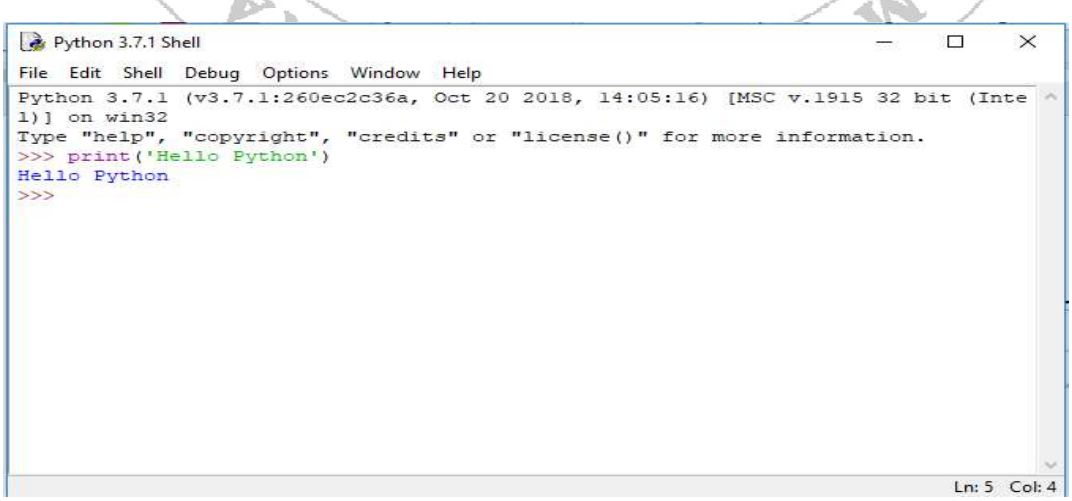
2) Starting Python IDLE

- Press start button and click on IDLE (Python3.732 bit) options.

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



- 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.



VIII. 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		

IX. Resources used (Additional)

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

X. 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. Explain Python Path
5. State use of pip
6. List different modes of Programming in Python
7. Describe procedure to execute program using Interactive Mode
8. State the steps involved in executing the program using Script Mode

(Space for answers)

.....

.....

.....

.....

.....

.....

XI. Exercise

(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)

XII. References/Suggestions for further Reading

1. https://www.howtogeek.com/197947/how-to-install-Python-on-windows/
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
5. https://www.tutorialspoint.com/Python/Python_basic_syntax.htm
6. https://www.youtube.com/watch?v=skIy7ZhT_tk
7. https://www.youtube.com/watch?v=3xp-ixFbDuE

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No.2 Write simple Python program using operators: a) Arithmetic Operators, b) Logical Operators, c) Relational Operators, d) Conditional Operators, e) Bitwise Operators, f)Ternary Operators.

I. Practical Significance

Operators are used to perform operations on values and variables. Operators can manipulate individual items and returns a result. The data items are referred as operands or arguments. Operators are either represented by key words or special characters. Here are the types of operators supported by Python:

- Arithmetic Operators
- Logical Operators
- Relational or Comparison Operators
- Conditional Operators
- Bitwise Operators
- Ternary Operators

Students will be able to use various operators to check the condition and get appropriate result by performing different operation with the help of supported operators.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
- **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 in diverse and multidisciplinary fields.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Write a program to use all types of Operators

IV. Relevant Course Outcome(s)

CO1- Develop python program using basic syntactical constructs.

V. Practical Outcomes (PrOs)

Develop Python program using operators: Arithmetic Operators, Logical Operators, and Bitwise Operators

VI. Relevant Affective Domain related Outcome(s)

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

3. Follow ethical practices

VII. Minimum Theoretical Background

Arithmetic Operators: Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, division, % modulus, exponent etc.

Operator	Meaning	Description	Example	Output
+	Addition	Adds the value of the left and right operands	>>>1+4	5
-	Subtraction	Subtracts the value of the right operand from the value of the left operand	>>>10-5	5
*	Multiplication	Multiplies the value of the left and right operand	>>>5*2	10
/	Division	Divides the value of the left operand by the right operand	>>>10/2	5
**	Exponent	Performs exponential calculation	>>>2**3	8
%	Modulus	Returns the remainder after dividing the left operand with the right operand	>>>15%4	3
//	Floor Division	Division of operands where the Solution is a quotient left after removing decimal numbers	>>>17//5	3

Logical Operators: Logical operators perform Logical AND, Logical OR and Logical NOT operations. For logical operators following condition are applied.

- For AND operator – It returns TRUE if both the operands (right side and left side) are true
- For OR operator- It returns TRUE if either of the operand (right side or left side) is true
- For NOT operator- returns TRUE if operand is false

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

Relational Operators (Comparison Operators): Relational Operators are used to compare the values.

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(a==b) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(a!=b) is true.
<>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(a<>b) is true. This is similar to != operator.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a>b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a<b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a>=b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a<=b) is true.

Conditional Operators: The Python conditional operator, also known as the ternary operator, provides a shorthand way of writing an if-else statement. It is a way to evaluate a condition and return one of two values based on the result of that condition.

Syntax:
value_if _true if condition else value_if_false
Example:

```
num = 5
result = "Positive" if num> 0 else "Negative"
print(result)
# Output: Positive
```

Bitwise Operators: Bitwise operators act on bits and performs bit by bit operation. If a=10(1010) and b=4(0100)

Operator	Meaning	Description	Example	Output
&	Bitwise AND	Operator copies a bit, to the result, if it exists in both operands	>>>(a & b)	0
|	Bitwise OR	It copies a bit, if it exists in either operand.	>>>(a | b)	14
~	Bitwise NOT	It is unary and has the effect of 'flipping' bits.	>>>(~a)	-11

^	Bitwise XOR	It copies the bit, if it is set in one operand but not both.	>>>(a ^ b)	14
>>	Bitwise right shift	The left operand's value is moved right by the number of bits specified by the right operand.	>>>(a>>2)	2
<<	Bitwise left shift	The left operand's value is moved Left by the number of bits specified by the right operand.	>>>(a<<2)	40

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Mention the use of //, ** ,% operator in Python
2. Describe ternary operator in Python
3. Describe about different Logical operators in Python with appropriate examples.
4. Describe about different Arithmetic operators in Python with appropriate examples.
5. Describe about different Bitwise operators in Python with appropriate examples.

(Space for answers)

.....

.....

.....

XI. 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 program to convert U. S. dollars to Indian rupees.
2. Write a program to convert bits to Megabytes, Gigabytes and Terabytes
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 answers)

.....

.....

.....

.....

.....

.....

XII. 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

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 3 Write simple Python program to demonstrate use of conditional statements: a) if, b) if…else, c) Nested if, d) Switch case

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 otherwise.

While developing a computer application one need to identify all possible situation. One situation is to repeat execution of certain code block/ line of code. Such situation can be taken care by looping system. Like all other high level programming language, Python also supports all loop structure. To keep a computer doing useful work we need repetition, **looping** back over the same block of code again and again. This practical will describe the different kinds of **loops in Python**.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem

The practical is expected to develop the following skills:

- Write a program to implement control flow operations.

IV. Relevant Course Outcome(s)

CO1- Develop python program using basic syntactical constructs

V. Practical Outcomes (PrOs)

- Develop Python program using Conditional operators: if, if-else and Nested if statement.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

a) if Statement: if statement is the simplest decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if condition:
    #Statement(s)
```

Example:

```
i=10
if(i > 20):
        print ("10 is less than 20")
 print ("We are Not in if ")
```

Output: We are Not in if

b) if-else Statement: The if statement alone tells us that if a condition is true, it will execute a block of statements and if the condition is false, it won't. But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with if statement to execute a block of code when the condition is false.

Syntax:

```
if (condition):
   # if Condition is true, Executes this block
else:
   # if condition is false, Executes this block
```

Example:

```
i=10;
if(i<20):
   print ("i is smaller than 20")
else:
   print ("i is greater than 25")
```

Output:

i is smaller than 20

c) Nested-if Statement:

A nested if is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement. Yes, Python allows us to nest if statements within if statements. i.e., we can place an if statement inside another if statement.

Syntax:

```
if (condition1):
   # Executes when condition1 is true
   if (condition2):
        # Executes when condition2 is true
        # if Block is end here
# if Block is end here
```

Example:

```
i =10
```

```
if(i ==10):
  # First if statement
  if(i < 20):
    print ("i is smaller than 20")
  # Nested - if statement will only be executed if statement above is trueif
  (i < 15):
    print ("i is smaller than 15 too")
  else:
    print ("i is greater than 15")
```

Output:

i is smaller than 20
i is smaller than 15 too

d) Switch case statement:

There are no built-in switch case statements in Python. However, Python offers multiple methods to replace the switch case functionality to harness all the benefits of switch.

Methods to Implement Switch Case in Python:

- Using Dictionary
- Using Classes
- Using Cif-elif Statements

VIII. 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		

IX. Resources used (Additional)

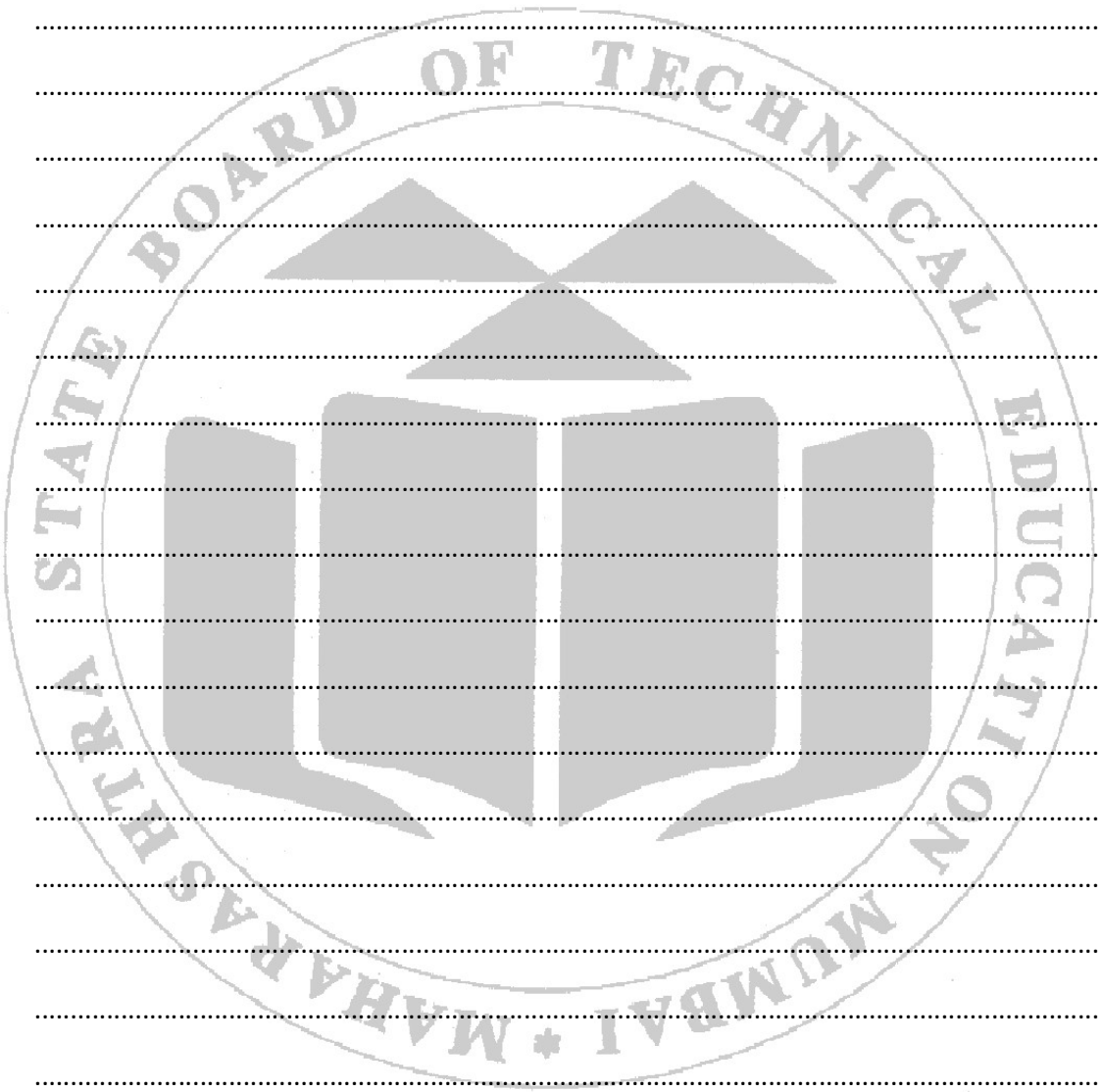
Sr. No.	Name of Resource	Specification	Quantity	Remarks (If any)
1.	Computer System			
2.	Operating System			
3.	Development Software			

X. Practical related Questions

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

1. List operators used in if conditional statement
2. Differentiate between if-else and nested-if statement

(Space for answers)



.....

XI. 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 program to check whether a number is even or odd
2. Write a program to check the largest number among the three numbers
3. Write a program to check if the input year is a leap year of not
4. Write a program to check if a Number is Positive, Negative or Zero
5. Write a program that takes the marks of 5 subjects and displays the grade.
6. Write a program to implement switch case using else if ladder

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XII. 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
7. https://www.tutorialspoint.com/Python/Python_loops.htm
8. https://www.youtube.com/watch?v=zFvoXxeoosI
9. https://www.youtube.com/watch?v=IS7AAVl77kA

XIII. Assessment Scheme

	Performance indicators	Weightage
	Process related (15 Marks)	**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
	Product related (10 Marks)	**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 4 Develop a simple Python program to demonstrate use of control loop: while

I. Practical Significance

While developing a computer application one need to identify all possible situation. One situation is to repeat execution of certain code block/ line of code. Such situation can be taken care by looping system. Like all other high level programming language, Python also supports all loop structure. To keep a computer doing useful work we need repetition, **looping** back over the same block of code again and again. This practical will describe the different kinds of **loops in Python**.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem

The practical is expected to develop the following skills:

a. Write a Python Program using 'while' loop

b. Write a Python Program using 'do… while' loop

IV. Relevant Course Outcome(s)

CO1- Develop python program using basic syntactical constructs.

V. Practical Outcomes (PrOs)

Write Python program to demonstrate use of looping statements:

a. 'while' loop

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

A loop statement allows us to execute a statement or group of statements multiple times. Python programming language provides following types of loops to handle looping requirements.

a) while loop: A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

```
while expression:
  #statement(s)
```

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

Example

```
#!/usr/bin/Python
count = 0
while (count < 5):
   print 'The count is ', count
   count = count + 1
print "Good bye!"
```

Output:

The count is 0

The count is 1

The count is 2

The count is 3

The count is 4

Good bye!

VIII. 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		

IX. Resources used (Additional)

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

X. 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. What would be the output from the following Python code segment?

```
x=10
while x>5:
    print (x)
  x - = 1
```

2. Write a program to print first 10 integers and their squares using while loop.

(Space for Answer)

XI. 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 Reverse a Given Number
3. Write a Python program takes in a number and finds the sum of digits in a number.

(Space for answers)

XII. 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=IS7AAVl77kAS

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 5: 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

While developing a computer application one need to identify all possible situation. One situation is to repeat execution of certain code block/ line of code. Such situation can be taken care by looping system. Like all other high level programming language, Python also supports all loop structure. To keep a computer doing useful work we need repetition, **looping** back over the same block of code again and again. This practical will describe the different kinds of **loops in Python**.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem

The practical is expected to develop the following skills:

a. Write a program using conditional statements
b. Write a Program using control loops

IV. Relevant Course Outcome(s)

CO1- Develop python program using basic syntactical constructs

V. Practical Outcomes (PrOs)

- Develop Python program using Looping statements: for loop

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

A loop statement allows us to execute a statement or group of statements multiple times. Python programming language provides following types of loops to handle looping requirements.

a) for loop: It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for iterating_var in sequence:
  # statements(s)
```

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable *iterating_var*. Next, the statements block is executed. Each item in the list is assigned to *iterating_var*, and the statement(s) blockis executed until the entire sequence is exhausted.

Example

```
#!/usr/bin/Python
for letter in 'Python':                    #   First   Example
  print 'Current Letter :', letter
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:                       # Second Example
  print 'Current fruit :', fruit
print "Good bye!"
```

Output:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango Good bye!
```

b) nested loops: Python programming language allows to use one loop inside anotherloop. Following section shows few examples to illustrate the concept.

Syntax

```
for iterating_var in sequence:
  for iterating_var in sequence:
    #statements(s)
  #statements(s)
```

Example

```
x = [1, 2]
y = [4, 5]
 for i in x:
  for j in y:
   print(i, j)
```

Output:

```
1 4
1 5
2 4
2 5
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. What would be the output from the following Python code segment?

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
  for x in adj:
    for y in fruits:
      print(x, y)
```

2. Change the following Python code from using a while loop to for loop:

```
x=1
while x<10:
      print x,
      x+=1
```

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. 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 following patterns using for loop:

```
    *
  *   *
*   *   *
  *   *   *
```

2. Write a Python program to find the sum of first 10 natural numbers using for loop.
3. Write a Python program to print Fibonacci series.
4. Write a Python program to print multiplication table of a given number.
5. Write a Python program to calculate factorial of a number.

(Space for answers)

.....

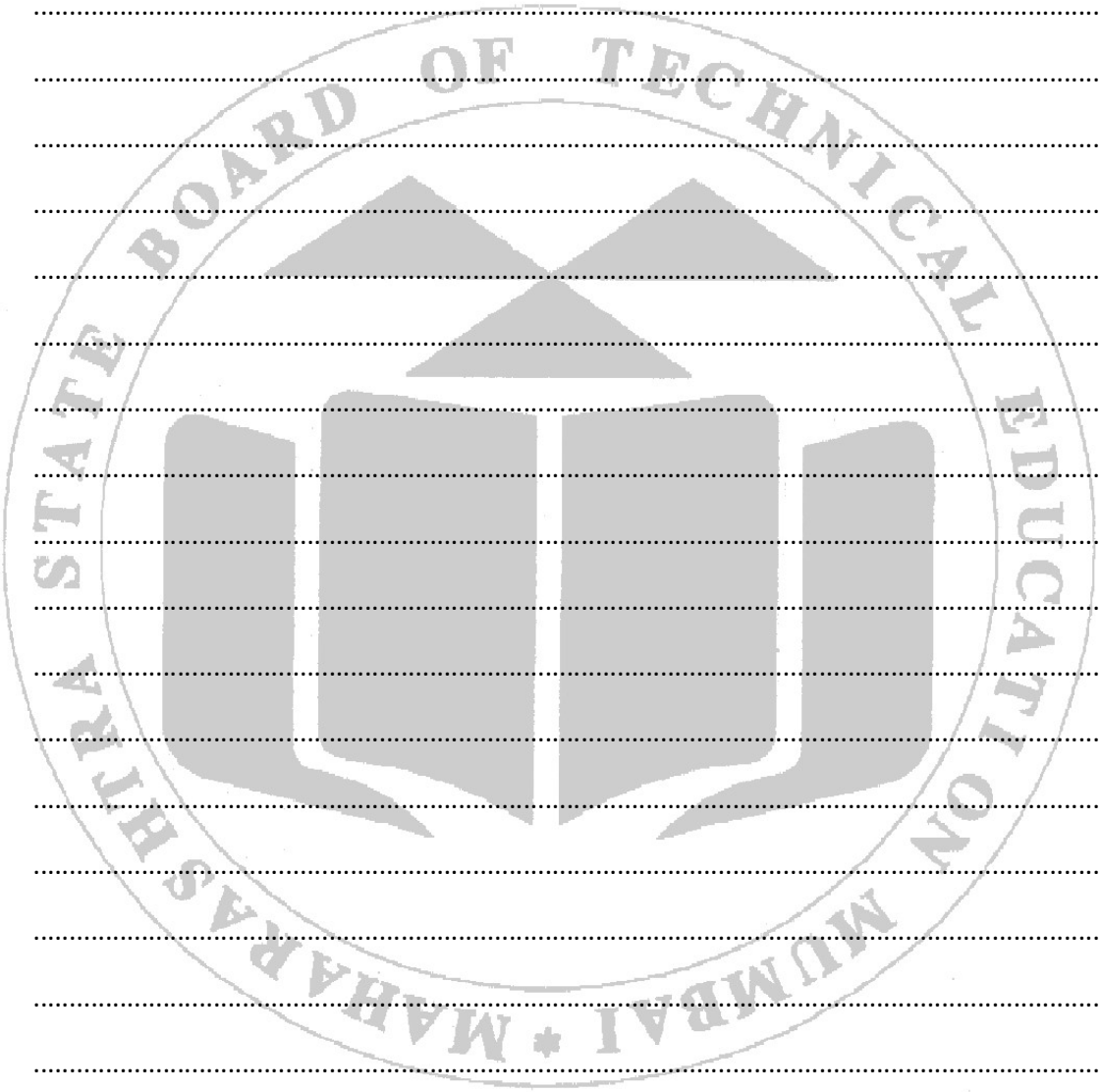
.....

.....

.....

.....

.....



XII. 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
7. https://www.tutorialspoint.com/Python/Python_loops.htm
8. https://www.youtube.com/watch?v=zFvoXxeoosI
9. https://www.youtube.com/watch?v=IS7AAVl77kA

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	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 lists: a) Create b) Access c) Update d) Delete elements in list

I. Practical Significance

A list can be defined as a collection of values or items of different types. The items inthe list are separated with the comma (,) and enclosed with the square brackets []. The elements or items in a list can be accessed by their positions i.e. indices. This practical will make student acquainted with use of list and operations on list in Python.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem The practical is expected to develop the following skills:

a. Write a program using List and performs operations like create, access, delete. statements

IV. Relevant Course Outcome(s)

CO 2- Perform operations on sequence structures in python.

V. Practical Outcome(PrOs)

Develop Python program to perform following operations on Lists: Create list, Access list Update list (Add item, Remove item) and Delete list

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

a. A list is a collection which is ordered and changeable.
b. A list is a collection of items or elements; the sequence of data in a list is ordered.
c. The elements or items in a list can be accessed by their positions i.e. indices
d. In Python lists are written with square brackets. A list is created by placing all the items (elements) inside a square brackets [ ], separated by commas.
e. Lists are mutable. The value of any element inside the list can be changed at any point of time.
f. The index always starts with 0 and ends with n-1, if the list contains n elements.

a) Creating List: Creating a list is as simple as putting different comma-separated values between square brackets.

Example:

```
>>>List1=['Java','Python','Perl']
>>>List2=[10,20,30,40,50]
```

b) Accessing List: To access values in lists, use the square brackets for slicing alongwith the index or indices to obtain value available at that index.

Example:

```
>>>List2
[10,20,30,40,50]
>>>List2[1]
20
>>>List2[1:3]
[20,30]
>>>List2[5]
Traceback (most recent call last):
File "<pyshell#71>", line 1, in <module>
List2[5]
IndexError: list index out of range
```

c) Updating List: You can update single elements of lists by using the assignment (=) operator with the specified index or update multiple elements in the list using the slicing and assignment operator, and you can add to elements in a list with the append() method.

```
>>>List2
[10,20,30,40,50]
>>>List2[0]=60          #Updating first item
[60,20,30,40,50]
>>>List2[3:4]=70,80
>>>[60,20,30,70,80,50]
```

i) We can add one item to a list using append() method or add several items using extend() method.

```
>>> list1=[10,20,30]
>>> list1
[10, 20, 30]
>>> list1.append(40)
>>> list1
[10, 20, 30, 40]
>>> list1.extend([60,70])
>>> list1
[10, 20, 30, 40, 60, 70]
```

ii) We can also use + operator to combine two lists. This is also called concatenation

```
>>> list1=[10,20,30]
>>> list1
[10, 20, 30]
>>> list1+[40,50,60]
[10, 20, 30, 40, 50, 60]
```

iii) The * operator repeats a list for the given number of times.

```
>>> list2 ['A', 'B']
>>> list2 *2
['A', 'B', 'A', 'B']
```

iv) We can insert one item at a desired location by using the method insert()

```
>>> list1
[10, 20]
>>> list1.insert(1,30)
>>> list1
[10, 30, 20]
```

d) Deleting List: To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know.

i) 'del' keyword: We can delete one or more items from a list using the keyword del. Itcan even delete the list entirely. But it does not store the value for further use.

Example:

```
>>> list=[10,20,30,40,50]
>>> del list[2]
 >>> list
[10, 20, 40, 50]
```

ii) remove() method: We use the remove() method if we know the item that we wantto remove or delete from the list (but not the index)

Example:

```
>>> list=[10,20,30,40,50]
>>> list.remove(30)
>>> list
[10, 20, 40, 50]
```

VIII. 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		

IX. Resources used (Additional)

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

X. Practical related Questions

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

1. When to used list
2. Describe various list functions.
3. Write syntax for a method to sort a list
4. Write syntax for a method to count occurrences of a list item in Python
5. How to concatenate list
6. Justify the statement “Lists are mutable”

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. 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 sum all the items in a list.
2. Write a Python program to multiplies all the items in a list.
3. Write a Python program to get the largest number from a list.
4. Write a Python program to get the smallest number from a list.
5. Write a Python program to reverse a list.
6. Write a Python program to find common items from two lists.
7. Write a Python program to select the even items of a list.

(Space for answers)

XII. References / Suggestions for further Reading

1. https://www.tutorialspoint.com/Python/Python_lists
2. https://www.javatpoint.com/Python-lists
3. https://www.youtube.com/watch?v=9rLdQP3g4fw

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	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

Like list python supports new tuple as a distinct structure. Tuple are used to store sequence of python objects which are static in nature. A tuple is immutable which meansit cannot be changed. Just like a list, a tuple contains a sequence of objects in order but once created a tuple, it cannot be changed anything about it.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem. The practical is expected to develop the following skills:

- Write a Python Program using tuple

IV. Relevant Course Outcome(s)

CO 2- Perform operations on sequence structures in python.

V. Practical Outcome (PrOs)

Write Python program to perform following operations on Tuples:

1. Create Tuple
2. Access Tuple
3. Update Tuple
4. Delete Tuple

VI. Relevant Affective Domain related Outcomes

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

VII. Minimum Theoretical Background

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists.The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

a) Creating a Tuple: Creating a tuple is as simple as putting different comma- separated values. Optionally you can put these comma-separated values between parentheses also.

Example

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
```

The empty tuple is written as two parentheses containing nothing

```
tup1 = ()
```

To write a tuple containing a single value you have to include a comma, even though there is only one value.

```
tup1 = (50,)
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and soon.

b) Accessing Values in Tuples

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

Example:

```
#!/usr/bin/Python
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print "tup1[0]: ", tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

Output:

```
tup1[0]: physics
tup2[1:5]:  [2, 3, 4, 5]
```

c) Updating Tuples: Tuples are immutable which means you cannot update or changethe values of tuple elements. You are able to take portions of existing tuples to create new tuples.

Example:

```
#!/usr/bin/Python
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')
# Following action is not valid for tuples
# tup1[0] = 100
# So let's create a new tuple as follows
tup3 = tup1 + tup2
print tup3
```

Output:

```
(12, 34.56, 'abc', 'xyz')
```

d) Delete Tuple Elements: Removing individual tuple elements is not possible. To explicitly remove an entire tuple, just use the **del** statement.

Example:

```
#!/usr/bin/Python
tup = ('physics', 'chemistry', 1997, 2000)
print tup
del tup
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Define empty tuple. Write syntax to create empty tuple.
2. Write syntax to copy specific elements existing tuple into new tuple.
3. Compare tuple with list (Any 4 points)

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. 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 tuple and find the minimum and maximum number from it.
2. Write a Python program to find the repeated items of a tuple.
3. Print the number in words for Example: 1234 => One Two Three Four

(Space for answers)

.....

.....

.....

.....

.....

XII. References / Suggestions for further Reading

1. https://www.tutorialspoint.com/Python/Python_loops.htm
2. https://www.youtube.com/watch?v=NI26dqhs2Rk
3. https://www.youtube.com/watch?v=fAw8pM_dQP4

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	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

A set is an unordered collection of items. Every element is unique (no duplicates) and must be immutable (which cannot be changed). A set is created by placing all the items (elements) inside curly braces {}, separated by comma or by using the built-in function set(). It can have any number of items and they may be of different types (integer, float, tuple, string etc.). This practical will make students to work on Set data structure supported in Python.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem. The practical is expected to develop the following skills:

- Write a program using Set and performs operations like create, access, update and delete

IV. Relevant Course Outcome(s)

CO 2- Perform operations on sequence structures in python.

V. Practical Outcome (PrOs)

Develop Python program to perform following operations on Sets: Create, Access,Update and Delete set

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Mathematically a set is a collection of items not in any particular order. A Python set issimilar to this mathematical definition with below additional conditions.

- The elements in the set cannot be duplicates.
- The elements in the set are immutable (cannot be modified) but the set as a whole is mutable.
- There is no index attached to any element in a Python set. So they do not supportany

indexing or slicing operation.

The sets in Python are typically used for mathematical operations like union, intersection, difference and complement etc.

a) Creating a set:

A set is created by using the set() function or placing all the elements within a pair ofcurly braces.

Example:

```
>>> a={1,3,5,4,2}
>>> print("a=",a)
a= {1, 2, 3, 4, 5}
>>> print(type(a))
<class 'set'>
```

b) Accessing values in a set: We cannot access individual values in a set. We can only access all the elements together. But we can also get a list of individual elements by looping through the set.

Example:

```
Num=set([10,20,30,40,50])
for n in Num:
 print(n)
```

Output:

```
10
20
30
40
50
```

c) Updating items in a set: We can add elements to a set by using add() method. There is no specific index attached to the newly added element.

Example:

```
Num=set([10,20,30,40,50])
Num.add(60)
print(Num)
```

Output:

```
{10,20,30,40,50,60}
```

d) Removing items in set:

We can remove elements from a set by using discard() method. There is no specificindex attached to the newly added element.

Example:

```
Num=set([10,20,30,40,50])
Num.discard(50)
print(Num)
```

Output:

```
{10,20,30,40}
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Describe the various set operations.
2. Describe the various methods of set.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. 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 create a set, add member(s) in a set and remove one item from set.
2. Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.
3. Write a Python program to find maximum and the minimum value in a set.
4. Write a Python program to find the length of a set.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

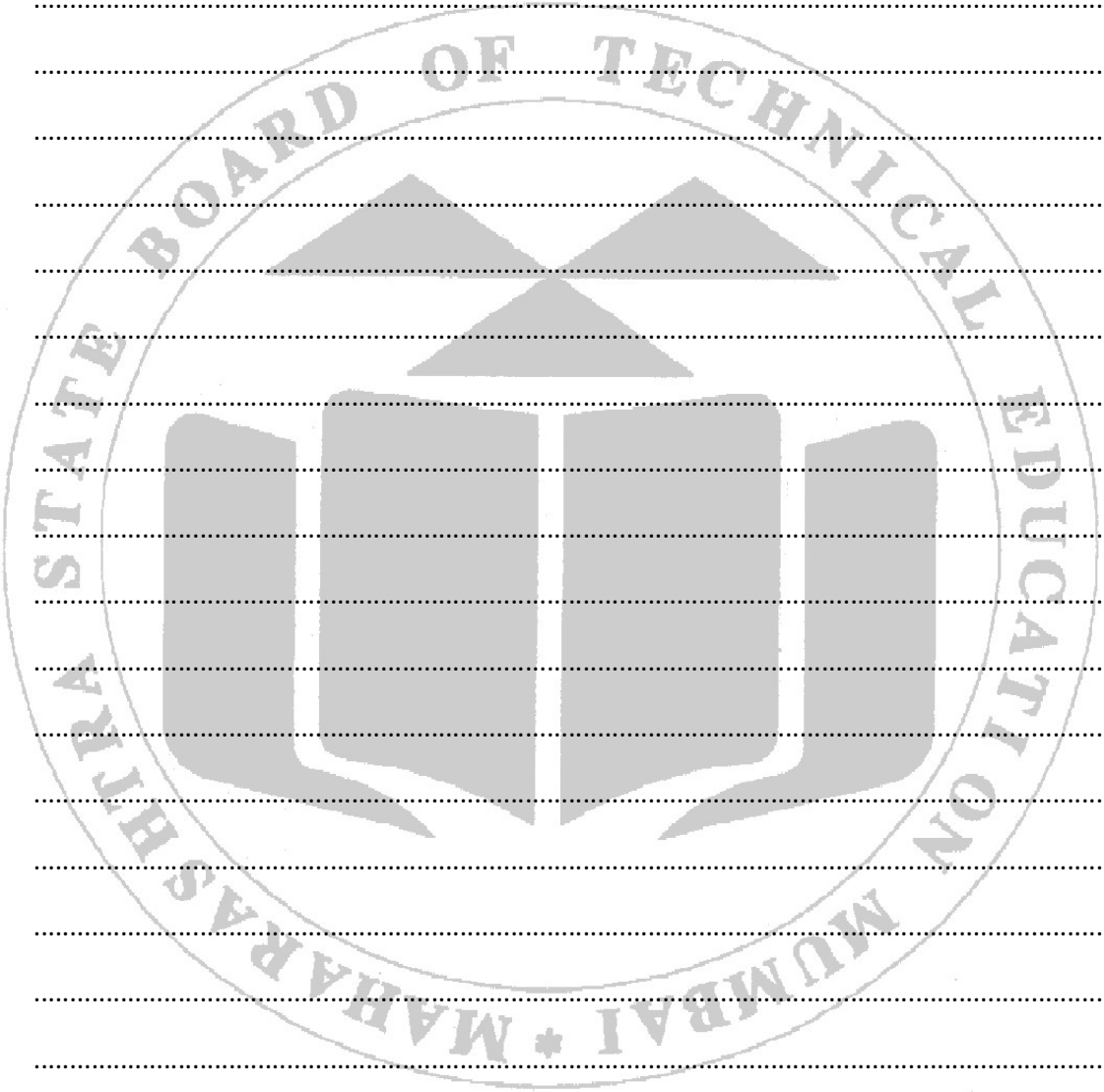
.....

.....

.....

.....

.....



XII. References / Suggestions for further Reading

- https://www.w3schools.com/Python/Python_sets.asp
- https://www.geeksforgeeks.org/sets-in-Python/
- https://www.tutorialspoint.com/Python/Python_sets
- https://www.youtube.com/watch?v=MEPlLAjPvXY
- https://www.youtube.com/watch?v=r3R3h5ly_8g

XIII. Assessment Scheme

	Performance indicators	Weightage
	Process related (15 Marks)	**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
	Product related (10 Marks)	**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	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

As Java supports hash-map, Python supports Dictionary data structure. This data structure will let user store a data with in a form of key value pair. In this a key is immutable and value can be any python object which can store any data. One can find relation between key and value which can get desired value. The dictionary is the data type in Python which can simulate the real-life data arrangement where some specific value exists for some particular key. This practical will enable student to work on dictionary and demonstrate work of key value pair.

II. Relevant Program Outcomes (POs)

a) **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems

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

c) **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem. The practical is expected to develop the following skills:

a) Write a Python Program using Dictionary

b) Develop simple application using Dictionary

IV. Relevant Course Outcome(s)

CO 2- Perform operations on sequence structures in python.

V. Practical Outcome(PrOs)

Write Python program to perform following operations on Dictionaries:

1. Create Dictionary
2. Access Dictionary elements
3. Update Dictionary
4. Delete Dictionary
5. Looping through Dictionary

VI. Relevant Affective Domain related Outcomes

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

VII. Minimum 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 beimmutable objects like strings or numbers. They must also be unique within a dictionary.

a) Creating the Dictionary

The dictionary can be created by using multiple key-value pairs enclosed with the small brackets () and separated by the colon (:). The collections of the key-value pairs are enclosed within the curly braces {}.

The syntax to define the dictionary is given below.

```
#!/usr/bin/Python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

b) Accessing Values in Dictionary: To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value.

Example:

```
#!/usr/bin/Python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

Output:

```
dict['Name']:    Zara
dict['Age']: 7
```

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:

```
#!/usr/bin/Python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8;          # update existing entry
dict['School'] = "DPS School";      # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Output:

```
dict['Age']:  8
dict['School']:  DPS School
```

d) Delete Dictionary Elements: You can either remove individual dictionary elements or

clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement.

Example:

```
#!/usr/bin/Python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

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

del dict ;        # delete entire dictionary
```

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 iterationof the for loop, you will get both the keys its relevant values in the output.

Example

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

for key, value in dict.items():

    print(key, ' - ', value)
```

The above example contains both the keys and the values in the output. The text 'Relatedto' in the output showing the given key is related to the given value in the output.

```
Name - Zara
Age - 7
Class - First
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. What is the output of the following program?

```
dictionary = {  'MSBTE' : 'Maharashtra State Board of Technical Education',
                'CO' : 'Computer Engineering',
                'SEM' : 'Sixth'
              }
  del dictionary['CO'];
  for key, values in dictionary.items():
      print(key)
  dictionary.clear()
  for key, values in dictionary.items():
      print(key)
  del dictionary
  for key, values in dictionary.items():
      print(key)
```

2. What is the output of the following program?

```
dictionary1 = {  'Google' : 1,
          'Facebook' : 2,
          'Microsoft' : 3
         }
        dictionary2 = { 'GFG' : 1,
         'Microsoft' : 2,
         'Youtube' : 3
         }
       dictionary1.update(dictionary2)
       for key, values in dictionary1.items():
            print(key, values)
```

3. What is the output of the following program?

```
temp = dict()
temp['key1'] = {'key1' : 44, 'key2' : 566}
temp['key2'] = [1, 2, 3, 4]
for (key, values) in temp.items():
  print(values, end = "")
```

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. 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 sort (ascending and descending) a dictionary by value.
2. Write a Python script to concatenate following dictionaries to create a new one.
   a. Sample Dictionary:
   b. dic1 = {1:10, 2:20}
   c. dic2 = {3:30, 4:40}
   d. dic3 = {5:50,6:60}
3. Write a Python program to combine two dictionary adding values for common keys.
   a. d1 = {'a': 100, 'b': 200, 'c':300}
   b. d2 = {'a': 300, 'b': 200, 'd':400}
4. Write a Python program to print all unique values in a dictionary.

a. Sample Data: [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

5. Write a Python program to find the highest 3 values in a dictionary.

(Space for answers)

XII. References / Suggestions for further Reading

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

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 10,11 a) Apply math built - in function in Python. b) Develop user defined Python function for given problem: write a function with minimum 2 arguments

I. Practical Significance

Like any other 3rd generation language, Python also supports in-built functions to perform some traditional operations on String and Numbers. The math module is used to access mathematical functions in the Python. All methods of these functions are used for integer or real type objects, not for complex numbers. The math module is a standard module in Python and is always available. To use mathematical functions underthis module, you have to import the module using import math. String functions will let user to develop a code using basic and advance functions which works on string. This practical will let learner to use mathematical and string related functions.
All object-oriented programming languages supports reusability. One way to achieve this is to create a function. Like any other programming languages Python supports creation of functions and which can be called within program or outside program. Reusability and Modularity to the Python program can be provided by a function by calling it multiple times. This practical will make learner use of modularized programming using functions.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.
The practical is expected to develop the following skills:

- Write a program using Math built-in functions
- Write a program using string built-in functions
- Write a Python Program using function that will accept 2 arguments

IV. Relevant Course Outcome(s)

CO 2- Perform operations on sequence structures in python.

V. Practical Outcome (PrOs)

- Write Python program to demonstrate math built- in functions
- Write Python program to demonstrate string built – in functions
- Function with minimum 2 arguments

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Math's built-in Functions: Here is the list of all the functions and attributes defined in math module

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

String build-in functions:

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

Function	Description
capitalize()	Converts the first character to upper case
count()	Returns the number of times a specified value occurs in a string
endswith()	Returns true if the string ends with the specified value
find()	Searches the string for a specified value and returns the position of where it was found
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdigit()	Returns True if all characters in the string are digits
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isspace()	Returns True if all characters in the string are whitespaces
isupper()	Returns True if all characters in the string are upper case
lower()	Converts a string into lower case
replace()	Returns a string where a specified value is replaced with a specified Value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
split()	Splits the string at the specified separator, and returns a list
Strip()	Remove spaces at the beginning and at the end of the string:
startswith()	Returns true if the string starts with the specified value
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code which can be called whenever required.

a) Creating a function: In Python, we can use **def** keyword to define the function.

Syntax:

```
def my_function():
   function-suite
return <expression>
```

b) Calling a function: To call the function, use the function name followed by the parentheses.

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

Output:

hello world

c) Arguments in function: The information into the functions can be passed as the argumenta. The arguments are specified in the parentheses. We can give any number of arguments, but we have to separate them with a comma.

Example

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

Output:

Sum: 5

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Describe about string formatting operator with example.
2. Give the syntax and example of title() and capitalize() methods.
3. Give the syntax and significance of string functions: title() and strip().
4. What is the output of the following

```
def find_sum(*numbers):
    result = 0
        for num in numbers:
        result = result + num
        print("Sum = ", result)
find_sum(1, 2, 3)
find_sum(4, 9)
```

(Space for answers)

XI. 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 function that accepts a string and calculate the number of upper-case letters and lower-case letters.

2. Write a Python program to generate a random float where the value is between 5and 50 using Python math module.
3. Write a Python function that takes a number as a parameter and check the number isprime or not.
4. Write a Python function to calculate the factorial of a number (a non-negativeinteger). The function accepts the number as an argument.

(Space for answers)

XII. References / Suggestions for further Reading

1. https://www.tutorialspoint.com/Python-mathematical-functions
2. https://www.programiz.com/Python-programming/modules/math
3. https://www.geeksforgeeks.org/mathematical-functions-Python-set-1-numeric- functions/
4. https://www.learnpython.org/en/Basic_String_Operations
5. https://www.youtube.com/watch?v=EkYrfV7M1ks
6. https://www.w3schools.com/Python/Python_ref_string.asp
7. https://www.tutorialspoint.com/Python/Python_strings
8. https://www.tutorialspoint.com/Python/Python_functions.htm

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 12, 13 a) Create a program to demonstrate use of built-in module (e.g. numeric, mathematical functional and programming module) in Python.
b) Write program to create a user - defined module (e.g : building calculator) in Python.

I. Practical Significance

While developing a computer application, more often than not the line of code (LOC) can go beyond developer's control. Such codes are tightly coupled and hence are difficult to manage. Also, one cannot reuse such code in other similar application. Python supports modularized coding which allow developers to write a code in smaller blocks. A module can define functions, classes and variables. A module can also include runnable code. By using module students will be able to group related code that will make the code easier for them to understand and use.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Write a Python Program to create built-in module
- Write a Python Program to create user defined module

IV. Relevant Course Outcome(s)

CO 3- Implement modules, packages in python for given problem.

V. Practical Outcome(PrOs)

- Built-in module (e.g. keyword, math, number, operator)
- User defined module

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Built-in Modules

Built-in modules are written in C and integrated with the Python interpreter. Each built- in module contains resources for certain system-specific functionalities such as OS

management, disk IO, keyword, math, number, operator etc. The standard library also contains many Python scripts (with the .py extension) containing useful utilities.

To display a list of all available modules, use the following command in the Python console:

>>> help('modules')

a) Python - Math Module

Some of the most popular mathematical functions are defined in the math module. These include trigonometric functions, representation functions, logarithmic functions, angle conversion functions, etc. In addition, two mathematical pie and Euler's number constants are also defined in this module.

Example

```
>>> import math
>>>math.pi
3.141592653589793
```

b) Python - OS Module

It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

Example

We can create a new directory using the **mkdir()** function from the OS module.

```
>>> import os
>>>os.mkdir("d:\\tempdir")
```

c) Python - Random Module

Functions in the **random** module depend on a pseudo-random number generator function random(), which generates a random float number between 0.0 and 1.0.

Example

```
>>>import random
>>>random.random()
0.645173684807533
```

User-defined Module

The user-defined module is written by the user at the time of program writing.

i) Creation a User-defined Module

To create a module just write a Python code in a file with file extension as.py:

Example

A module in a file with extension as module_name.py in Python is created.

```
def accept_int():
```

```
val = int(input("Please enter any integer integer: "))
print('The integer value is', val)
```

ii) Accessing a User-defined Module

Now we will access the module that we created earlier, by using the import statement.

Example

import the module in Python.

```
import module_name
```

Output

Please enter any integer: 22

The integer value is 22

VIII. 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		

IX. Resources used (Additional)

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

X. 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. What is the output of the following program?

```
import math
print math.sqrt(25)
print math.pi
print math.degrees(2)
print math.radians(60)
print math.sin(2)
print math.cos(0.5)
print math.tan(0.23)
print math.factorial(4)
```

2. What is the output of the following program?

```
import random
```

```
print random.randint(0, 5)
print random.random()
print random.random() * 100
List = [1, 4, True, 800, "Python", 27, "hello"]
print random.choice(List)
```

3. What is the output of the following program?

```
import datetime
from datetime import date
import time
print time.time()
print date.fromtimestamp(454554)
```

(Space for answers)

XI. 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 create a user defined module that will ask your college name and will display the name of the college.
2. Write a Python program that will calculate area and circumference of circle using inbuilt Math Module
3. Write a Python program that will display Calendar of given month using Calendar Module

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

XII. References / Suggestions for further Reading

1. https://www.tutorialsteacher.com/Python/Python-builtin-modules
2. https://www.tutorialspoint.com/Python/Python_modules
3. https://www.youtube.com/watch?v=7GXaobCrBb4
4. https://www.youtube.com/watch?v=dtGsLWfYnKY

XIII. Assessment Scheme

	Performance indicators	Weightage
	Process related (15 Marks)	**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
	Product related (10 Marks)	**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 14, 15 a) Develop Python program to demonstrate use of NumPy packages for creating, accessing and performing different array operations.
b) Write a program to create user defined packages in Python.

I. Practical Significance

Though Python is simple to learn language but it also very strong with its features. As mentioned earlier Python supports various built in packages. Apart from built-in package user can also make their own packages i.e. User Defined Packages. **Numpy** is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. This practical will allow students to write a code.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Write a Python Program to create built-in package
- Write a Python Program to create user defined package

IV. Relevant Course Outcome(s)

CO 4- Design classes for given problem.

V. Practical Outcome (PrOs)

- Built-in packages (e.g. NumPy, Pandas, Matplotlib)
- User defined package

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Steps for Installing numpy in windows OS

1. goto Command prompt
2. run command pip install numpy
3. open IDLE Python Interpreter
4. Check numpy is working or not

```
>>> import numpy
>>> import numpy as np
>>> a=np.array([10,20,30,40,50])
>>> print(a)
[10 20 30 40 50]
```

Example:

```
>>> student=np.dtype([('name','S20'),('age','i1'),('marks','f4')])
>>> a=np.array([('Vijay',43,90),('Prasad',38,80)],dtype=student)
>>> print(a)
[('Vijay', 43, 90.) ('Prasad', 38, 80.)]
```

Example:

```
>>> print(a)
[10 20 30 40 50 60]
>>> a.shape=(2,3)
>>> print(a)
[[10 20 30]
 [40 50 60]]
>>> a.shape=(3,2)
>>> print(a)
[[10 20]
 [30 40]
 [50 60]]
```

Creating and accessing a Python Package

Steps to create package in Python

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.py
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 Mahendra
Scorpio
Bolero
Xylo
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Describe Numpy Array
2. Why should we use Numpy rather than Matlab, Octave or Yorick?

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. 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 create two matrices and perform addition, subtraction, multiplication and division operation on matrix.
2. Write a Python program to concatenate two strings.
3. Write a NumPy program to generate six random integers between 10 and 30.

(Space for answers)

XII. References / Suggestions for further Reading

1. https://www.tutorialspoint.com/numpy/
2. https://www.geeksforgeeks.org/create-access-Python-package/
3. http://cs231n.github.io/Python-numpy-tutorial/

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 16 Write a program in Python to demonstrate following operations: (a) Method overloading (b) Method overriding

I. Practical Significance

While developing a large program with advance features of object-oriented programming one need to make use of more than one function having same name but they perform different task altogether. This will improve in readability of program but at time creates confusion in invoking desired function. Fortunately, Python supports overloading and overriding of a function which allows developers to have different functions with same name having same or different arguments. This practical will let learner to develop programs using method overloading and overriding concept of python.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Write a Python Program based on Method Overloading
- Write a Python Program based on Method Overriding

IV. Relevant Course Outcome(s) : CO 4- Design classes for given problem.

V. Practical Outcome (PrOs)

Write a program in Python to demonstrate following operations:

1. Method overloading
2. Method Overriding

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Method overloading

To overload a method in Python, we need to write the method logic in such a way that depending upon the parameters passed, a different piece of code executes inside the function. Take a look at the following example:

```
Class Student:
   Def hello(self, name=None):
        if name is not None:
             print('Hey' +name)
        else:
            print('Hey')
#Creating a class instance
        std = Student()
#Call the method
        std.hello()
#Call the method and pass a parameter
        std.hello('Prasad')
```

Output

```
Hey
Hey Prasad
```

Method Overriding

The method overriding in Python means creating two methods with the same name but differ in the programming logic. The concept of Method overriding allows us to change or override the Parent Class function in the Child Class.

Example

```
#Python Method Overriding
 class Employee:
    def message(self):
       print('This message is from Employee Class')
  class Department(Employee):
   def message(self):
       print('This Department class is inherited from Employee')
 emp = Employee()
emp.message()
print('.....')
dept=Department()
dept.message()
```

Output

```
This message is from Employee Class
.....
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. State the difference between method overriding and overloading
2. What is the output of the following program?

```
# parent class
class Animal:
 #properties
        multicellular = True
        #Eukaryotic means Cells with Nucleus
        eukaryotic = True
        #function breath
        def breathe(self):
            print("I breathe oxygen.")
       # function feed
        def feed(self):
            print("I eat food.")
#child class
class Herbivorous(Animal):
#function feed
      def feed(self):
          print("I eat only plants. I am vegetarian.")
herbi = Herbivorous()
herbi.feed()
#calling some other function
herbi.breathe()
```

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a Python program to create a class to print an integer and a character with two methods having the same name but different sequence of the integer and the character parameters. For example, if the parameters of the first method are of the form (int n, char c), then that of the second method will be of the form (char c, int n)
2. Write a Python program to create a class to print the area of a square and a rectangle. The class has two methods with the same name but different number of parameters. The method for printing area of rectangle has two parameters which are length and breadth respectively while the other method for printing area of square has one parameter which is side of square.
3. Write a Python program to create a class 'Degree' having a method 'getDegree'

that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

(Space for answers)

.....

.....

.....

XII. References/Suggestions for further Reading

1. https://stackabuse.com/overloading-functions-and-operators-in-Python/
2. https://www.tutorialgateway.org/method-overriding-in-Python/
3. https://www.youtube.com/watch?v=HIkoX4pVmrI

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 17 Write a program in Python to demonstrate following operations: (a) Single Inheritance (b) Multilevel Inheritance (c) Multiple Inheritance (d) Hybrid Inheritance (e) Hierarchical Inheritance

I. Practical Significance

Inheritance is a mechanism in which one class (derived class) acquires the property of another class (base class). With inheritance, we can reuse the variables and methods of the existing class. The existing class is called base class and the new class is called derived class. Hence, inheritance facilitates reusability and is an important concept of object-oriented programming. Types of inheritance are: single inheritance, multiple inheritance, multi-level inheritance, hierarchical inheritance and hybrid inheritance.
This practical will let learner to develop programs using inheritance concept of python.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.
The practical is expected to develop the following skills:

a) Write a Python Program based on Inheritance

IV. Relevant Course Outcome(s) : CO 4- Design classes for given problem.

V. Practical Outcome (PrOs)

Write a program in Python to demonstrate following operations:

1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance
4. Hybrid Inheritance
5. Hierarchical Inheritance

VI. Relevant Affective Domain related Outcome(s)

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

Minimum Theoretical Background

Single Inheritance

- The mechanism of designing or constructing classes from other classes is called Single Inheritance. The new class is called derived class or child class & the class from which this derived class has been inherited is the base class or parent class.
- In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class.

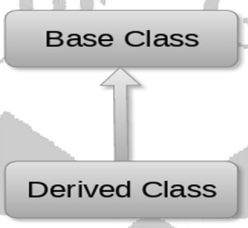


Fig. No. 12.1: Single Inheritance

Syntax:

```
class BaseClass1
  #Body of base class
    class DerivedClass (BaseClass1):
  #Body of derived class
```

Multilevel Inheritance

In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class. This is similar to a relationship representing a child and a grandfather

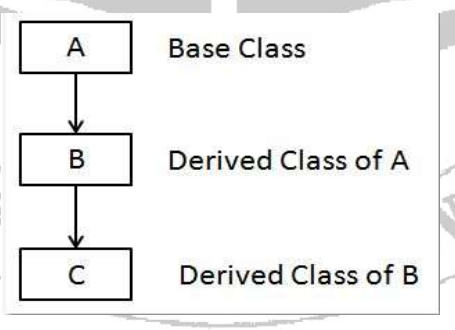


Fig. No. 12.2: Multilevel Inheritance

Syntax:

```
class baseclass:
        # Base class code here
class derivedclass1(baseclass):
        # Derived class 1 code here
class derivedclass2(derivedclass1):
        # Derived class 2 code here
```

Multiple Inheritance

When a class derived from two or more base classes, such inheritance is called Multiple Inheritance. It allows us to combine the features of several existing classes into a single class.

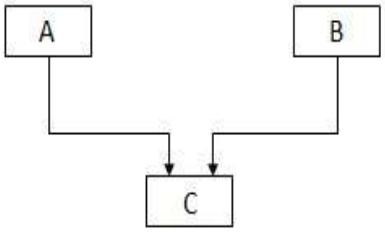


Fig. No. 12.3: Multiple Inheritance

Syntax:

```
class baseclass1:
        # Base class 1 code here
class baseclass2:
        # Base class 2 code here
class derivedclass (baseclass1, baseclass2):
        # Derived class code here
```

Hierarchical Inheritance

Hierarchical Inheritance is a type of inheritance in which a single base class is inherited by multiple derived classes. In this scenario, each derived class shares common attributes and methods from the same base class, forming a hierarchy of classes.

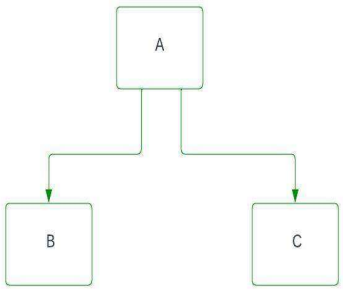


Fig. No. 12.4: Hierarchical Inheritance

Syntax:

```
class baseclass:
        # Base class code here
class derivedclass1 (baseclass):
        # Derived class 1 code here
class derivedclass2 (baseclass):
        # Derived class 2 code here
```

Hybrid Inheritance

Hybrid inheritance is a blend of multiple inheritance types. In Python, the supported types of inheritance are single, multiple, multilevel, hierarchical, and hybrid. In hybrid inheritance, classes are derived from more than one base class, creating a complex inheritance structure.

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. State the difference between Multiple and Multilevel Inheritance.
2. State the use of inheritance
3. List different types of inheritance

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a Python program to implement different types of inheritance.

(Space for answers)

.....

.....

.....

.....

.....

XI. References/Suggestions for further Reading

1. https://www.w3schools.com/Python/Python_inheritance.asp
2. https://www.geeksforgeeks.org/inheritance-in-Python/

XII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 18 Write a program in python for handling array to demonstrate following operations: a) Array declaration, b) Insertion, c) Deletion, d) Append, e) Index, f) Reverse.

I. Practical Significance

Python is simple to learn language but it also very strong with its features. It provides a high-performance multidimensional array object, and tools for working with arrays. An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. Arrays are an important data structure for any programming language. Python uses arrays to store collections of similar data, saving space and time. This practical will allow students to write a code using array.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Write a Python Program for handling array to demonstrate the operations of arrays.

IV. Relevant Course Outcome(s): CO 5- Implement linear data structure in python.

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

1. Array operations such as insertion, deletion, append, reverse.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Creating an array in Python

Array in Python can be created by importing an array module.

array(data_type, value_list) is used to create array in Python with data type and value list specified in its arguments.

Syntax:

```
from array import *
        arrayName = array(typecode, [initializers])
```

Example:

```
import array as arr
        a = arr.array('i', [2, 4, 5, 6])
```

Accessing an array in Python

We can access the array elements using the respective indices of those elements.

Example:

```
import array as arr
a = arr.array('i', [2, 4, 5, 6])
print("First element is:", a[0])
print("Second element is:", a[1])
print("last element is:", a[-1])
```

Output:

```
First element is: 2
Second element is: 4
last element is: 6
```

Basic Array Operations:

- **Insertion**

Insert operation is used to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of the built-in array.insert() function in Python.

Example:

```
import array as ar
num = ar.array('i', [1,2,3,4,5])
num.insert(1,9)
```

- **Deletion**

Deletion refers to removing an existing element from the array and re-organizing all elements of an array. The built-in remove() method is there in Python.

Example:

```
import array as ar
num = ar.array('i', [1, 2, 3, 5, 7, 10])
num.remove(7)
```

In the pop() method, you specify the index of the element that you want to remove.

- **Append**

We can add one item to the array using the append() method, or add several items using the extend() method.

Example:

```
import array as ar
num = ar.array('i', [1, 2, 3])
```

```
num.append(4)
print(num)
num.extend([5, 6, 7])
print(num)
```

- **Index**

  The index() method returns the position at the first occurrence of the specified value.

  Example:

```
fruits = [4, 55, 64, 32, 16, 32]
x = fruits.index(32)
```

- **Reverse**

  The reverse() method reverses the sorting order of the elements.

  Example:

```
array = [1, 2, 3, 4, 5]
array.reverse()
print(array)
```

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Enlist the operations performed by using array in Python?

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a Python program to append a new item to the end of the array.
2. Write a Python program to reverse the order of the items in the array.
3. Write a Python program to remove a specified item using the index of an array.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XII. References/Suggestions for further Reading

1. https://www.geeksforgeeks.org/python-arrays/
2. https://www.w3schools.com/python/python_arrays.asp
3. https://www.javatpoint.com/python-arrays

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 19 Write a program in python for linked list to demonstrate following operations: Insertion, deletion, Update, Merging to list

I. Practical Significance

Python is simple to learn language but it also very strong with its features. As mentioned earlier. A linked list is a sequence of data elements, which are connected together via links. This practical will allow students to write a code for insert, delete, merge and searching of elements in the list.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem. The practical is expected to develop the following skills:

- Write a Python Program for Linked list to demonstrate the operations on the list.

IV. Relevant Course Outcome(s): CO 5- Implement linear data structure in python.

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

1. Linked List operations such as Insertion, deletion, update, Merging to list.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Linked List in Python

A linked list is a sequence of data elements, which are connected together via links. Each data element contains a connection to another data element in form of a pointer. Python does not have linked lists in its standard library.

There are various linked list operations that allow us to perform different actions on linked lists. For example, the insertion operation adds a new element to the linked list.

1. **Insertion in a Linked List:** Inserting element in the linked list involves reassigning the pointers from the existing nodes to the newly inserted node. Depending on whether the new data element is getting inserted at the beginning or at the middle or at the end of the linked list. You can add elements to the beginning, middle or end of the linked list.
2. **Delete from a Linked List:** You can delete either from the beginning, end or from a particular position.
3. **Search an Element on a Linked List:** You can search an element on a linked list using a loop.
4. **Traverse a Linked List:** Displaying the contents of a linked list is very simple. We keep moving the temp node to the next one and display its contents.
5. **Merging to list** -Given two sorted linked lists consisting of N and M nodes respectively. The task is to merge both of the lists (in place) and return the head of the merged list.

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Which operations are performed in linked list?
2. Differentiate between Linked List and Array.

(Space for answers)

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a Python program to delete the first item from a singly linked list.
2. Write a Python program to create a singly linked list, append some items and iterate through the list**.**

(Space for answers)

XII. References/Suggestions for further Reading

1. https://www.tutorialspoint.com/python_data_structure/python_linked_lists.
2. https://www.programiz.com/dsa/linked-list-operations
3. https://www.w3resource.com/python-exercises/data structure/python_linked_lists.

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 20: Write a program in python to demonstrate queue data structure operations: a) Enqueue, b) Dequeue, c) Display

I. Practical Significance

A queue is a linear type of data structure used to store the data in a sequentially. The concept of queue is based on the FIFO, which means "**First in First Out**". It is also known as "first come first severed". The queue has the two ends front and rear. The next element is inserted from the **rear** end and removed from the **front** end. This practical will allow students to write a code for insert, delete, traverse, and display the elements of the queue.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.

The practical is expected to develop the following skills:

- Write a Python Program to demonstrate queue data structure.

IV. Relevant Course Outcome(s): CO 5- Implement linear data structure in python.

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

1. Write a program in Python to demonstrate queue data structure.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Queue data structure in Python

A queue is a linear data structure that follows the **First-In-First-Out (FIFO)** principle. It operates like a line where elements are added at one end (**rear**) and removed from the other end (**front**).

Basic Operations of Queue data structure

- Enqueue (Insert): Adds an element to the rear of the queue.
- Dequeue (Delete): Removes and returns the element from the front of the queue.
- Display: Returns the element of the queue without removing it.
- Empty: Checks if the queue is empty.
- Full: Checks if the queue is full.

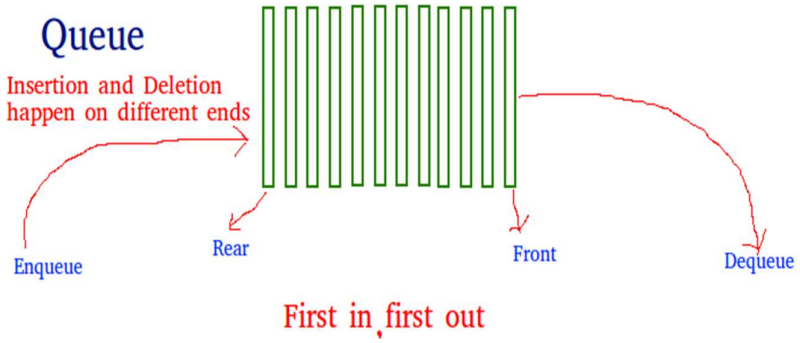


Fig. 15.1 Queue Data Structure

VIII. 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		

IX. Resources used (Additional)

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

X. 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. What is the output of the following program:

```
from queue import Queue
q = Queue(maxsize = 3)
print(q.qsize())
```

```
q.put('a')
q.put('b')
q.put('c')
print("\nFull: ", q.full())
print("\nElements dequeued from the queue")
print(q.get())
print(q.get())
print(q.get())
print("\nEmpty: ", q.empty())
q.put(1)
print("\nEmpty: ", q.empty())
print("Full: ", q.full())
```

2. Enlist the Operations in Queue.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a Python program for implementation of Queue Data Structure.

(Space for answers)

.....

.....

XII. References/Suggestions for further Reading

1. https://www.geeksforgeeks.org/queue-in-python/
2. https://www.makeuseof.com/queue-python-implement-how/
3. https://realpython.com/queue-in-python/
4. https://www.javatpoint.com/queue-in-python

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 21: Use the Searching Techniques in data structures: a) Linear Search, b) Binary Search

I. Practical Significance

In order to perform some operation on specific data element, the data has to be searched in data collection, and the system requires searching methods. Most commonly used methods are Linear Search and Binary Search.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.
The practical is expected to develop the following skills:

- Write a Python Program to implement Linear Search.
- Write a Python Program to implement Binary Search.

IV. Relevant Course Outcome(s): CO 5- Implement linear data structure in python.

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

- Write a Python Program to implement Linear Search.
- Write a Python Program to implement Binary Search.

VI. Relevant Affective Domain related Outcome(s)

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

Minimum Theoretical Background

Linear Search in Python

Linear Search is a method for searching an element in a collection of elements. In Linear Search, each element of the collection is visited one by one in a sequential fashion to find the desired element. Linear Search is also known as **Sequential Search.**

How Does Linear Search Algorithm Work?

In Linear Search Algorithm,

- Every element is considered as a potential match for the key and checked for the same.

- If any element is found equal to the key, the search is successful and the index of that element is returned.
- If no element is found equal to the key, the search yields “No match found”.

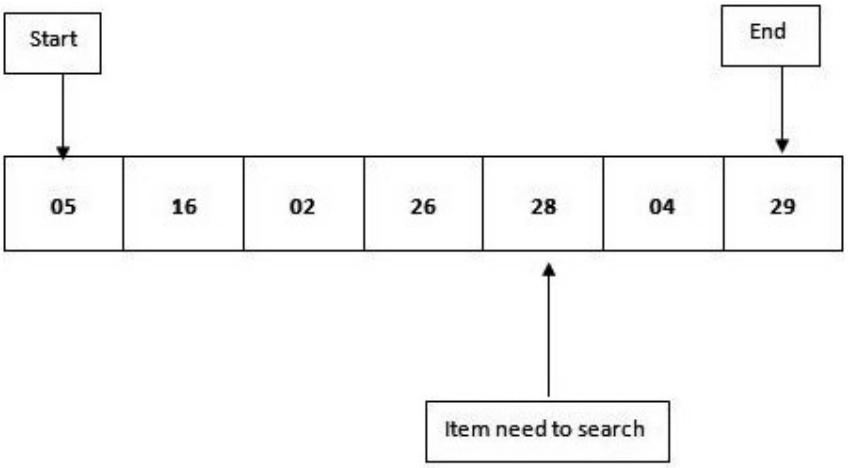


Fig. 16.1 Linear Search

Binary Search in Python

The Binary Search algorithm searches through an array and returns the index of the value it searches for. Binary Search is much faster than Linear Search, but requires a sorted array to work. The Binary Search algorithm works by checking the value in the center of the array. If the target value is lower, the next value to check is in the center of the left half of the array. This way of searching means that the search area is always half of the previous search area, and this is why the Binary Search algorithm is so fast. This process of halving the search area happens until the target value is found, or until the search area of the array is empty.

In this algorithm,

- Divide the search space into two halves by finding the middle index “mid”

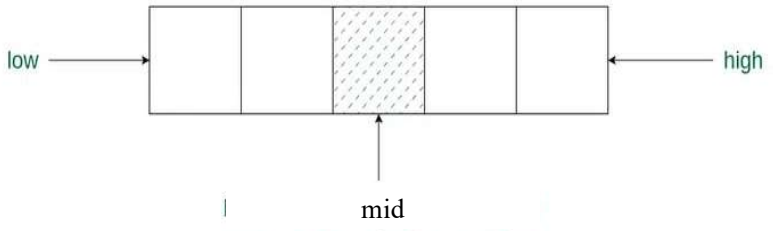


Fig. 16.2 Finding the middle index “mid” in Binary Search Algorithm

- Compare the middle element of the search space with the key.
- If the key is found at middle element, the process is terminated.
- If the key is not found at middle element, choose which half will be used as the next search space.
- If the key is smaller than the middle element, then the left side is used for next search.
- If the key is larger than the middle element, then the right side is used for next search.

- This process is continued until the key is found or the total search space is exhausted.

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. Differentiate between Linear Search and Binary Search.
2. Write a program in python to implement Linear Search and Binary Search.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

X. Exercise

Note: Faculty must ensure that every student use different input value.

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

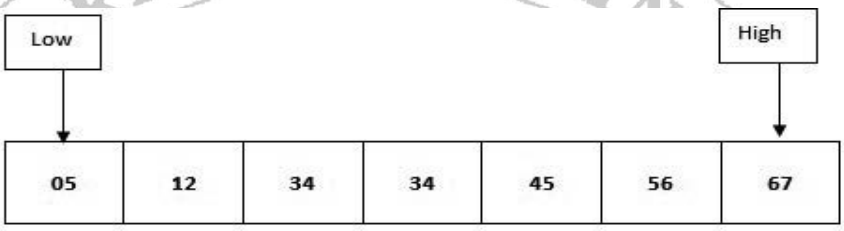
1.Consider the following list to perform the Linear Search.

56 36 89 56 01 00 67 59

a) Search the item 01 from the above list and write the item is found or not with procedure.

b) Search the item 55 from the above list and write the item is found or not with procedure.

2. Consider a following list and write down the steps to perform binary search to search the following numbers. a) 12, b) 67, c) 100



(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. References/Suggestions for further Reading

1. https://www.geeksforgeeks.org/python-program-for-linear-search/
2. https://www.javatpoint.com/linear-search-in-python
3. https://www.w3schools.com/dsa/dsa_algo_linearsearch.php
4. https://www.geeksforgeeks.org/binary-search/
5. https://www.w3schools.com/dsa/dsa_algo_binarysearch.php

XII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 22 Write the python program to implement following sorting techniques: a) Bubble Sort b) Insertion Sort

I. Practical Significance

In order to perform some operation on specific data element, the data has to be sorted in data collection, and the system requires sorting methods. Most commonly used methods are Bubble Sort and Insertion Sort for sorting data from the given list.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.
The practical is expected to develop the following skills:

- Write a Python Program to implement Bubble Sort.
- Write a Python Program to implement Insertion Sort.

IV. Relevant Course Outcome(s): CO 5- Implement linear data structure in python.

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

- Write a Python Program to implement Bubble Sort.
- Write a Python Program to implement Insertion Sort.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Bubble Sort in Python

In Bubble Sort method the list is rearranged by exchanging the two adjacent elements if they are not in order (order may be ascending or descending).

Bubble sort algorithm starts by comparing the first two elements of an array and swapping, if necessary, i.e., if you want to sort the elements of array in ascending order and if the first element is greater than second then, the elements are swapped but, if the first element is smaller than second, elements are not swapped. Then, again second and third elements are compared and swapped if it is necessary and this process continues until last and second last element are compared and swapped. This completes the first iteration of bubble sort.

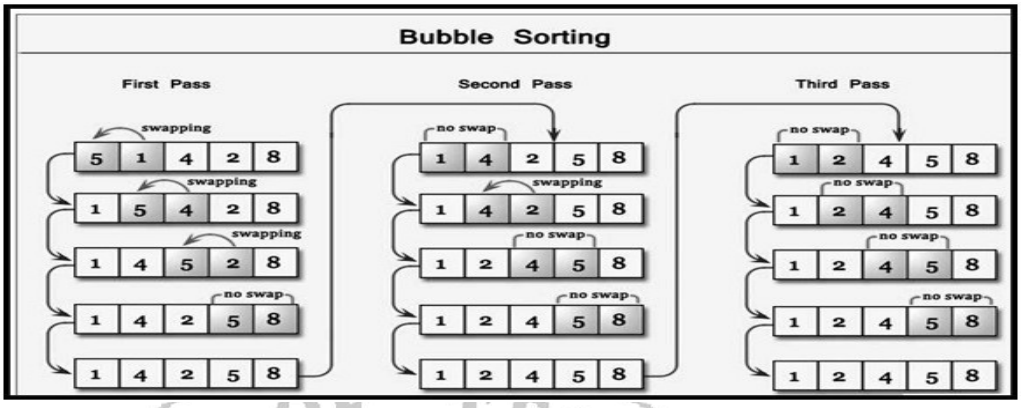


Fig.17.1 Bubble Sort

Insertion Sort in Python

Insertion Sort is a simplest data Sorting algorithm which sorts the array elements by shifting elements one by one and inserting each element into its proper position. Consider we sort playing cards in our hands.

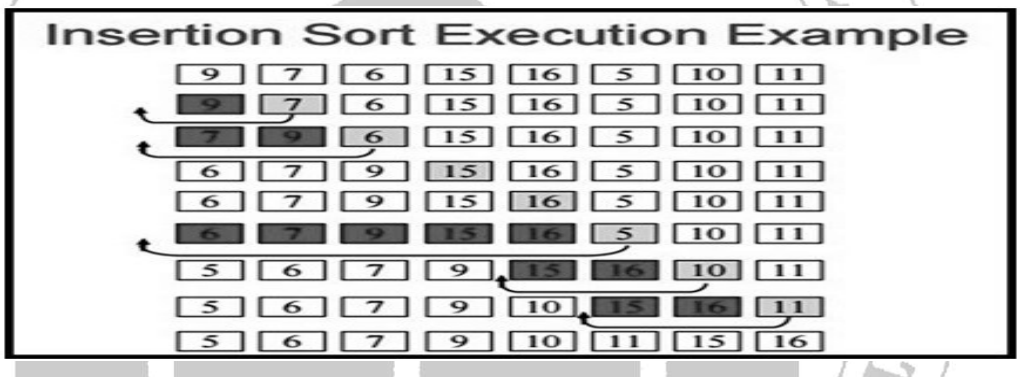


Fig.17.2 Insertion Sort

VIII. 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		

IX. Resources used (Additional)

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

X. 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 in python to implement Bubble Sort & Insertion Sort.
2. Differentiate between Bubble Sort & Insertion Sort.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Sort the given list in ascending order using Bubble Sort. Show each iteration. 105, 18, 62, 48, 01, 23, 11
2. Sort the given list in ascending order using Insertion Sort. Show each iteration. 42, 35, 68, 12, 16, 15

(Space for answers)

.....

.....

XII. References/Suggestions for further Reading

1. https://www.javatpoint.com/bubble-sort-in-python
2. https://www.programiz.com/dsa/bubble-sort
3. https://www.geeksforgeeks.org/bubble-sort-algorithm/
4. https://www.geeksforgeeks.org/python-program-for-insertion-sort/
5. https://www.programiz.com/dsa/insertion-sort
6. https://www.javatpoint.com/insertion-sort-in-python

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 23: Write a program in python to evaluate an expression.

I. Practical Significance

Evaluate an expression represented by a String. For evaluation of arithmetic expressions following methods of stack are required- infix, prefix and postfix representation

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem.
The practical is expected to develop the following skills:

- Write a Python Program to evaluate an expression using stack.

IV. Relevant Course Outcome(s): CO 5- Implement linear data structure in python. CO 6- Develop a python program to implement tree data structure.

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

- Write a Python Program to evaluate an expression using stack.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Evaluate an expression using stack

This is well known that the computer system can understand and work only on binary paradigm. In which an arithmetic operation can take place between two operands only like A + B, C * D, D / A. generally an arithmetic expression may consist of more than one operator and two operands, for example (A + B) * (D / (J + D)). Such form of the arithmetic expression is commonly known as the infix expression. Normally the evaluation of the any arithmetic expression does not take place in its infix form. Here we are introducing the method for evaluating the infix expression form computation point of view. The stack is found to be more efficient to evaluate an infix arithmetical expression by first converting to a prefix or postfix expression and then evaluating these converted expressions. This approach will eliminate the repeated scanning of an infix expression in order to obtain its value. Therefore, there are three basic notations are referred for the representation of any complex arithmetic expression.

These forms are as follows:

a) If the operator symbols are placed before its operands, then the expression is in prefix form.

b) If the operator symbols are placed after its operands, then the expression is in postfix form.

c) If the operator symbols are placed between the operands, then the expression is in infix form.

Hence, a normal arithmetic expression is normally called as infix expression i.e. A+B. A Polish mathematician found a way to represent the same expression called polish notation or prefix expression by keeping operators as prefix i.e. +AB. We use the reverse way of the above expression for our evaluation. The representation is called Reverse Polish Notation (RPN) or postfix expression i.e. AB+.

VIII. 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		

IX. Resources used (Additional)

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

X. 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. Define Polish Notation.
2. Convert the expression (A + B) / (C – D) into postfix expression and then evaluate it for A = 10, B = 20, C = 15, D = 5. Display the stack status after each operation.

(Space for answers)

.....

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a program in python to evaluate an expression.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XII. References/Suggestions for further Reading

1. https://www.geeksforgeeks.org/expression-evaluation/
2. https://www.scribd.com/document/550953100/4-a-Evaluation-of-an-expression-using-stack

XIII. Assessment Scheme

	Performance indicators	Weightage
	Process related (15 Marks)	**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
	Product related (10 Marks)	**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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

Practical No. 24 Write a program to create binary tree from the given list using Binary tree module in Python.

I. Practical Significance

Tree is a non-linear data structure. It is a hierarchical data structure that has nodes connected through links. The topmost node of the tree which has no parent is known as the **root** node. **Binary Tree** is a form of a tree whose nodes cannot have more than two children. Each node of the binary tree has two pointers associated with it, one points to the left child, and the other points to the right child of the node. It is an unordered tree having no fixed organized structure for the arrangement of nodes. This practical will allow students to write a code for Binary tree module in Python.

II. Relevant Program Outcomes (POs)

- **Basic and Discipline specific knowledge**: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems
- **Engineering Tools, Experimentation and Testing**: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- **Life-long learning**: Ability to analyze individual needs and engage in updating in the context of technological changes.

III. Competency and Practical Skills

Develop general purpose programming using Python to solve problem. The practical is expected to develop the following skills:

- Write a Python Program to demonstrate Binary Search Tree (BST).

IV. Relevant Course Outcome(s): CO 6- Develop a python program to implement tree data structure

V. Practical Outcome (PrOs)

Write Python program to demonstrate use of:

1. Write a program in Python to demonstrate Binary Tree Operations.

VI. Relevant Affective Domain related Outcome(s)

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

VII. Minimum Theoretical Background

Binary Tree in Python

A binary tree is a data structure in which every node or vertex has at most two children. In Python, a binary tree can be represented in different ways with different data structures (dictionary, list) and class representations for a node. However, binary tree library helps to directly implement a binary tree. It also supports heap and binary search tree (BST). This module does not come pre-installed with Python's standard utility module. To install it type the below command in the terminal.

pip install binarytree

Creating Node:

The node class represents the structure of a particular node in the binary tree. The attributes of this class are values, left, right.

Syntax: binarytree.Node(value, left=None, right=None)

Parameters:

value: Contains the data for a node. This value must be number.

left: Contains the details of left node child.

right: Contains details of the right node child.

Operations on a Binary Search Tree:

The following operations are performed on a binary search tree.

1. Search
2. Insertion
3. Deletion
4. Traversing

Build a binary tree from the List:

Instead of using the Node method repeatedly, we can use build() method to convert a list of values in to a binary tree. Here, a given list contains the nodes of tree such that the element at index i has its left child at index 2*i+1, the right child at index 2*i+2 and parent at (i – 1)//2. The elements at index j for j>len(list)//2 are leaf nodes. None indicates the absence of a node at that index. We can also get the list of nodes back after building a binary tree using values attribute.

Syntax:

binarytree.build(values)

Parameters:

values: List representation of the binary tree. Returns: root of the binary tree.

VIII. 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		

IX. Resources used (Additional)

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

X. 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 to implement tree traversal techniques.

(Space for answers)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI. Exercise

Note: Faculty must ensure that every student use different input value.

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

1. Write a Python program to check whether a given binary tree is valid binary search tree or not.
2. Write a Python program to create a binary tree from given list using Binary Tree module.

(Space for answers)

XII. References/Suggestions for further Reading

1. https://www.pythonpool.com/adjacency-list-python/#google_vignette
2. https://www.w3schools.com/python/python_arrays.asp
3. https://www.programiz.com/dsa/graph-adjacency-list#google_vignette
4. https://www.geeksforgeeks.org/print-adjacency-list-for-a-directed-graph/
5. https://www.educative.io/blog/8-python-data-structures

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (10 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (25 Marks)	**100%**

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