I

Name……………………………………………………………………………..
Roll No. …………………………….Year 20………..20……………
Exam Seat No.……………………………………………………………

COMPUTER GROUP | SEMESTER – VI | DIPLOMA IN ENGINEERING AND TECHNOLOGY

A LABORATORY MANUAL FOR PROGRAMMING WITH PYTHON (22616)

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

A Laboratory Manual for

Programming with Python

(22616)

Semester– VI

(CO/CW/CM/IF)

Maharashtra State

Board of Technical Education, Mumbai

(Autonomous)(ISO:9001:2015) (ISO/IEC27001:2013)

Maharashtra State

Board of Technical Education, Mumbai

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

4th Floor, Government Polytechnic Building, 49, Kherwadi,

Bandra (East), Mumbai -400051.

(Printed on November 2019)

Maharashtra State
Board of Technical Education

Certificate

This is to certify that Mr. / Ms. …………………………………. Roll No……………………… of Sixth Semester of Diploma in …………………………………………………. of Institute …………………………………………………………...... (Code…………) has completed the term work satisfactorily in subject **Programming with Python (22616)** for the academic year 20…….to 20…...... as prescribed in the curriculum.

Place ……………….	Enrollment No……………………
Date:	Exam Seat No. ………………….....

Subject Teacher **Head of the Department** **Principal**

Seal of
the
Institute

Preface

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

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

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

Python is a high-level, interpreted, reflective, dynamically typed, open source, multi-paradigm and general purpose programming language. It is quite powerful and easy. It offers no special tools or features that let you do things that you cannot do with other languages, but its elegant design and combination of certain features make Python a pleasure to use.

Although all care has been taken to check for mistakes in this laboratory manual, yet it is impossible to claim perfection especially as this is the first edition. Any such errors and suggestions for improvement can be brought to our notice and are highly welcome.

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

Following programme outcomes are expected to be achieved significantly out of the ten programme outcomes and Computer Engineering programme specific outcomes through the practical's of the course on **Programming with Python**

PO1	**Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
PO2	**Discipline knowledge:** Apply Computer engineering discipline-specific knowledge to solve core computer engineering related problems.
PO3	**Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
PO4	**Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
PO5	**The engineer and society:** Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to practice in field of Computer engineering.
PO6	**Environment and sustainability:** Apply Computer engineering solutions also for sustainable development practices in societal and environmental contexts and demonstrate the knowledge and need for sustainable development.
PO7	**Ethics:** Apply ethical principles for commitment to professional ethics, responsibilities and norms of the practice also in the field of Computer engineering.
PO8	**Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams.
PO9	**Communication:** Communicate effectively in oral and written form.
PO10	**Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

Practical- Course Outcome matrix

Course Outcomes (COs)

a) Display message on screen using Python script on IDE.
b) Develop Python program to demonstrate use of Operators
c) Perform operations on data structures in Python.
d) Develop functions for given problem.
e) Design classes for given problem.
f) Handle exceptions.

Sr. No.	Title of the Practical	CO a.	CO b.	CO c.	CO d.	CO e.	CO f.
1	Install and configure Python IDE	√	-	-	-	-	-
2	Write simple Python program to display message on screen	√	-	-	-	-	-
3	Write simple Python program using operators: a) Arithmetic Operators b) Logical Operators c) Bitwise Operators	-	√	-	-	-	-
4	Write simple Python program to demonstrate use of conditional statements: a) ‘if’ statement b) ‘if … else’ statement c) Nested ‘if’ statement	-	√	-	-	-	-
5	Write Python program to demonstrate use of looping statements: a) ‘while’ loop b) ‘for’ loop c) Nested loops	-	√	-	-	-	-
6	Write Python program to perform following operations on Lists: a) Create list b) Access list c) Update list (Add item, Remove item) d) Delete list	-	-	√	-	-	-
7	Write Python program to perform following operations on Tuples: a) Create Tuple b) Access Tuple c) Update Tuple d) Delete Tuple	-	-	√	-	-	-
8	Write Python program to perform following operations on Tuples: a) Create Set b) Access Set elements c) Update Set d) Delete Set	-	-	√	-	-	-

9	Write Python program to perform following operations on Dictionaries: a) Create Dictionary b) Access Dictionary elements c) Update Dictionary d) Delete Set e) Looping through Dictionary	-	-	√	-	-	-
10	a) Write Python program to demonstrate math built- in functions (Any 2 programs) b) Write Python program to demonstrate string built – in functions (Any 2 programs)	-	-	-	√	-	-
11	Develop user defined Python function for given problem: a) Function with minimum 2 arguments b) Function returning values	-	-	-	√	-	-
12	Write Python program to demonstrate use of: a) Built in module (e.g. keyword, math, number, operator) b) user defined module.	-	-	-	√	-	-
13	Write Python program to demonstrate use of: a) built-in packages (e.g. NumPy, Pandas) b) user defined packages	-	-	-	√	-	-
14	Write a program in Python to demonstrate following operations: a) Method overloading b) Method overriding	-	-	-	-	√	-
15	Write a program in Python to demonstrate following operations: a) Simple inheritance b) Multiple inheritance	-	-	-	-	√	-
16	Write a program in Python to handle user defined exception for given problem	-	-	-	-	-	√

List of Industry Relevant Skills

The following industry relevant skills of the competency “Develop general purpose programming using Python to solve problem” are expected to be developed in you by performing practicals of this laboratory manual.

1. Develop Applications using Python.
2. Write and Execute Python programs using functions, classes and Exception handling

Brief Guidelines to Teachers

Hints regarding strategies to be used

1. Teacher shall explain prior concepts to the students before starting each experiment.
2. For practical’s requiring tools to be used, teacher should provide the demonstration of the practical emphasizing the skills, which the student should achieve.
3. Involve students in the activities during the conduct of each experiment.
4. Teachers should give opportunity to students for hands-on after the demonstration.
5. Assess the skill achievement of the students and COs of each unit.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
8. Teacher may provide additional knowledge and skills to the students even though that may not be covered in the manual but are expected from the students by the industries.
9. Teacher may suggest the students to refer additional related literature of the reference books/websites/seminar proceedings etc.
10. During assessment teacher is expected to ask questions to the students to tap their knowledge and skill related to that practical.

Instructions for Students

Student shall read the points given below for understanding the theoretical concepts and practical applications.

1. Students shall listen carefully the lecture given by teacher about importance of subject, learning structure, course outcomes.
2. Students shall organize the work in the group of two or three members and make a record of all observations.
3. Students shall understand the purpose of experiment and its practical implementation.
4. Students shall write the answers of the questions during practical.
5. Student should feel free to discuss any difficulty faced during the conduct of practical.
6. Students shall develop maintenance skills as expected by the industries.
7. Student shall attempt to develop related hands on skills and gain confidence.
8. Students shall refer technical magazines; websites related to the scope of the subjects and update their knowledge and skills.
9. Students shall develop self-learning techniques.
10. Students should develop habit to submit the write-ups on the scheduled dates and time.

Content Page

List of Practical's and Progressive Assessment Sheet

S. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks(50)	Dated sign. of teacher	Remarks (if any)
	elements c) Update Dictionary d) Delete Set e) Looping through Dictionary						
10.	a) Write Python program to demonstrate math built- in functions (Any 2 programs) b) Write Python program to demonstrate string built – in functions (Any 2 programs)	61					
11.	Develop user defined Python function for given problem: a) Function with minimum 2 arguments b) Function returning values	67					
12.	Write Python program to demonstrate use of: a) Builtin module (e.g. keyword, math, number, operator) b) user defined module.	73					
13.	Write Python program to demonstrate use of: a) built-in packages (e.g. NumPy, Pandas) b) user defined packages	80					
14.	Write a program in Python to demonstrate following operations: a) Method overloading b) Method overriding	86					
15.	Write a program in Python to demonstrate following operations: a) Simple inheritance b) Multiple inheritance	92					
16.	Write a program in Python to handle user defined exception for given problem	98					
				Total Marks			
				Total Marks (Scaled to 25 Marks)			

Practical No. 1: Install and configure Python IDE

I. Practical Significance

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

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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

IV. Relevant Course Outcome(s)

Display message on screen using Python script on IDE.

V. Practical Outcomes(PrOs)

Setup a Python programming development environment

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

Installing Python in Windows:

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

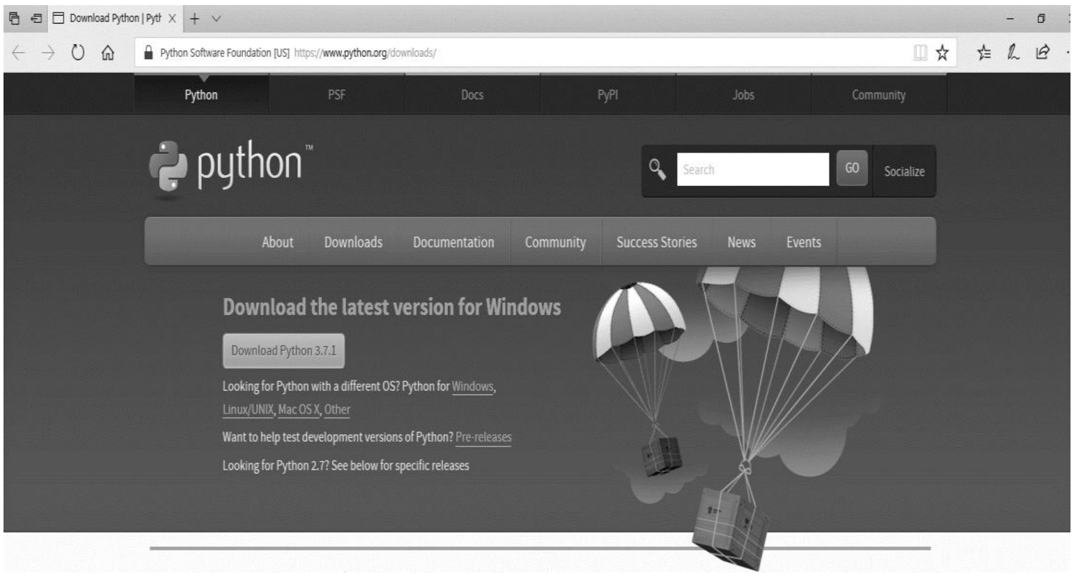
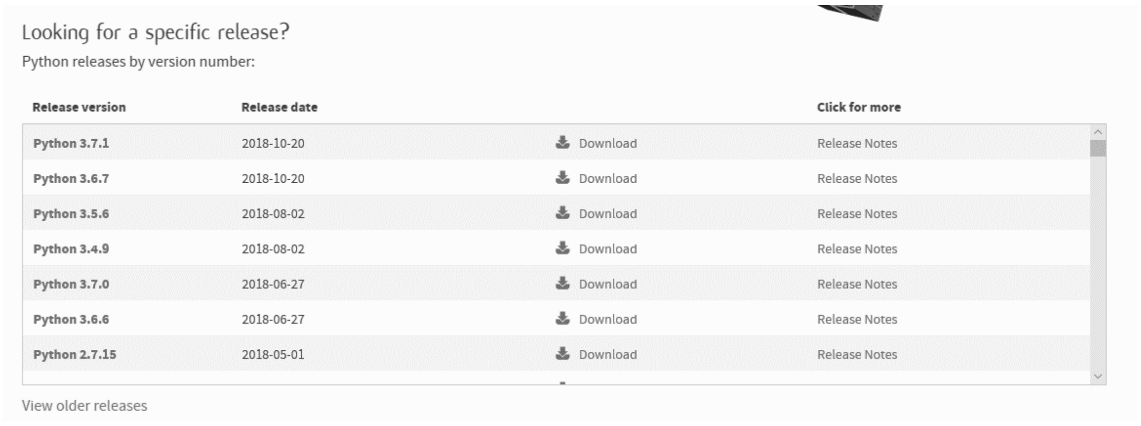

Download Python | Pyth
Python Software Foundation [US] https://www.python.org/downloads/
Python
PSF
Docs
PyPI
Jobs
Community
python
Search
GO
Socialize
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Download Python 3.7.1
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other
Want to help test development versions of Python? Pre-releases
Looking for Python 2.7? See below for specific releases

Fig. 1: Home Page

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

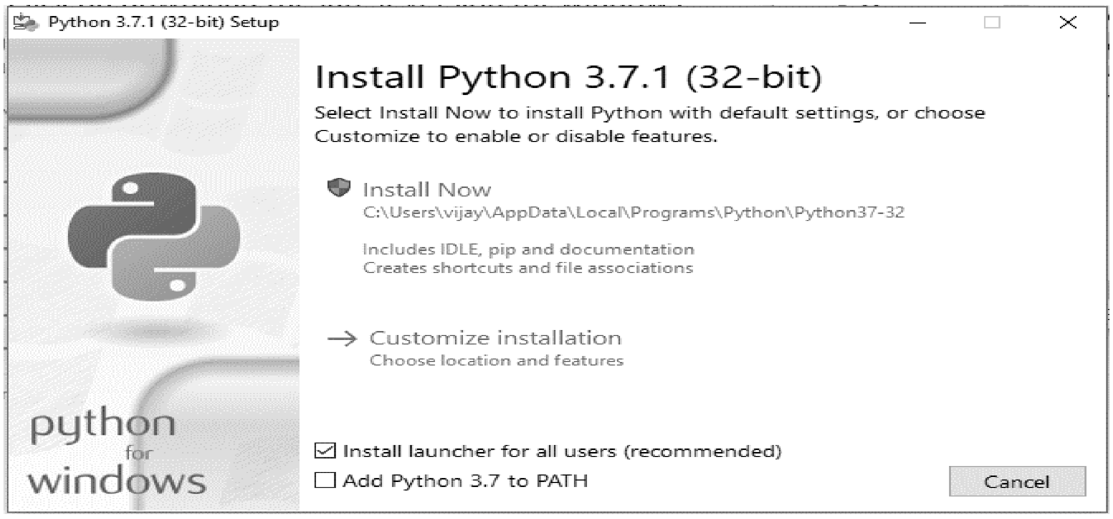

Looking for a specific release?
Python releases by version number:

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

View older releases

Fig.2: Python release versions

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


Python 3.7.1 (32-bit) Setup

Install Python 3.7.1 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\vijay\AppData\Local\Programs\Python\Python37-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

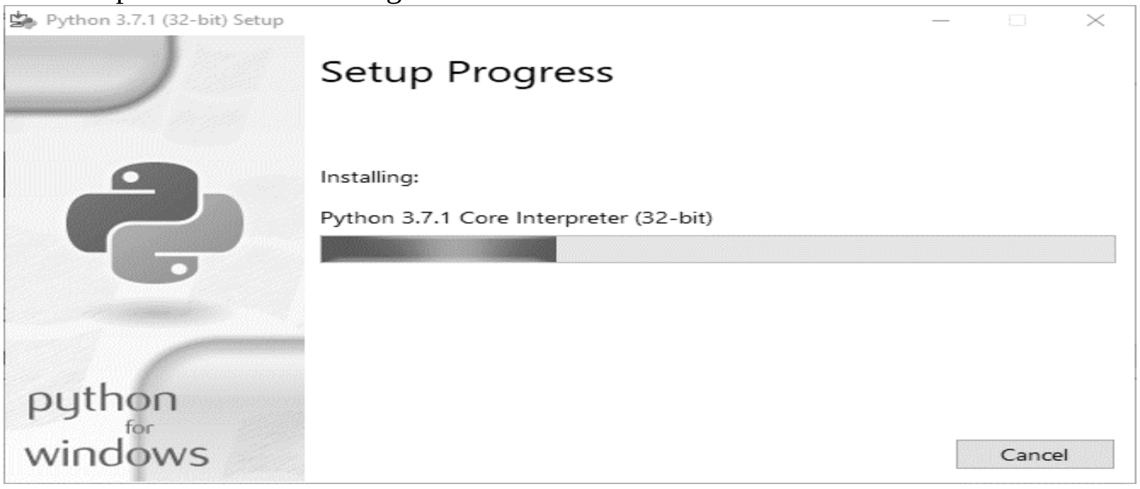
Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.7 to PATH

Cancel

Fig. 3: Installation Type

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



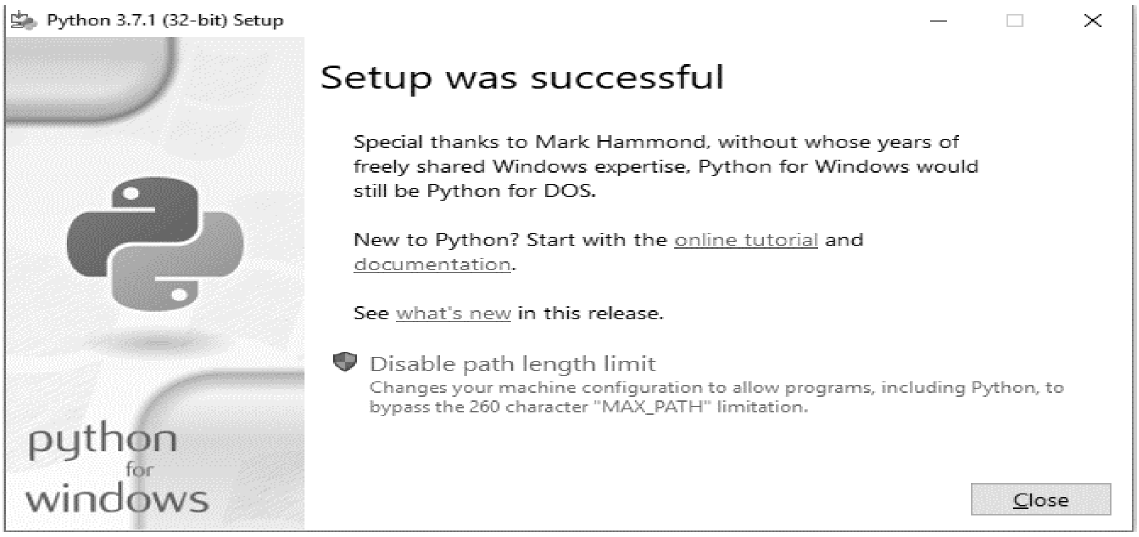
Setup Progress

Installing:

Python 3.7.1 Core Interpreter (32-bit)

Fig. 4: Setup Progress

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



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

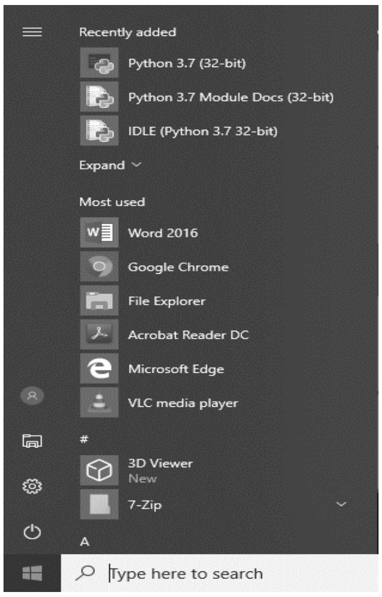
Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Fig. 5: Setup Completion

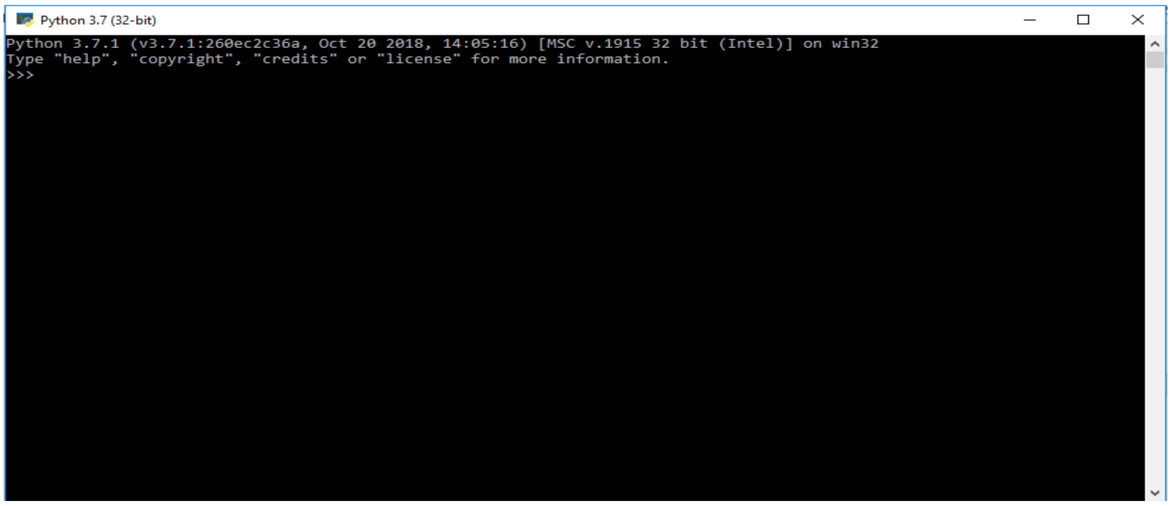
Starting Python in different modes:

1) Starting Python (Command Line)

- Press start button

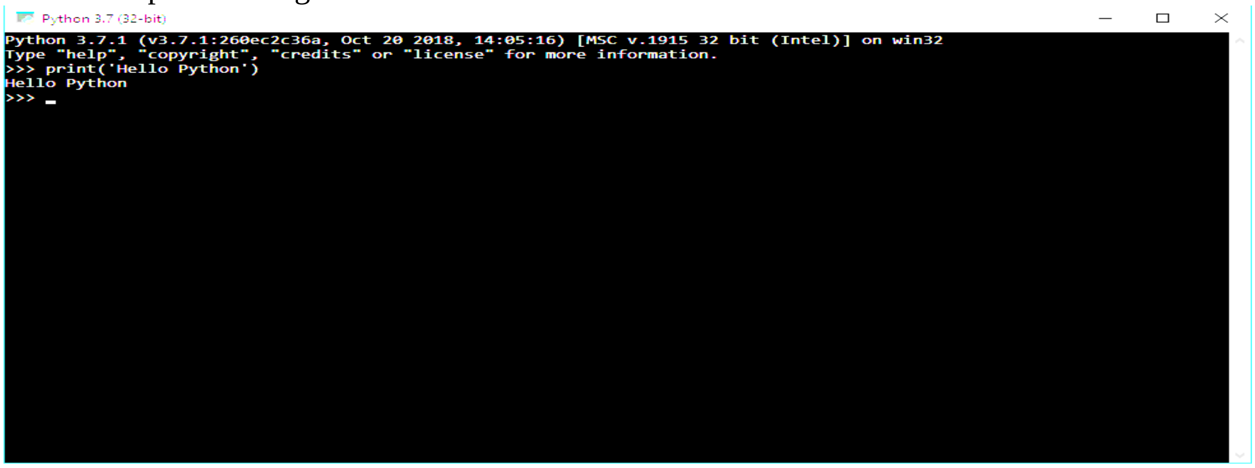


- Click on all programs and then click on Python 3.7 (32 bit). You will see the Python interactive prompt in Python command line.



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

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

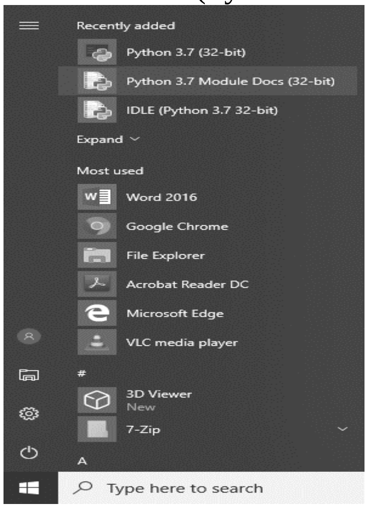


```
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello Python')
Hello Python
>>> _
```

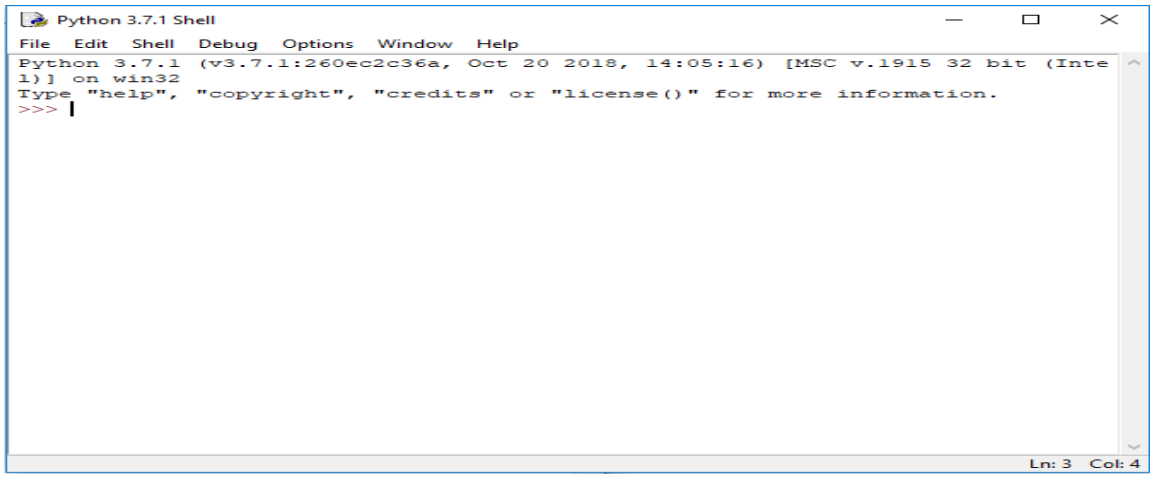
- To exit from the command line of Python, press Ctrl+z followed by Enter or Enter exit() or quit() and Enter.

2) Starting Python IDLE

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

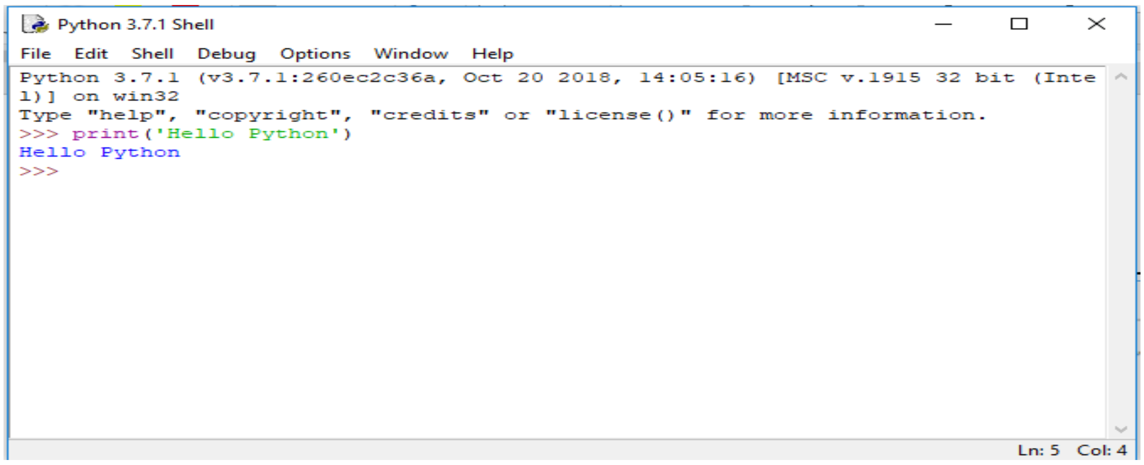


- You will see the Python interactive prompt i.e. interactive shell.



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

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



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

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	Pyhton 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.
2. List key features of Python
3. Explain Python Path
4. State use of pep and pip

(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.

(Space for answers)

XII. References / Suggestions for further Reading

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

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1.	Logic Formulation	10%
2.	Debugging Ability	20%
3.	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4.	Expected output	10%
5.	Timely Submission of report	10%
6.	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 2: Write simple Python program to display message on screen

I. Practical Significance

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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

III. Competency and Practical Skills

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

1. Write a simple Python Program using Interactive mode.
2. Write a simple Python Program using Script mode.

IV. Relevant Course Outcome(s)

Display message on screen using Python script on IDE.

V. Practical Outcome (PrOs)

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

Different modes for executing Python program.

Interactive Mode Programming

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

```
$ Python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
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:

```
$ Python test.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 +x test.py     # This is to make file executable
$./test.py
```

This produces the following result –

```
Hello, Python!
```

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	Pyhton 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. List different modes of Programming in Python
2. Describe procedure to execute program using Interactive Mode
3. State the steps involved in executing the program using Script Mode
4. State the procedure to make file executable

(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 display your name using Interactive Mode
2. Write a Python program to display “MSBTE” using Script Mode

(Space for answers)

XII. References / Suggestions for further Reading

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

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 3: Write simple Python program using operators: Arithmetic Operators, Logical Operators, Bitwise 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 keywords or special characters. Here are the types of operators supported by Python:

- Arithmetic Operators
- Assignment Operators
- Relational or Comparison Operators
- Logical Operators
- Bitwise Operators
- Identity Operators
- Membership 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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 Arithmetic, Logical and Bitwise Operators

IV. Relevant Course Outcome(s)

Develop Python program to demonstrate use of Operators

V. Practical Outcomes(PrOs)

Develop Python program using operators: Arithmetic Operators, Logical Operators, 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	Output

and	Logical AND	If both the operands are true then condition becomes true.	>>>(8>9) and (2<9)
			>>>(9>8)and(2<9)
				False
				True

or	Logical OR	If any of the two operands are non-zero then condition becomes true.	>>>(2==2) or (9<20)
			>>>(3!=3) or (9>20)
				True
				False

not	Logical NOT	Used to reverse the logical state of its operand.	>>>not(8>2) >>>not (2 > 10)	False True

Bitwise Operators: Bitwise operators acts 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	The left operand's value is moved	>>>(a<<2)	40

	shift	left by the number of bits specified by the right operand.		

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	Pyhton 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)

....

....

....

....

....

....

....

....

....

....

....

....

....

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)

....

....

....

....

....

....

XI. References / Suggestions for further Reading

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

XII. Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 4: Write simple Python program to demonstrate use of conditional statements: if’ statement, ‘if … else’ statement, Nested ‘if’ statement

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

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 conditional statements

IV. Relevant Course Outcome(s)

Develop Python program to demonstrate use of Operators

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 means 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 true
   if  (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
```

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	Pyhton 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. 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 find out absolute value of an input number
3. Write a program to check the largest number among the three numbers
4. Write a program to check if the input year is a leap year of not
5. Write a program to check if a Number is Positive, Negative or Zero
6. Write a program that takes the marks of 5 subjects and displays the grade.

(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

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 5: Write Python program to demonstrate use of looping statements: ‘while’ loop, ‘for’ loop and Nested loop

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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Environment and sustainability:** Apply Computer engineering solutions also for sustainable development practices in societal and environmental contexts and demonstrate the knowledge and need for sustainable development.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 ‘while’ loop
- Write a Python Program using ‘for’ loop
- Develop a Python Program using Nested loop

IV. Relevant Course Outcome(s)

Develop Python program to demonstrate use of Operators

V. Practical Outcomes(PrOs)

Write Python program to demonstrate use of looping statements:

- ‘while’ loop
- ‘for’ loop
- Nested loops

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!
```

b) 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) block is 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!
```

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

Syntax

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

The syntax for a nested while loop statement in Python programming language is as follows –

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

A final note on loop nesting is that you can put any type of loop inside of any other type of loop. For example a for loop can be inside a while loop or vice versa.

Example

The following program uses a nested for loop to find the prime numbers from 2 to 100

```
#!/usr/bin/Python
i = 2
while(i < 20):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   if (j > i/j) : print i, " is prime"
   i = i + 1
print "Good bye!"
```

Output:

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
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	Pyhton 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. 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 loop:

a.
```
*
**
***
****
```

b.
```
  *
 ***
*****
 ***
  *
```

c.
```
1010101
 10101
  101
   1
```

2. Write a Python program to print all even numbers between 1 to 100 using while loop.
3. Write a Python program to find the sum of first 10 natural numbers using for loop.
4. Write a Python program to print Fibonacci series.
5. Write a Python program to calculate factorial of a number
6. Write a Python Program to Reverse a Given Number
7. Write a Python program takes in a number and finds the sum of digits in a number.
8. Write a Python program that takes a number and checks whether it is a palindrome or not.

(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=IS7AAVl77kA

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 6: Write Python program to perform following operations on Lists: Create list, Access list, Update list (Add item, Remove item), Delete list

I. Practical Significance

A list can be defined as a collection of values or items of different types. The items in the 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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Ethics:** Apply ethical principles for commitment to professional ethics, responsibilities and norms of the practice also in the field of Computer engineering.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 List and performs operations like create, access, delete. statements

IV. Relevant Course Outcome(s)

Perform operations on data 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 list is a collection which is ordered and changeable.
- A list is a collection of items or elements; the sequence of data in a list is ordered.

- The elements or items in a list can be accessed by their positions i.e. indices
- 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.
- Lists are mutable. The value of any element inside the list can be changed at any point of time.
- 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 along with 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 or multiple elements of lists by giving the slice on the left-hand side of the 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 Operator: We can delete one or more items from a list using the keyword del. It can 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 Operator: We use the remove operator if we know the item that we want to 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	Pyhton 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. When to used list
2. Describe various list functions.
2. Write syntax for a method to sort a list
3. Write syntax for a method to count occurrences of a list item in Python
4. How to concatenate list
5. Justify the statement “Lists are mutable”
6. Describe the use pop operator in list

(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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 7: Write Python program to perform following operations on Tuples: Create Tuple, Access Tuple, Update Tuple, Delete Tuple

I. Practical Significance

Like list python supports new tuple as a distinct structure. Tuple are use to store sequence of python objects which are static in nature. A tuple is immutable which means it 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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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)

Perform operations on data 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 so on.

b) Accessing Values in Tuples

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

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 change the 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. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded. 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	Pyhton 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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 8: Write Python program to perform following operations on Set: Create Set, Access Set elements, Update Set, Delete Set

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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **The engineer and society:** Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to practice in field of Computer engineering.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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)

Perform operations on data 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 is similar 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 support any 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 of curly 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 specific index 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	Pyhton 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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 9: Write Python program to perform following operations on Dictionaries: Create Dictionary, Access Dictionary elements, Update Dictionary, Delete Dictionary, 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)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **The engineer and society:** Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to practice in field of Computer engineering.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 Dictionary
- Develop simple application using Dictionary

IV. Relevant Course Outcome(s)

Perform operations on data 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 be immutable 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 iteration of 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 'Related to' 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	Pyhton 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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 10:

- **Write Python program to demonstrate math built-in functions (Any 2 programs)**
- **Write Python program to demonstrate string built-in functions (Any 2 programs)**

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 under this 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.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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

IV. Relevant Course Outcome(s)

Develop functions for given problem.

V. Practical Outcome(PrOs)

- Write Python program to demonstrate math built- in functions (Any 2 programs)
- Write Python program to demonstrate string built – in functions (Any 2 programs)

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

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	Pyhton 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().

(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 5 and 50 using Python math module.

(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

XIII. Assessment Scheme

	Performance indicators	**Weightage**
	Process related (35 Marks)	**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
	Product related (15 Marks)	**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 11: Develop user defined Python function for given problem:

- **Function with minimum 2 arguments**
- **Function returning values**

I. Practical Significance

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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Ethics:** Apply ethical principles for commitment to professional ethics, responsibilities and norms of the practice also in the field of Computer engineering.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 function that will accept 2 arguments
- Write a Python Program using function that will return a value

IV. Relevant Course Outcome(s)

Develop functions for given problem.

V. Practical Outcome(PrOs)

Develop user defined Python function for given problem:

1. Function with minimum 2 arguments
2. Function returning values

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

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

```
#defining the function
def func (name):
    print("Hi ",name);
  #calling the function
func("ABC")
```

Output:

```
hi ABC
```

d) return Statement: The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Example

```
# Function definition is here
def sum( arg1, arg2 ):
   # Add both the parameters and return them."
   total = arg1 + arg2
   print "Inside the function: ", total
   return total;
# Now you can call sum function
total = sum(10, 20 );
print "Outside the function: ", total
```

Output:

```
Outside the function: 30
```

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	Pyhton 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?

```
def myfunc(text, num):
    while num > 0:
        print(text)
     num = num - 1
myfunc('Hello', 4)
```

2. What is the output of the following program?

```
num = 1
def func():
    num = 3
    print(num)
func()
print(num)
```

(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 takes a number as a parameter and check the number is prime or not.
2. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.
3. Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters.

(Space for answers)

XII. References / Suggestions for further Reading

1. https://www.tutorialspoint.com/Python/Python_functions.htm
2. https://www.javatpoint.com/Python-functions
3. https://www.youtube.com/watch?v=9Os0o3wzS_I
4. https://www.youtube.com/watch?v=NE97ylAnrz4

XIII. Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 12: Write Python program to demonstrate use of:
a) Built-in module (e.g. keyword, math, number, operator)
b) User defined module

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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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)

Develop functions for given problem.

V. Practical Outcome(PrOs)

Write a program in Python to demonstrate use of:

1. Built-in module (e.g. keyword, math, number, operator)
2. 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

a) 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')
```

b) 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
```

c) 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")
```

d) 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
```

e) 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 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	Pyhton 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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 13: Write Python program to demonstrate use of:

- **built-in packages (e.g. NumPy, Pandas)**
- **user defined packages**

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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Environment and sustainability:** Apply Computer engineering solutions also for sustainable development practices in societal and environmental contexts and demonstrate the knowledge and need for sustainable development.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 build-in packages
- Write a Python Program using user defined packages

IV. Relevant Course Outcome(s)

Develop functions for given problem.

V. Practical Outcome(PrOs)

Write Python program to demonstrate use of:

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

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	Pyhton 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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 14: 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 knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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)

Develop functions 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
------------
This Department class is inherited from Employee
```

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	Pyhton 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 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 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 (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 15: Write a program in Python to demonstrate following operations: Simple inheritance, Multiple inheritance

I. Practical Significance

Inheritance allows us to define a class that inherits all the methods and properties from another class. The existing class is called as Parent class or base class and the new class which inherited from base class is called Child class or derived class.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **The engineer and society:** Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to practice in field of Computer engineering.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 on simple inheritance
- Develop program on multiple inheritance

IV. Relevant Course Outcome(s)

Design classes for given problem.

V. Practical Outcome(PrOs)

Write a program in Python to demonstrate following operations:

- Simple inheritance
- Multiple inheritance

VI. Relevant Affective Domain related Outcome(s)

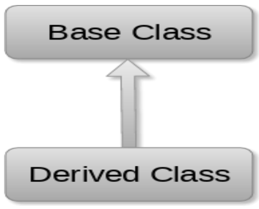
1. Follow safety practices
2. Demonstrate working as a leader / a team member.
3. Follow ethical practices

VII. Minimum Theoretical Background

a) Simple Inheritance:

- The mechanism of designing or constructing classes from other classes is called 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.



Syntax:

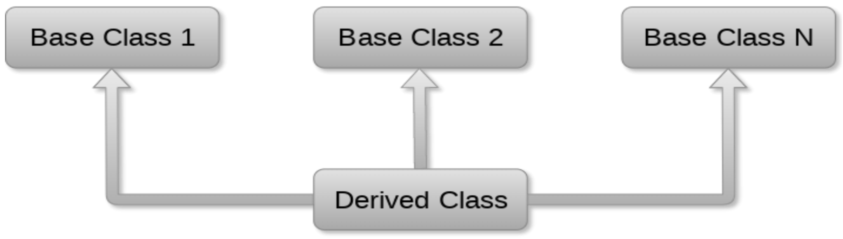
```
class BaseClass1
#Body of base class
class DerivedClass(BaseClass1):
#body of derived - class
```

Example:

```
# Parent class created
class Parent:
    parentname = ""
    childname = ""
     def show_parent(self):
     print(self.parentname)
 # Child class created inherits Parent class
class Child(Parent):
    def show_child(self):
        print(self.childname)
ch1 = Child()                # Object of Child class
ch1.parentname = "Vijay"     # Access Parent class attributes
ch1.childname = "Parth"
ch1.show_parent()            # Access Parent class method
ch1.show_child()             # Access Child class method
```

b) Multiple inheritance

- Python provides us the flexibility to inherit multiple base classes in the child class.
- Multiple Inheritance means that you're inheriting the property of multiple classes into one. In case you have two classes, say A and B, and you want to create a new class which inherits the properties of both A and B.
- So it just like a child inherits characteristics from both mother and father, in Python, we can inherit multiple classes in a single child class.



Syntax:

```
Syntax:
class A:
    # variable of class A
    # functions of class A
class B:
    # variable of class A
    # functions of class A
class C(A, B):
    # class C inheriting property of both class A and B
    # add more properties to class C
```

Example:

```
class Add:
    def Addition(self,a,b):
        return a+b;
class Mul:
    def Multiplication(self,a,b):
        return a*b;
class Derived(Add,Mul):
    def Divide(self,a,b):
        return a/b;
d = Derived()
print(d.Addition(10,20))
print(d.Multiplication(10,20))
print(d.Divide(10,20))
```

Output:

```
30
200
0.5
```

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	Pyhton 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 use of inheritance
2. List different types of inheritance

(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 class Employee with data members: name, department and salary. Create suitable methods for reading and printing employee information
2. Python program to read and print students information using two classes using simple inheritance.
3. Write a Python program to implement multiple inheritance

(Space for answers)

....

....

....

....

....

....

XII. References / Suggestions for further Reading

1. https://www.w3schools.com/Python/Python_inheritance.asp
2. https://www.geeksforgeeks.org/inheritance-in-Python/
3. https://www.javatpoint.com/inheritance-in-Python
4. https://www.youtube.com/watch?v=-hmx1GDqbQE

XIII. Assessment Scheme

Performance indicators		**Weightage**
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	

Practical No. 16: Write a program in Python to handle user defined exception for given problem

I. Practical Significance

Exception are uncovered errors that can occur during run time of a program. Python facilitates its developer to handle such exceptional cases which can then avoid abrupt termination of a program. One can create user-defined or custom exceptions by creating a new exception class in Python. Exceptions need to be derived from the Exception class, either directly or indirectly. This practical will let students to write their own exceptions and avoid program from abrupt termination.

II. Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Ethics:** Apply ethical principles for commitment to professional ethics, responsibilities and norms of the practice also in the field of Computer engineering.
- **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

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 handle user defined exception

IV. Relevant Course Outcome(s)

Handle exceptions

V. Practical Outcome(PrOs)

Write a program in Python to handle user defined exception for given problem

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 has many built-in exceptions which forces your program to output an error when something in it goes wrong. However, sometimes you may need to create custom exceptions that serves your purpose.
A list of common exceptions:

- ZeroDivisionError: Occurs when a number is divided by zero.

- NameError: It occurs when a name is not found. It may be local or global.
- IndentationError: If incorrect indentation is given.
- IOError: It occurs when Input Output operation fails.
- EOFError: It occurs when the end of the file is reached, and yet operations are being performed.

In Python, users can define such exceptions by creating a new class. This exception class has to be derived, either directly or indirectly, from Exception class. Most of the built-in exceptions are also derived from this class.

Example:

```
def input_age(age):
   try:
       assert int(age) > 18
   except ValueError:
       return 'ValueError: Insert integer value'
   else:
       return 'You are Adult'
```

output:

```
>>> print(input_age(25))
You are Adult
>>> print(input_age('abc'))
ValueError: Insert integer value
```

Exception handling in Python

If the Python program contains suspicious code that may throw the exception, we must place that code in the try block. The try block must be followed with the except statement which contains a block of code that will be executed if there is some exception in the try block.

Example:

```
try:
    a = int(input("Enter a:"))
    b = int(input("Enter b:"))
    c = a/b;
    print("a/b = %d"%c)
except Exception:
    print("can't divide by zero")
else:
    print("Hi I am else block")
```

Output:

```
Enter a:10
Enter b:0
can't divide by zero
```

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	Pyhton 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 Exception.
2. How to handle exception in Python?

(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 Check for ZeroDivisionError Exception.
2. Write a Python program to create user defined exception that will check whether the password is correct or not?

(Space for answers)

XII. References / Suggestions for further Reading

1. https://www.geeksforgeeks.org/user-defined-exceptions-Python-examples/
2. https://www.tutorialspoint.com/How-to-implement-user-defined-exception-in-Python
3. https://www.quora.com/How-do-I-create-a-user-defined-exception-in-Python

XIII. Assessment Scheme

Performance indicators		**Weightage**
Process related (35 Marks)		**70%**
1	Logic Formulation	10%
2	Debugging Ability	20%
3	Follow ethical practices	40%
Product related (15 Marks)		**30%**
4	Expected output	10%
5	Timely Submission of report	10%
6	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of student Team Members

1. ……………………..
2. ……………………...
3. ………………………
4. ………………………

Marks Obtained			**Dated signature of Teacher**
Process Related (35)	**Product Related (15)**	**Total (50)**	