SCHEME : K

Name : _____

Roll No. : _____ Year : 20___ 20___

Exam Seat No. : _____

LABORATORY MANUAL FOR

PYTHON PROGRAMMING-314004

COMPUTER ENGINEERING GROUP

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

Vision

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

Mission

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

Quality Policy

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

Core Values

MSBTE believes in the following:

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

A Practical Manual
for

Python Programming
(314004)

Semester-IV

Diploma in Engineering and Technology

(AA/ AT/ BD/ CM/ CO/ CW/ HA/ IF/ IH/ IX/ IZ)

Maharashtra State Board of Technical Education, Mumbai

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

'K' Scheme Curriculum

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

Maharashtra State Board of Technical Education
Certificate

This is to certify that Mr. /Ms. …… Roll No…… of the Fourth Semester of Diploma in …… Engineering/Technology (Program Code - .....4K) of the Institute …… (Inst. Code……) has completed the practical work satisfactorily for the course Python Programming (Course Code: 314004) for the academic year 20….. – 20…..... as prescribed in the curriculum.

Place ……. Enrollment No……

Date: …..... Exam Seat No. ……..…..

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

Seal of the Institute


STATE BOARD OF TECHNICAL EDUCATION MUMBAI • MAHARASHTRA

Preface

Python Programming (314004) laboratory manual is meticulously crafted to equip fourth semester diploma engineering students with valuable practical learning experiences aligned with MSBTE 'K' Scheme Curriculum.

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. To achieve this, each practical is mapped with prescribed theory learning outcomes (TLOs), lab learning outcomes (LLOs) and course outcomes (COs). Course facilitators can adopt suitable pedagogical methods to impart the course with an aim to achieve the prescribed course outcomes effectively.

The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accordingly, the 'K' scheme laboratory manual development team designed the practical'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 manual also provides guidelines to teachers and instructors to effectively facilitate student-centred 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 versatile and powerful programming language widely used in various fields, including web development, data science, artificial intelligence, and more. Its simplicity and readability make it an excellent choice for beginners, while its robust features and libraries cater to advanced programming needs. This lab manual is structured to facilitate a hands-on learning experience. Each lab session includes detailed explanations of key concepts, step-by-step instructions for practical exercises, and real-world examples to illustrate the applications of Python. By working through these exercises, you will gain a solid understanding of Python programming and develop the ability to apply your knowledge to solve complex problems. 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.

We sincerely hope that this manual proves to be an instrumental resource in your professional journey towards becoming software developer.

Program Outcomes (POs) to be achieved through Practical:

PO1	Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
PO2	Problem analysis: Identify and analyses well-defined engineering problems using codified standard methods.
PO3	Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
PO4	Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
PO5	Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.
PO6	Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
PO7	Life-long learning: Ability to analyses individual needs and engage in updating in the context of technological changes.

Practical Course Outcome Matrix

Course Outcomes (COs)

CO1	Develop python programs using control flow statements.
CO2	Perform operations on various data structures in Python.
CO3	Develop packages to solve given problem using python.
CO4	Apply object-oriented approach to solve given problem using python.
CO5	Use relevant built-in python package to develop application.

Sr. No.	Title of the Practical	CO1	CO2	CO3	CO4	CO5
1	Install given Python IDE.	✓				
2	*1. Write python program display welcome message on screen. 2. Implement the python program to read data from user and display data on screen.	✓				
3	*Implement a python programs using following operators: 1. Arithmetic 2. Relational & logical 3. Assignment 4. Bitwise 5. Membership 6. Identity	✓				
4	*Implement a python program to demonstrate the use of following conditional statements: 1. if statement 2. if..else statement 3. if..elif..else statement 4. nested if statement	✓				
5	*Implement a python program to demonstrate the use of following looping statements: 1. while loop 2. for loop 3. nested loop	✓				
6	Implement python program to demonstrate the use of loop control statements. [continue, pass, break, else]	✓				
7	*Implement a python program to perform following operations on the List: 1. Create a List 2. Access List 3. Update List 4. Delete List		✓			

Sr. No.	Title of the Practical	CO1	CO2	CO3	CO4	CO5
8	Implement Python program to demonstrate the use of built-in functions/methods on List (Any Eight Functions/methods)		✓			
9	*Implement python program to perform following operations on the Tuple: 1. Create a Tuple 2. Access Tuple 3. Print Tuple 4. Delete Tuple 5. Convert tuple into list and vice-versa		✓			
10	*Implement a python program to perform following operations on the Set: 1. Create a Set 2. Access Set 3. Update Set 4. Delete Set		✓			
11	Implement a python program to perform following functions on Set: 1. Union 2. Intersection 3. Difference 4. Symmetric Difference		✓			
12	*Implement a python program to perform following operations on the Dictionary: 1. Create a Dictionary 2. Access Dictionary 3. Update Dictionary 4. Delete Dictionary 5. Looping through Dictionary 6. Create Dictionary from list		✓			
13	Write a user define function to implement following features: 1. Function without argument 2. Function with argument 3. Function returning value			✓		
14	*Implement user defined function for given problem: 1. Function positional/required argument 2. Function with keyword argument 3. Function with default argument 4. Function with variable length argument			✓		
15	Write Python program to demonstrate use of following advanced functions: 1. lambda 2. map 3. reduce			✓		

Sr. No.	Title of the Practical	CO1	CO2	CO3	CO4	CO5
16	Write a python program to create and use a user defined module for a given problem.			✓		
17	Write a python program to demonstrate the use of following module: 1. math module 2. random module 3. os module			✓		
18	*Write python program to create and use a user defined package for a given problem.			✓		
19	Write a python program to use of numpy package to perform operation on 2D matrix. Write a python program to use of matplotlib package to represent data in graphical form.				✓	
20	*Develop a python program to perform following operations: 1. Creating a Class with method 2. Creating Objects of class 3. Accessing method using object				✓	
21	*Write a python program to demonstrate the use of constructors: 1. Default 2. Parameterized 3. Constructor Overloading				✓	
22	*Implement a python program to demonstrate 1. Method Overloading 2. Method Overriding				✓	
23	Write python program to demonstrate data hiding.				✓	
24	*Write a python program to implement 1. Single inheritance 2. Multiple Inheritance 3. Multilevel inheritance				✓	
25	*Implement Python program to perform following operations using panda package: 1. Create Series from Array 2. Create Series from List 3. Access element of series 4. Create DataFrame using List or dictionary					✓
26	Implement python program load a CSV file into a Pandas DataFrame and perform operations.					✓
27	*Write python GUI program to import Tkinter package and create a window and set its title.					✓
28	Write python GUI program that adds labels and buttons to the Tkinter window.					✓

Sr. No.	Title of the Practical	CO1	CO2	CO3	CO4	CO5
29	Write program to create a connection between database and python.					✓
30	Implement python program to select records from the database table and display the result.					✓

List of Industry Relevant Skills

The following industry relevant skills of the competency " Develop applications using python to solve given 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 Packages.

Guidelines to Teachers

1. Teachers should align the explanation of the topic to teaching learning outcome (TLOs).
2. Refer to laboratory learning outcome (LLOs) for the execution of the practical to focus on the defined objectives.
3. Promote life-long learning by training the students to equip themselves with essential knowledge, skills and attitudes.
4. If required, provide demonstration for the practical emphasizing on the skills that the student should achieve.
5. Teachers should give opportunity to the students for exhibiting their skills after the demonstration.
6. Provide feedback and/or suggestions and share insights to improve effectiveness.
7. Assess students' skill achievement related to COs of each unit.

Instructions for Students

1. 100% attendance is compulsory for all practical sessions.
2. Students must adhere to ethical practices.
3. Plagiarism is strictly prohibited.
4. Students should accomplish the requisites of Teamwork, Collaboration and Group Dynamics during the practical sessions.
5. Conscious practice to develop professional communication on your own in and out of class is essential to achieve the course objectives.
6. All the students must follow the schedule of practical sessions, complete the assigned work/activity and submit the assignment in stipulated time as instructed by the course teacher.
7. Follow formal attire and maintain personal appearance.

Content Page

List of Practical and Formative Assessment Sheet

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (50)	Dated Sign. of Teacher	Remarks (if any)
1	Install given Python IDE.					
2	*1. Write python program display welcome message on screen. 2. Implement the python program to read data from user and display data on screen.					
3	*Implement a python programs using following operators: 1. Arithmetic 2. Relational & logical 3. Assignment 4. Bitwise 5. Membership 6. Identity					
4	*Implement a python program to demonstrate the use of following conditional statements: 1. if statement 2. if..else statement 3. if..elif..else statement 4. nested if statement					
5	*Implement a python program to demonstrate the use of following looping statements: 1. while loop 2. for loop 3. nested loop					
6	Implement python program to demonstrate the use of loop control statements. [continue, pass, break, else]					
7	*Implement a python program to perform following operations on the List: 1. Create a List 2. Access List 3. Update List 4. Delete List					

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (50)	Dated Sign. of Teacher	Remarks (if any)
8	Implement Python program to demonstrate the use of built-in functions/methods on List (Any Eight Functions/methods)					
9	*Implement python program to perform following operations on the Tuple: 1. Create a Tuple 2. Access Tuple 3. Print Tuple 4. Delete Tuple 5. Convert tuple into list and vice-versa					
10	*Implement a python program to perform following operations on the Set: 1. Create a Set 2. Access Set 3. Update Set 4. Delete Set					
11	Implement a python program to perform following functions on Set: 1. Union 2. Intersection 3. Difference 4. Symmetric Difference					
12	*Implement a python program to perform following operations on the Dictionary: 1. Create a Dictionary 2. Access Dictionary 3. Update Dictionary 4. Delete Dictionary 5. Looping through Dictionary 6. Create Dictionary from list					
13	Write a user define function to implement following features: 1. Function without argument 2. Function with argument 3. Function returning value					

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (50)	Dated Sign. of Teacher	Remarks (if any)
14	*Implement user defined function for given problem: 1. Function positional/required argument 2. Function with keyword argument 3. Function with default argument 4. Function with variable length argument					
15	Write Python program to demonstrate use of following advanced functions: 1. lambda 2. map 3. reduce					
16	Write a python program to create and use a user defined module for a given problem.					
17	Write a python program to demonstrate the use of following module: 1. math module 2. random module 3. os module					
18	*Write python program to create and use a user defined package for a given problem.					
19	Write a python program to use of numpy package to perform operation on 2D matrix. Write a python program to use of matplotlib package to represent data in graphical form.					
20	*Develop a python program to perform following operations: 1. Creating a Class with method 2. Creating Objects of class 3. Accessing method using object					
21	*Write a python program to demonstrate the use of constructors: 1. Default 2. Parameterized 3. Constructor Overloading					

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (50)	Dated Sign. of Teacher	Remarks (if any)
22	*Implement a python program to demonstrate 1. Method Overloading 2. Method Overriding					
23	Write python program to demonstrate data hiding.					
24	*Write a python program to implement 1. Single inheritance 2. Multiple Inheritance 3. Multilevel inheritance					
25	*Implement Python program to perform following operations using panda package: 1. Create Series from Array 2. Create Series from List 3. Access element of series 4. Create DataFrame using List or dictionary					
26	Implement python program load a CSV file into a Pandas DataFrame and perform operations.					
27	*Write python GUI program to import Tkinter package and create a window and set its title.					
28	Write python GUI program that adds labels and buttons to the Tkinter window.					
29	Write program to create a connection between database and python.					
30	Implement python program to select records from the database table and display the result.					
				Total		

***Total marks to be transferred to proforma published by MSBTE**

Note:

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

Practical No. 1: Install 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. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO1 - Develop python programs using control flow statements.

IV. Laboratory Learning Outcome(s)

LLO.1 Install the given Python IDE.

V. Relevant Affective Domain related Outcomes

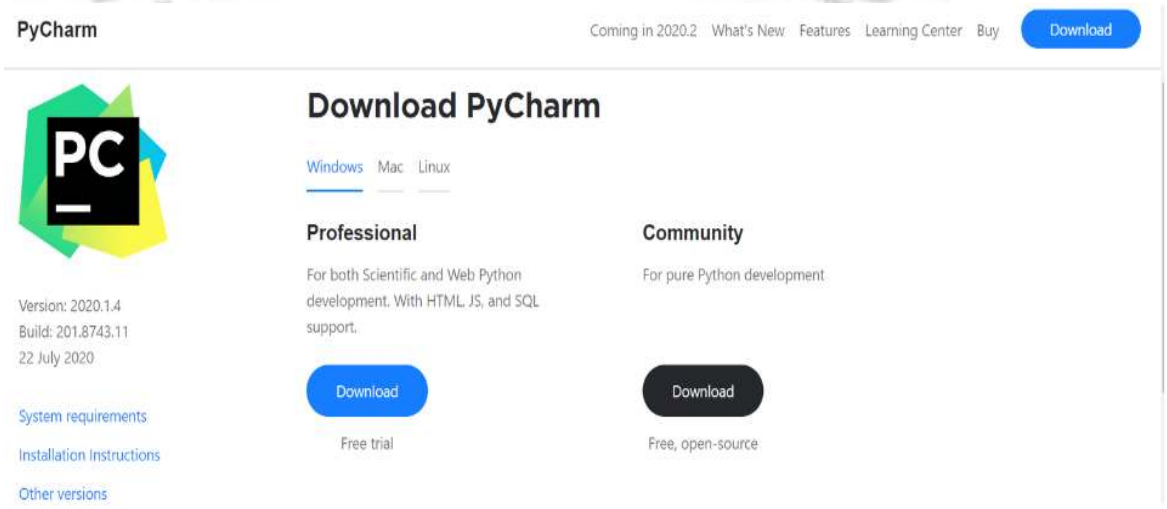
1) Follow safety practices.
2) Demonstrate working as a leader / a team member.
3) Follow ethical practices.

VI. Relevant Theoretical Background

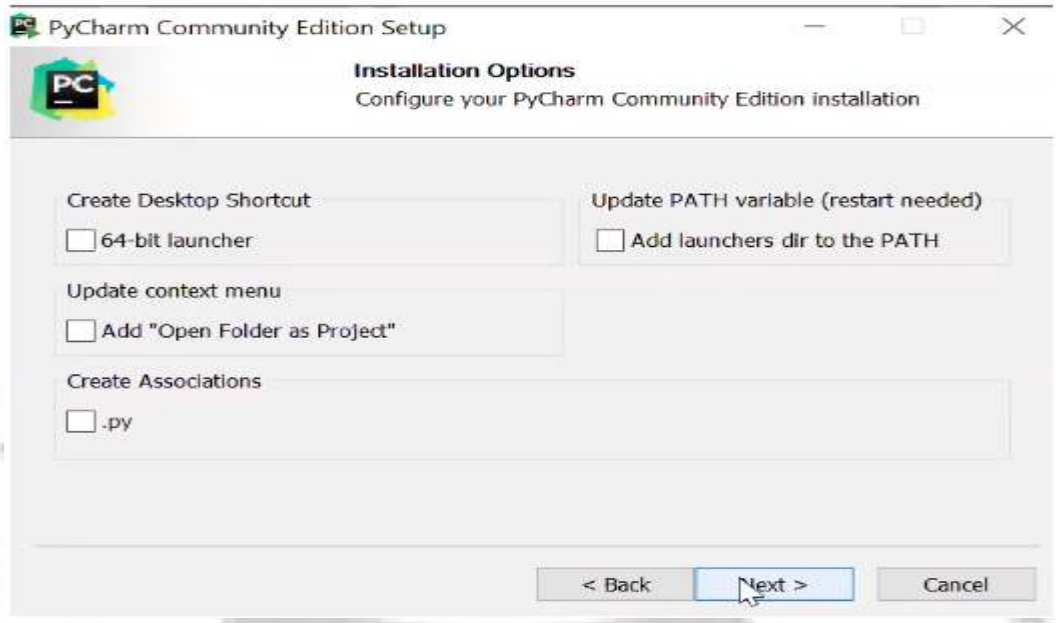
Python was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV Show called 'Monty Python's Flying Circus' and not after Python- the snake.

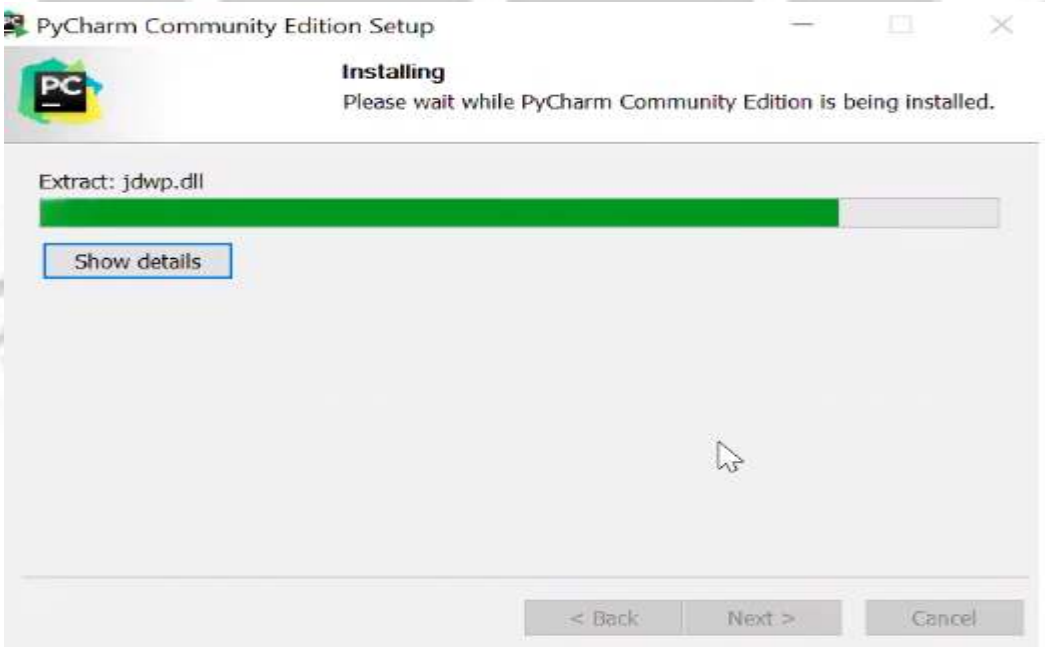
Installing PyCharm:

1. **Download PyCharm:**
   - Go to the JetBrains PyCharm website.
   - Choose the appropriate version for your operating system (Windows, macOS, or Linux).
   - Select the Community edition if you are looking for a free version.
2. **Install PyCharm:**
   - Go to https://www.jetbrains.com/pycharm/download/#section=windows . This will display the latest version of PyCharm. Underneath community, download the free, open-source version of PyCharm.

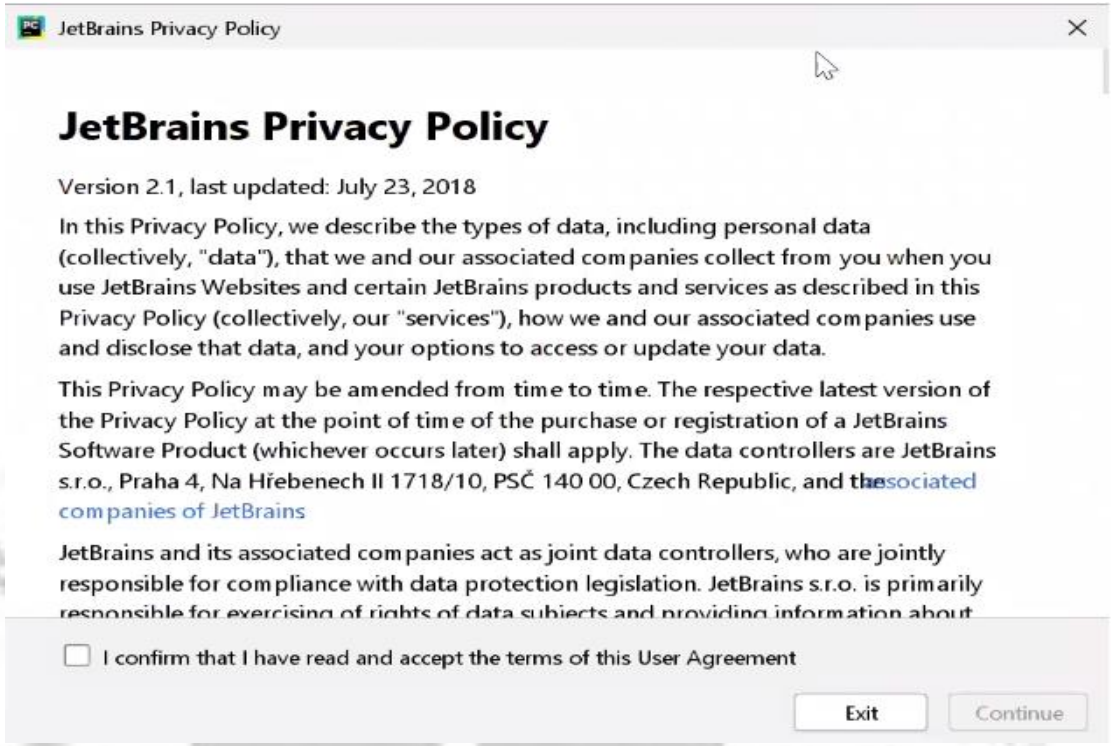


- Click next and go with PyCharm's default settings. PyCharm should then start installing itself.

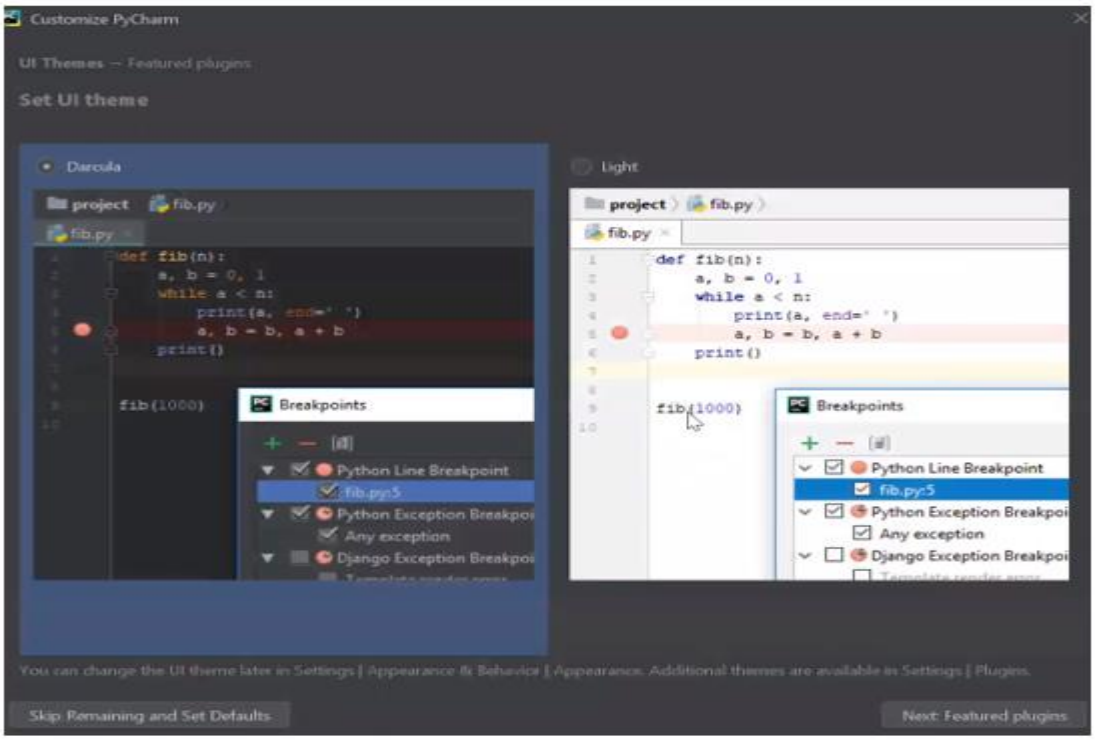




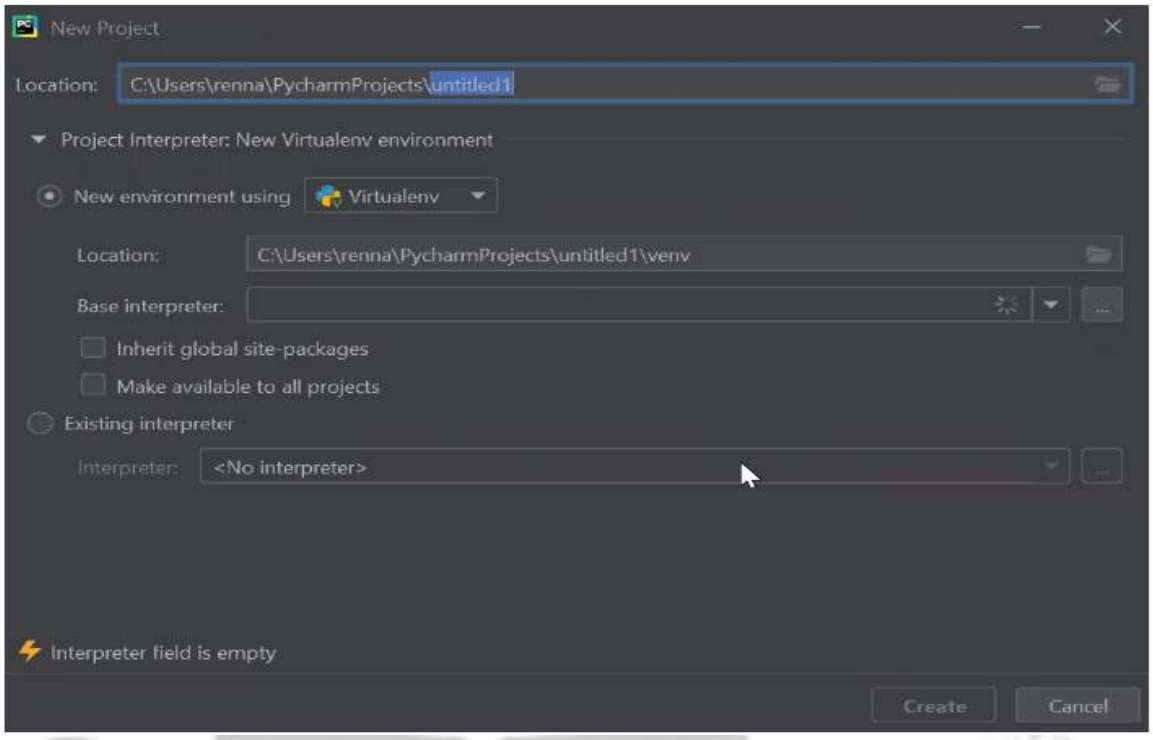
- We will now test if PyCharm works by running the program. You will be prompted to read JetBrains Privacy Policy. Once you have read it, confirm that you have the User Agreement.

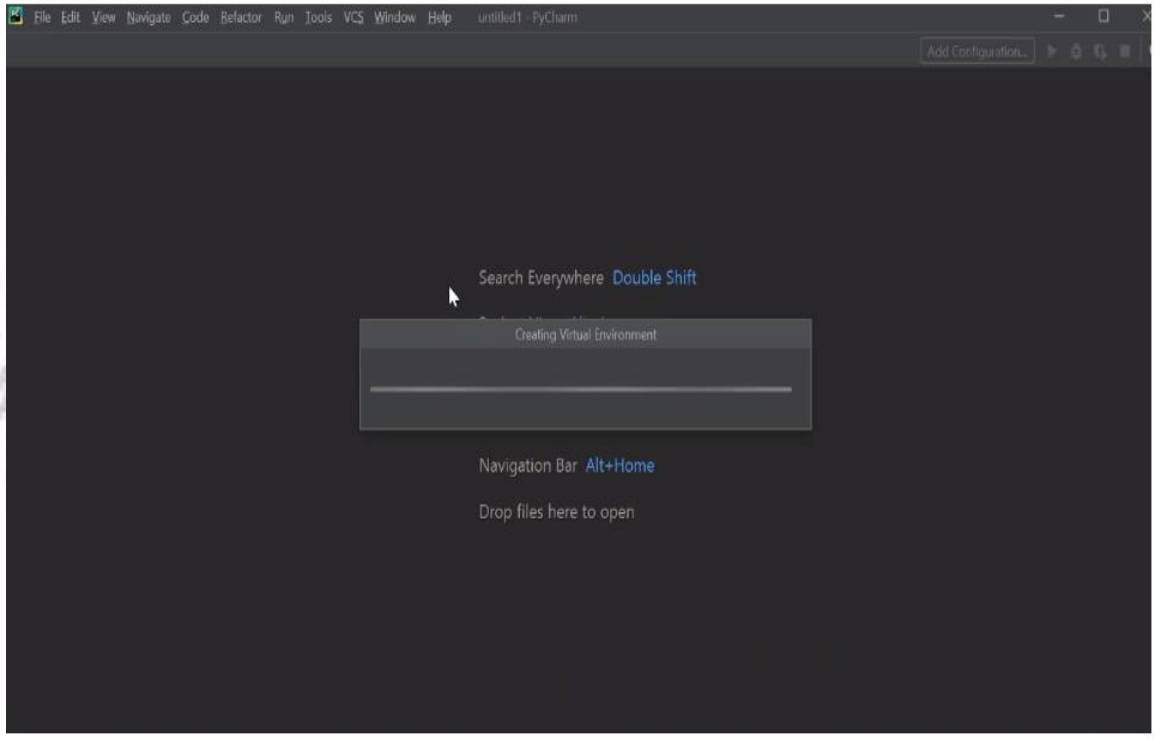


- You can now choose between different colours for your code editing software. Usually, black is easier on the eyes but this can be changed later. Click your option and then click "skip remaining and set defaults".



- A screen like this should now appear. Allow PyCharm to set up everything and then click "create". PyCharm will then set itself up internally.





https://www.youtube.com/watch?v=RH1bLpb-U_U

Configuring PyCharm

a. **Launch PyCharm:**

Open PyCharm after installation. The first time you open it, you may need to configure some initial settings.

b. **Create a New Project:**
On the welcome screen, click on "Create New Project."
Choose a location for your project.
Select the Python interpreter. If you don't have Python installed, you can install it from this link https://www.jetbrains.com/help/pycharm/installation-guide.html

c. **Configure Python Interpreter:**
PyCharm should automatically detect your Python installation. If not, you can manually specify the path to the Python executable.
To configure the interpreter later, go to File > Settings > Project: <project name> > Python Interpreter.

d. **Set Up Virtual Environment (Optional but recommended):**
During the project creation, PyCharm will prompt you to create a virtual environment. This isolates your project dependencies from the global Python installation.
Choose "New environment" using Virtualenv, specify the location, and ensure the base interpreter is the Python executable.

e. **Install Packages:**
To install additional Python packages, go to File > Settings > Project: <project name> > Python Interpreter.
Click on the "+" icon to search for and install packages like numpy, pandas, etc.

f. **Configure Editor Settings:**
Go to File > Settings and explore the various options to customize the editor's appearance, behavior, and key bindings.
Set your preferred color scheme, font, code style, and other preferences.

g. **Version Control Integration:**
If you use version control (like Git), go to File > Settings > Version Control.
Configure your VCS by specifying the path to the Git executable and setting up repositories.

h. **Run and Debug Python Code:**
To run a Python script, right-click on the file in the project explorer and select Run <filename>.
Use the built-in debugger to step through your code, inspect variables, and troubleshoot issues.
>>> print("Welcome to python")
Welcome to python

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1) Write steps to install python on windows.
2) Print the version of Python.
3) List key features of python.
4) What are the key features of Python?
5) What are the applications of Python?
6) State use of pip & pep.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/how-to-install-python-on-windows/
2. https://www.python.org/download/releases/2.5/msi/
3. https://www.maketecheasier.com/set-up-Python-windows10
4. https://www.youtube.com/watch?v=yivyNCtVVDk
5. https://www.youtube.com/watch?v=OOytKCeaNBo

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

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 program using Interactive and Script mode.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO1 - Develop python programs using control flow statements.

IV. Laboratory Learning Outcome(s)

LLO.2 Write python program for performing basic input and output operation in given problem

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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 3.8.7rc1 (tags/v3.8.7rc1:e320109, Dec 7 2020, 16:32:47) [MSC v.1928 32 bit (Intel)] on win32

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!"); 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!

>>> print("Welcome to python")
Welcome to python

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. List different modes of Programming in Python
2. State the steps involved in executing the program using Script Mode.
3. Write a Python program to display “MSBTE” using Script Mode.

……..

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-program-to-print-hello-world/
2. https://www.ics.uci.edu/~pattis/common/handouts/Pythoneclipsejava/Python.html
3. https://www.toppr.com/guides/python-guide/examples/python-examples/write-python-hello-world-program/
4. https://www.youtube.com/watch?v=bv4X8T5r98E
5. https://www.youtube.com/watch?v=9jY-Jz1pqrA

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely submission of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

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 mode.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO1 - Develop python programs using control flow statements.

IV. Laboratory Learning Outcome(s)

LLO.3 Write python program to solve given expression.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Operators: Are used to perform operations on values and variables. These are the special symbols that carry out arithmetic and logical computations.
Operands: The value the operator operates on is known as the Operand.

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

Operator	Meaning	Description	Syntax	Example	Output
+	Addition	Adds the value of left & right operands	>>>x+y	>>> 5+2	7

-	Subtraction	Subtracts the value of the right operand from the value of the left operand	>>>x-y	>>> 5-2	3
*	Multiplication	Multiplies the value of the left and right operand	>>>x*y	>>> 5*2	10
/	Division	Divides the value of the left operand by the right operand	>>>x/y	>>> 5/2	2.5
**	Exponent	Performs exponential calculation	>>>x**y	>>> 5**2	25
%	Modulus	Returns the remainder after dividing the left operand with the right operand.	>>>x%y	>>> 5%2	1
//	Floor Division	Division of operands where the solution is a quotient left after removing decimal numbers	>>>x//y	>>> 5//2	2

Logical Operators: Python logical operators are used to combine conditional statements, allowing you to perform operations based on multiple conditions. These Python operators, alongside arithmetic operators, are special symbols used to carry out computations on values and variables.

Operator	Meaning	Description	Example	Output
and	Logical AND	Returns True if both the operands are 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: Python bitwise operators are used to perform bitwise calculations on integers. The integers are first converted into binary and then operations are performed on each bit or corresponding pair of bits, hence the name bitwise operators. The result is then returned in decimal format.

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 shift	The left operand's value is moved left by the number of bits specified by the right operand.	>>> (a<<2)	40

Membership Operators: Membership operator in Python test a sequence, such as a string, a list, or a tuple, for membership. They check whether a sequence appears in an object or not.
There are two membership operators in Python. To check if any sequence is present in any object, we use in, and to check if a sequence is not present in any object, we use not in.
We get the output in the form of a Boolean value that is True or False.

Operator	Meaning	Description	Example	Output

in	to check if a sequence is present in any object	Returns True if a sequence with the specified value is present in the object	>>> item=10
			>>> list_1=[1,2,3,4,5]
			>>> item in list_1
				False

not in	To check if any sequence is not present in any object	Returns True if a sequence with the specified value is not present in the object	>>> item=10
			>>> list_1=[1,2,3,4,5]
			>>> item not in list_1
				True

Identity Operators: The Python Identity Operators are used to compare the objects if both the objects are actually of the same data type and share the same memory location.

Operator	Description	Example	Output

is	Returns True if both objects refers to same memory location, else returns False	>>> x=2
		>>> y=3
		>>> print(x is z)
			True

is not	Returns False if both object refers to same memory location, else returns True	>>> x=2
		>>> y=3
		>>> print(x is not z)
			True

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty.
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Describe ternary operator in Python.
2. Describe about different Bitwise operators in Python with appropriate examples.
3. Describe about different Logical operators in Python with appropriate examples.
4. Write a program to find the square root of a number.
5. Write a program to convert bits to Megabytes, Gigabytes and Terabytes.
6. Write a program to swap the value of two variables.
7. Write a program to calculate surface volume and area of a cylinder.

……..

……..

……..

……..

……..

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.programiz.com/python-programming/operators
2. https://www.geeksforgeeks.org/python-operators/
3. https://www.geeksforgeeks.org/python-bitwise-operators/
4. https://www.youtube.com/watch?v=LK4kIPuEgUU
5. https://www.youtube.com/watch?v=v5MR5JnKcZI

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

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

I. Practical Significance

In both real life and programming, decision-making is crucial. We often face situations where we need to make choices, and based on those choices, we determine our next actions. Similarly, in programming, we encounter scenarios where we must make decisions to control the flow of our code.

Conditional statements in Python play a key role in determining the direction of program execution. Among these, If-Else statements are fundamental, providing a way to execute different blocks of code based on specific conditions. As the name suggests, If-Else statements offer two paths, allowing for different outcomes depending on the condition evaluated.

Types of Control Flow in Python

- Python if Statement
- Python if….else Statement
- Python nested if Statement
- Python elif
- Ternary Statement | Short Hand If Else Statement

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO1 - Develop python programs using control flow statements.

IV. Laboratory Learning Outcome(s)

LLO.4 Write python program for solving given problem using various If statements.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Differentiate between if-else and nested-if statement about different Logical operators in Python with appropriate examples.
2. Write a program to check the largest number among the three numbers
3. Write a program to check if the input year is a leap year of not 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 grades.
4. List operators used in if conditional statement.

……..

……..

……..

……..

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python3-if-if-else-nested-if-if-elif-statements/
2. https://www.geeksforgeeks.org/nested-if-statement-in-python/
3. https://www.programiz.com/python-programming/if-elif-else
4. https://www.scholarhat.com/tutorial/python/decision-making-statements-if-else-nested-if-else
5. https://www.youtube.com/watch?v=FvMPfrgGeKs&t=5s

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

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.

1. while loop
2. for loop
3. nested loop

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO1 - Develop python programs using control flow statements.

IV. Laboratory Learning Outcome(s)

LLO.5.1 Write python program for solving given problems using a while loop.

LLO.5.2 Write python program for solving given problem using for loop.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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.

d. **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!
```

e. **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:

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!
```

f. **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!
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Write a Python program that takes a number and checks whether it is a palindrome or not.
2. Write a Python program to print all even numbers between 1 to 100 using while loop.
3. Print the following patterns using loop:

   a.
   ```
   1010101
    10101
     101
      1
   ```

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

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

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

5. Write a Python Program to Reverse a given number.
6. Write a python program to find Factorial of given number.
7. Write a python program to find Fibonacci series for given number.
8. Find sum of four digit number.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/loops-in-python/
2. https://www.scholarhat.com/tutorial/python/while-for-nested-loop
3. https://www.softwaretestinghelp.com/python/looping-in-python-for-while-nested-loops/
4. https://www.youtube.com/watch?v=W4Dw-G2oWhY
5. https://www.youtube.com/watch?v=M0RsvJnaGYg

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 6: Write Python program to demonstrate use of loop control statements: continue, pass, break

I. Practical Significance

Loop control statements are used to change the flow of execution. These can be used if you wish to skip an iteration or stop the execution. The three types of loop control statements in python are break statement, continue statement, and pass statement.

1. continue
2. pass
3. break

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO1 - Develop python programs using control flow statements.

IV. Laboratory Learning Outcome(s)

LLO.6.1 Use loop control statements in python for solving given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Python provides the following control statements.

1. continue
2. pass
3. break

1. **continue:** This command skips the current iteration of the loop. The statements following the continue statement are not executed once the Python interpreter reaches the continue statement.

Syntax:

```
while True:
        statement(s)
        if x == 10:
                continue
        print(x)
```

Example:

```
for var in "MSBTE":
        if var == "B":
                continue
```

```
        print(var)
```

Output:

```
M
S
T
E
```

2. **pass:** The pass statement is used when a statement is syntactically necessary, but no code is to be executed.

   In Python programming, the pass statement is a null statement which can be used as a placeholder for future code.

   Suppose we have a loop or a function that is not implemented yet, but we want to implement it in the future. In such cases, we can use the pass statement.

   Syntax:

   ```
   pass
   ```

   Example:

   ```
   n = 10
   # use pass inside if statement
   if n > 10:
      pass
   print('Hello')
   ```

   Output:

   ```
   Hello
   ```

3. **break:** This command terminates the loop's execution and transfers the program's control to the statement next to the loop.

   'Break' in Python is a loop control statement. It is used to control the sequence of the loop. Suppose you want to terminate a loop and skip to the next code after the loop; break will help you do that. A typical scenario of using the Break in Python is when an external condition triggers the loop's termination.

   Syntax:

   ```
   break;
   ```

   Example:

   ```
   if letter == 'e' or letter == 's':
         break
   print('Current Letter :', letter)
   ```

   Output:

   ```
   Current Letter : e
   ```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Describe Keyword "continue" with example.
2. Describe pass with example.
3. Write program to demonstrate use of break.
4. Write program to demonstrate use of nested if-else.

……..

……..

……..

……..

……..

……..

……..

……..

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.youtube.com/watch?v=yCZBnjF4_tU
2. https://www.youtube.com/watch?v=DpE5TJNNb9Q
3. https://www.youtube.com/watch?v=97NdNoA3XUQ
4. https://www.geeksforgeeks.org/break-continue-and-pass-in-python/
5. https://www.geeksforgeeks.org/loops-and-control-statements-continue-break-and-pass-in-python/

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 7: Write Python program to perform following operations on Lists: Create list, Access list, Update list (Add item, Remove item), and 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.

1. Create list
2. Access list
3. Update list (Add item, Remove item)
4. Delete list

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO2 - Perform operations on various data structures in Python.

IV. Laboratory Learning Outcome(s)

LLO.7.1 Write python program to perform operations on list.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

List: In Python, a list is a type of data structure that holds an ordered collection of items, which means you can store anything from integers to strings, to other lists, and so on. They are mutable, allowing you to change their content without changing their identity.

- A list is a collection of different kinds of values or items.
- Python lists are mutable, we can change their elements after forming.
- The comma (,) and the square brackets [enclose the List's items] serve as separators.
- The index always starts with 0 and ends with n-1, if the list contains n elements.

b. **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]
```

c. **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
```

d. **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]
```

a) 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:

```
>>>List2 [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]
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1) Write syntax for a method to sort a list.
2) Justify the statement “Lists are mutable” not.
3) Describe various list functions.
4) Describe the use pop operator in list.
5) Describe various list functions.
6) Write a Python program to find common items from two lists.
7) Write a Python program to reverse a list.

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.scholarhat.com/tutorial/python/list-in-python
2. https://www.geeksforgeeks.org/python-lists/
3. https://www.programiz.com/python-programming/list
4. https://www.includehelp.com/python/program-for-various-list-operations.aspx
5. https://www.youtube.com/watch?v=9OeznAkyQz4

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 8: Write python program to use built-in functions/methods on list: cmp, len, max, list, append, count, extend, insert, pop, remove, etc.

I. Practical Significance

The list function in Python allows you to generate a mutable collection. List functions are global functions that can be used with any list. They take a list as an argument and return a value. Just like all built-in functions in Python, the list functions are first-class objects and are the functions that create or act on list objects and other sequences.

Most list functions act on list objects in-place. This is due to a list's characteristic called mutability, which enables us to modify the lists directly.

List methods, on the other hand, are built-in functions that are specific to the list data type.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO2 - Perform operations on various data structures in Python.

IV. Laboratory Learning Outcome(s)

LLO.8.1 Write python program to use built-in functions on list.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Built-in Functions: List functions are global functions that can be used with any list. These functions take a list as an argument and return a value. Some examples of list functions are len(), sorted(), and min().

Function	Description	Syntax	Output
cmp( )	The Python List cmp() method compares elements of two lists.	cmp(list1, list2)	The method returns 1 if the first list is greater than the second list and -1 if it's smaller; 0 is returned when both the lists are equal.
len( )	Gives the total length of the list.	len(list)	The Python List len() method is used to compute the size of a Python List.
max( )	Returns item from the list with max value.	max(list)	The Python List max() method compares the elements of the list and returns the maximum valued element.
min( )	Returns item from the list with min value.	min(list)	The Python list min () method compares the elements of the list and

			returns the element with minimum value.
list( )	Converts a tuple into list.	list(seq)	The Python List list() method is used to convert an object into a list. This method accepts sequence type objects as arguments and converts them to lists.

Built-in Methods: Methods are built-in functions that are specific to the list data type. These methods can only be used with lists and modify the list in place. Some examples of list methods are append(), insert(), and remove().

Method	Description	Syntax	Example
append( )	Appends object obj to list	list.append(obj)	>>>list = ['Mon', 'Tue', 'Wed'] >>>print("Existing list\n",list) # Append an element >>>list.append('Thu') >>>print("Appended one element: ",list)
count( )	Returns count of how many times obj occurs in list	list.count(obj)	>>>aList = [123, 'xyz', 'annu', 'mac', 123] >>>print("Count for 123 : ", aList.count("annu"))
extend( )	The extend() method appends the elements in the iterable individually to the original list.	list.extend(seq)	>>>aList = [123, 'xyz', 'annu', 'mac', 123] >>>aList.extend(bList) >>>print("Extended List : ", aList)
index( )	Returns the lowest index in list that obj appears	list.index(obj[, start[, end]])	>>>aList = [123, 'xyz', 'annu', 'mac']; >>>print("Index for annu : ", aList.index('annu'))
insert( )	insert() method inserts an object into a list at a specified position.	list.insert(index, obj)	>>>aList = [123, 'xyz', 'zara', 'abc'] >>>aList.insert(3, 2009)
pop()	Removes and returns last object or obj from list	list.pop(obj = list[-1])	>>>aList = [123, 'xyz', 'zara', 'abc'] >>>print("Popped Element : ", aList.pop()) >>>print("Updated List:") >>>print(aList)
remove( )	Removes object obj from list	list.remove(obj)	>>>aList = [123, 'xyz', 'zara', 'abc', 'xyz'] >>>aList.remove('xyz') >>>print("List : ", aList)
reverse( )	Reverses objects of list in place	list.reverse()	>>>aList = [123, 'xyz', 'zara', 'abc', 'xyz'] >>>aList.reverse() >>>print("List : ", aList)
sort ( )	Sorts objects of list, use compare func if given	list.sort(*, key=None, reverse=False)	>>>aList = ['123', 'xyz', 'zara', 'abc', 'xyz'] >>>aList.sort() >>>print("List : ", aList)

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1) Write syntax for a method to reverse a list.
2) Describe various list functions.
3) Describe the use of pop operator in list.
4) Describe the use extend & pop method in list.
5) Write a Python program to find common items from two lists.
6) Write a Python program to sort a list.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.tutorialspoint.com/built-in-list-functions-and-methods-in-python
2. https://www.datacamp.com/tutorial/python-list-function
3. https://www.w3schools.com/python/python_ref_functions.asp
4. https://www.youtube.com/watch?v=7rjJrQy9gi4
5. https://www.youtube.com/watch?v=cSKhKyANclU

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 9: Write python program to perform following operations on tuple: Create, Access, Print, Delete & Convert tuple into list and vice-versa

I. Practical Significance

The Like list python supports new tuple as a distinct structure. Tuple are used to store sequence of python objects which are static in nature. A tuple is immutable which 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. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO2 - Perform operations on various data structures in Python.

IV. Laboratory Learning Outcome(s)

LLO.9.1 Write python program to perform operations on tuple.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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 = ('annu', 'mac', 1997, 2000);
print tup;
del tup;
```

e) Convert Tuple to List: These two Python datatypes can seem to be similar but have different usage in context. The key difference between tuples and lists lies in their mutability. Conversion of the tuple to a list only occurs when you need to modify the element.

To convert Tuples to a List you need to first make some changes and then convert a tuple to a list as it is not possible for you to change the tuple directly into the list as Tuples are immutable. Now we will delve into the different methods to convert tuples into lists.

- Using the list() Function
- Using a for loop
- Using List Comprehension
- Using(*)operator
- Using the map() Function

1) **Using the list () Function:** The simplest and easiest way to convert a tuple to a list is by using the built-in list () function.

Example:

```
# Define a tuple
```

```
GFG_tuple = (1, 2, 3)
 # Convert the tuple to a list
GFG_list = list(GFG_tuple)
print(GFG_list)
```

Output:

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

2) **Using for loop:** Using for loop that iterates through each element in our tuple. For each iteration of the loop (i.e., for every item in the tuple), append () method adds the elements to the end of the list.

Example:

```
GFG_tuple = ( 1, 2, 3)
GFG_list = []
 for i in GFG_tuple:
    GFG_list.append(i)
 print(GFG_list)
```

Output:

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

3) **Using List Comprehension:** Using List comprehension is another way to perform this conversion. It helps a sequence to be built from another sequence in a clear and concise manner.

Example:

```
# Define a tuple
GFG_tuple = (1, 2, 3)
 # Convert the tuple to a list using list comprehension
GFG_list = [element for element in GFG_tuple]
print(GFG_list)
```

Output:

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

4) **Using (*) operator:** The * operator also known as unpacking in Python has a number of different uses. One of the use is to unpack a collection into positional arguments within a function call. We use this to convert a tuple into a list.

Example:

```
# Define a tuple
GFG_tuple = (1, 2, 3)
 # Convert the tuple to a list using *operator
GFG_list = [*GFG_tuple]
print(GFG_list)
```

Output:

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

5) using the map() Function: The map() function applies a given function in each item and returns a list of results.

Example:

```
# Define a tuple
GFG_tuple = (1, 2, 3)
 # Convert the tuple to a list using map function
GFG_list = list(map(lambda x: x, GFG_tuple))
 print(GFG_list)
```

Output:

[1, 2, 3]

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

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).
4) Create a tuple and find the minimum and maximum number from it.
5) Write a Python program to find the repeated items of a tuple.
6) Print the number in words for Example: 1234 => One Two Three Four

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-tuples/
2. https://www.programiz.com/python-programming/tuple
3. https://www.studytonight.com/python/tuples-in-python
4. https://www.youtube.com/watch?v=7syexRDpyrQ
5. https://www.youtube.com/watch?v=cDmFNDfOvzY

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 10: Write python program to perform following operations on the Set: Create set, Access Set, 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. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO2 - Perform operations on various data structures in Python.

IV. Laboratory Learning Outcome(s)

LLO.10.1 Write python program to manipulate the set.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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}
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1) Describe the various set operations.
2) Describe the various methods of set.
3) Write a Python program to create a set, add member(s) in a set and remove one item from set.
4) Write a Python program to find maximum and the minimum value in a set.
5) Write a Python program to find the length of a set.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.programiz.com/python-programming/examples/set-operation
2. https://www.geeksforgeeks.org/python-set-operations-union-intersection-difference-symmetric-difference/
3. https://www.programiz.com/python-programming/set
4. https://www.youtube.com/watch?v=Czr-_Sx9otk
5. https://www.youtube.com/watch?v=JP32Qhy0Z9U

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 11: Write python program to perform following functions on Set: Union, Intersection, Difference, Symmetric Difference

I. Practical Significance

A Set in Python is a collection of unique elements which are unordered and mutable. Python provides various functions to work with Set. In this article, we will see a list of all the functions provided by Python to deal with Sets. 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 functions supported by set data structure in python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO2 - Perform operations on various data structures in Python.

IV. Laboratory Learning Outcome(s)

LLO.11.1 Use built-in functions/methods on sets in python for solving given problems.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

A set is a collection of unordered values or items. The set in python has similar properties to the set in mathematics. Let us learn some of the most important features of set in python:

- A set in python is an iterable object.
- A set in python does not contain duplicate values.
- A set in Python is an unordered collection of items.
- A set in python is an elastic object (mutable) object but the elements of the set in non-mutable.
- We cannot use indexing on the sets as no index is attached to any element in the python set.
- Slicing operation cannot be performed on the set in python because sets do not have indexing.

Since the set in python is inspired by mathematics sets, we can perform various mathematic-based operations on sets like intersection, union, set difference, complement, etc. Let us learn about these operations and other operations of the set in python.

a) Union: Return a set that contains all items from both sets, duplicates are excluded.

Example:

```
>>>x = {"apple", "banana", "cherry"}
```

```
>>>y = {"google", "microsoft", "apple"}
>>>z = x.union(y)
>>>print(z)
```

Output:

```
{'banana', 'microsoft', 'apple', 'cherry', 'google'}
```

b) Intersection: Returns a set, that is the intersection of two other sets.

Example:

```
>>>x = {"apple", "banana", "cherry"}
>>>y = {"google", "microsoft", "apple"}
>>>z = x.intersection(y)
>>>print(z)
```

Output:

```
{'apple'}
```

c) Difference: Returns a set containing the difference between two or more sets.

Example:

```
>>>x = {"apple", "banana", "cherry"}
>>>y = {"google", "microsoft", "apple"}
>>> z = x.difference(y)
>>>print(z)
```

Output:

```
{'cherry', 'banana'}
```

d) Symmetric Difference: Returns a set with the symmetric differences of two sets.

Example:

```
>>>x = {"apple", "banana", "cherry"}
>>>y = {"google", "microsoft", "apple"}
>>> z = x.symmetric_difference(y)
>>>print(z)
```

Output:

```
{'banana', 'google', 'cherry', 'microsoft'}
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1) Explain mutable and immutable data structures.
2) Explain any six set function with example.
3) Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.
4) Difference between set & tuple.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.programiz.com/python-programming/examples/set-operation
2. https://www.geeksforgeeks.org/python-set-operations-union-intersection-difference-symmetric-difference/
3. https://www.programiz.com/python-programming/set
4. https://www.youtube.com/watch?v=Czr-_Sx9otk
5. https://www.youtube.com/watch?v=JP32Qhy0Z9U

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 12: Write python program to perform following operations on the Dictionary: Create, Access, Update, Delete, Looping through Dictionary, Create Dictionary from list

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. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO2 - Perform operations on various data structures in Python.

IV. Laboratory Learning Outcome(s)

LLO.12.1 Write python program to perform operations on dictionary.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

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

a) Creating the Dictionary: The 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 {}.

Example:

```
#!/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
```

e) 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.

Output:

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

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer 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.
   Sample Dictionary:
   dic1 = {1:10, 2:20}
   dic2 = {3:30, 4:40}
   dic3 = {5:50,6:60}
3) Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.
4) Write a Python program to combine two dictionary adding values for common keys.
   d1 = {'a': 100, 'b': 200, 'c':300}
   d2 = {'a': 300, 'b': 200, 'd':400}
5) Write a Python program to find the highest 3 values in a dictionary.
6) 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)
```

……

……

……

……

……

……

……

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

XI. References/Suggestions for further reading

1. https://www.softwaretestinghelp.com/python-dictionary-methods/
2. https://www.geeksforgeeks.org/python-dictionary/
3. https://www.programiz.com/python-programming/dictionary
4. https://www.youtube.com/watch?v=39MTu_x-7VI
5. https://www.youtube.com/watch?v=u0yr9B3nH8c

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 13: Write a user define function to implement following features: Function without argument, Function with argument, Function returning value

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. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO3 - Develop packages to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.13.1 Write function to solve given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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.

Example:

```
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
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

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) Write a Python function that takes a number as a parameter and check the number is prime or not.

3) Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.

4) Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters.

5) What is the output of the following program?

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

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-functions/
2. https://stackoverflow.com/questions/12738031/creating-a-new-function-as-return-in-python-function
3. https://realpython.com/python-return-statement/
4. https://www.youtube.com/watch?v=ArkXkpIUwjI
5. https://www.youtube.com/watch?v=ExQT_SL3eMw

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Debugging ability	

Practical No. 14: Write a user define function to implement for following problem: Function positional/required argument, Function with keyword argument, Function with default argument, Function with variable length argument

I. Practical Significance

All A function is a set of statements that take inputs, do some specific computation, and produce output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call the function. Functions that readily come with Python are called built-in functions. Python provides built-in functions like print (), etc. but we can also create your own functions. These functions are known as user defines functions. This practical will make learner use of modularized programming using functions.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO3 - Develop packages to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.14.1 Write python program to create function by selecting appropriate type of argument.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Python functions can contain two types of arguments: positional arguments and keyword arguments. Positional arguments must be included in the correct order. Keyword arguments are included with a keyword and equals sign.

A) Function Positional Arguments: An argument is a variable, value or object passed to a function or method as input. Positional arguments are arguments that need to be included in the proper position or order. The first positional argument always needs to be listed first when the function is called. The second positional argument needs to be listed second and the third positional argument listed third, etc.

Example:

An example of positional arguments can be seen in Python's complex() function. This function returns a complex number with a real term and an imaginary term. The order that numbers are passed to the complex() function determines which number is the real term and which number is the imaginary term.

If the complex number 3 + 5j is created, the two positional arguments are the numbers 3 and 5. As positional arguments, 3 must be listed first, and 5 must be listed second.

```
>>>complex(3, 5)
```

Output:

```
(5+3j)
```

Positional Arguments Specified by an Iterable: Positional arguments can also be passed to functions using an iterable object. Examples of iterable objects in Python include lists and tuples. The general syntax to use is:

Syntax:

```
function(*iterable)
```

Where function is the name of the function and iterable is the name of the iterable preceded by the asterisk * character.

Example:

```
>>>term_list = [3, 5]
>>>complex(*term_list)
```

Output:

```
(3+5j)
```

B) Function with Keyword Arguments: A keyword argument is an argument passed to a function or method which is preceded by a keyword and an equals sign.

Syntax:

```
function(keyword=value)
```

Where function is the function name, keyword is the keyword argument and value is the value or object passed as that keyword.

Example:

```
>>>complex(real=3, imag=5)
```

Output:

```
(3+5j)
```

Note: Keyword arguments are passed to functions after any required positional arguments. But the order of one keyword argument compared to another keyword argument does not matter. Note how both sections of code below produce the same output.

Example:

```
>>>complex(real=3, imag=5)
>>> complex(imag=5, real=3)
```

Output:

```
(3+5j)
(3+5j)
```

C) Function with Default arguments: A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument.

Example:

```
>>># Python program to demonstrate
```

```
>>># default arguments
>>>def myFun(x, y = 50):
        >>>print("x: ", x)
        >>>print("y: ", y)

>>># Driver code
>>>myFun(10)
```

Output:

```
x:  10
y:  50
```

D) Function with Variable Length Arguments: We can have both normal and keyword variable numbers of arguments.

- The special syntax *args in function definitions in Python is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list.
- The special syntax **kwargs in function definitions in Python is used to pass a keyworded, variable-length argument list. We use the name kwargs with the double star. The reason is that the double star allows us to pass through keyword arguments (and any number of them).

Example:

```
>>>def myFun1(*argv):
        >>>for arg in argv:
            >>>print (arg)

>>>def myFun2(**kwargs):
        >>>for key, value in kwargs.items():
            >>>print ("% s == % s" %(key, value))
>>># Driver code
>>>print("Result of * args: ")
>>>myFun1('Hello', 'Welcome', 'to', 'GeeksforGeeks')
>>>print("\nResult of **kwargs")
>>>myFun2(first ='Geeks', mid ='for', last ='Geeks')
```

Output:

```
Result of *args:
Hello
Welcome
to
GeeksforGeeks
Result of **kwargs
mid == for
first == Geeks
last == Geeks
```

E) Pass by Reference or pass by value in Python: One important thing to note is, in Python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created. Parameter passing in Python is the same as reference passing in Java. To confirm this Python's built-in id() function is used in the below example.

Example:

```
>>>def myFun(x):
        >>>print("Value received:", x, "id:", id(x))

>>># Driver's code
>>>x = 12
>>>print("Value passed:", x, "id:", id(x))
>>>myFun(x)
```

Output:

```
Value passed: 12 id: 11094656
Value received: 12 id: 11094656
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1) What are positional arguments?
2) Write program to demonstrate use of variable length arguments.
3) What is a Function Argument in Python?
4) Explain function with keyword argument.

5) What will be the output of following code?

```
def print_animals(*animals):
    for animal in animals:
      print(animal)

print_animals("Lion", "Elephant", "Wolf", "Gorilla")
```

6) What will be the output of following code

```
def print_food(**foods):
   for food in foods.items():
      print(food)
print_food(Lion="Carnivore", Elephant="Herbivore", Gorilla="Omnivore")
```

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/keyword-and-positional-argument-in-python/
2. https://stackoverflow.com/questions/5940180/python-default-keyword-arguments-after-variable-length-positional-arguments
3. https://www.geeksforgeeks.org/variable-length-argument-in-python/
4. https://www.toppr.com/guides/python-guide/references/methods-and-functions/function-argument/python-function-arguments-default-keyword-arbitrary/
5. https://www.youtube.com/watch?v=OCbSteqZsKQ

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 15: Write Python program to demonstrate use of following advanced functions: lambda, map, reduce

I. Practical Significance

Python is a dynamic yet straightforward typed language, and it provides multiple libraries and in-built functions. There are different methods to perform the same task and in Python lambda function proves to be king of all, which can be used everywhere in different ways.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO3 - Develop packages to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.15.1 Write python program to create function by selecting appropriate type of argument.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

A) The Lambda() Function: Python Lambda Functions are anonymous functions means that the function is without a name. As we already know the **def** keyword is used to define a normal function in Python.

Similarly, the **lambda** keyword is used to define an anonymous function in Python.

Syntax:

```
lambda arguments : expression
```

- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.

Example:

```
>>>str1 = 'GeeksforGeeks'
>>>upper = lambda string: string.upper()
>>>print(upper(str1))
```

Output:

```
GEEKSFORGEEKS
```

Example:

```
>>>sum = lambda x, y : x + y
```

```
>>>sum(3,4)
```

Output:

```
7
```

B) The map() Function: Python's map() method applies a specified function to each item of an iterable (such as a list, tuple, or string) and then returns a new iterable containing the results.
The first argument passed to the map function is itself a function, and the second argument passed is an iterable (sequence of elements) such as a list, tuple, set, string, etc.

Syntax:

```
map(function, iterable)
```

Example:

```
        # Using map() to square each element of the data list
>>>data = [1, 2, 3, 4, 5]
        >>># Map function returns the map object
>>>squares = map(lambda x: x*x, data)
        # Iterating the elements of the squares
>>>for i in squares:
        >>>print(i, end=" ")
        # Also, we can convert the map object into a list
>>>squares = list(map(lambda x: x*x, data))
>>>print(f"Squares: {squares}")
```

Output:

```
1, 4, 9, 16, 25
Squares: [1, 4, 9, 16, 25]
```

C) The reduce() Function: In Python, reduce() is a built-in function that applies a given function to the elements of an iterable, reducing them to a single value.

Syntax:

```
reduce(function, iterable[, initializer])
```

Example:

```
        # Examples to understand the reduce() function
>>>from functools import reduce
        # Function that returns the sum of two numbers
>>>def add(a, b):
        >>>return a + b
        # Our Iterable
>>>num_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
        # add function is passed as the first argument, and num_list is passed as the
        #second argument
>>>sum = reduce(add, num_list)
>>>print(f"Sum of the integers of num_list : {sum}")
        # Passing 10 as an initial value
>>>sum = reduce(add, num_list, 10)
>>>print(f"Sum of the integers of num_list with initial value 10 : {sum}")
```

Output:

```
Sum of the integers of num_list : 55
Sum of the integers of num_list with initial value 10 : 65
```

D) The filter() Function: The filter() function in Python filters elements from an iterable based on a given condition or function and returns a new iterable with the filtered elements.

Syntax:

filter(function, iterable)

Example:

```
        # Using filter() to filter even numbers from a list
>>>data = [1, 2, 3, 4, 5]
        # The filter function filters the even numbers from the data
        # and returns a filter object (an iterable)
>>>evens = filter(lambda x: x % 2 == 0, data)
        # Iterating the values of evens
>>>for i in evens:
        >>>print(i, end=" ")
        # We can convert the filter object into a list as follows:
>>>evens = list(filter(lambda x: x % 2 == 0, data))
        # Printing the evens list
>>>print(f"Evens = {evens}")
```

Output:

```
2 4
Evens = [2, 4]
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM > 2GB), with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

a. What does the filter() function in Python do?

b. What is the purpose of the map() function in Python?

c. What will be the output of following code?

```
from functools import reduce
def multiply(x, y):
return x * y
numbers = [1, 2, 3, 4, 5]
result = reduce(multiply, numbers)
print(result)
```

d. What will be the output of following code

```
numbers = [1, 2, 3, 4, 5]
squared = map(lambda x: x**2,numbers)
```

e. What will be the output of following code?

```
def even_check(num):
    if num % 2 == 0:
        return True
    else:
        return False
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = filter(even_check, numbers)
print(list(even_numbers))
```

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/
2. https://www.analyticsvidhya.com/blog/2022/02/lambda-functions-in-python-map-filter-and-reduce/
3. https://python-course.eu/advanced-python/lambda-filter-reduce-map.php
4. https://www.youtube.com/watch?v=YIOYJgLQYiY
5. https://www.youtube.com/watch?v=1GAC6KQUPeg

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 16: Write a python program to create and use a user defined module for a given problem

I. Practical Significance

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. User-defined python modules are the modules, which are created by the user to simplify their project.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO3 - Develop packages to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.16.1 Write user defined module to solve given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

A module is simply a Python file with a .py extension that can be imported inside another Python program. The name of the Python file becomes the module name. The module contains definitions and implementation of classes, variables, and functions that can be used inside another program.

Types of Python Modules:
There are two types of python modules:
1.Built-in python modules
2.User-defined python modules

a)Built-in python modules

Python built-in modules are a set of libraries that come pre-installed with the Python installation. These modules provide a wide range of functionalities, from file operations and system-related tasks to mathematical computations and web services. The use of these modules simplifies the development process, as developers can leverage the built-in functions and classes for common tasks, providing code reusability, and efficiency.

Some examples of Python built-in modules include “os”, “sys”, “math”, and “datetime”.

b)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 value: "))
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 value: 22
The integer value is 22
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2 GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1.State different types of python modules.

2.Explain how to import multiple modules in python script with example.

3.State advantages of modules in Python.

4.Write a Python program to create a user defined module that will ask your college name and will display the name of the college.

5.Write a Python program to define a module to find Fibonacci Numbers and import the module to another program.

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. 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

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 17: Write a python program to demonstrate the use of following module: 1. Math module 2. Random module 3. OS module

I. Practical Significance

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. Python built-in modules offer a diverse range of functionality and are readily accessible without the requirement of installing extra packages. With these built-in modules, Python provides a comprehensive set of tools and capabilities right out of the box, allowing developers to accomplish various tasks conveniently and without the hassle of additional installations.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO3 - Develop packages to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.17.1 Select appropriate module to solve given problem.

LLO.17.2 Use given module to solve problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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')
```

i) 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
```

ii) 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")
```

iii) 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
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2 GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1.Write a Python program to display calendar of given month using Calendar module.

2.Write a python program to calculate area of circle using inbuilt Math module.

3.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)
```

4.What is the output of the following program?

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

……

……

……

……

……

……

……

……

……

……

……

……

XI. 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

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 18: Write python program to create and use a user defined package for a given problem

I. Practical Significance

Python Packages are a way to organize and structure your Python code into reusable components. Packages help keep your code organized, make it easier to manage and maintain, and allow you to share your code with others. They're like a toolbox where you can store and organize functions and classes for easy access and reuse in different projects. Creating packages in Python allows students to organize their code into reusable and manageable modules.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO3 - Develop packages to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.18.1 Write user defined package to solve given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

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
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty.
1	Computer System	Computer (i3-i5 preferable RAM>2 GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Describe Package in Python.
2. Explain how to create your own package in Python with appropriate example.
3. Differentiate between user defined package and in-built package in Python.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-packages/
2. https://www.prepbytes.com/blog/python/python-packages/
3. https://www.geeksforgeeks.org/create-access-Python-package/

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 19: Write a python program to use of numpy package to perform operation on 2D matrix. Write a python program to use of matplotlib package to represent data in graphical form

I. Practical Significance

Though Python is simple to learn language but it also very strong with its features. Python supports various built in packages. Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. Matplotlib is a powerful plotting library in Python used for creating static, animated, and interactive visualizations. In this practical, students will be able to write a code to use numpy & matplotlib packages .

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO4 - Apply object-oriented approach to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.19.1 Use numpy and matplotlib package to solve given problem.

LLO.19.2 Select appropriate methods from numpy and matplotlib package to solve given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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]]
```

Matplotlib is a popular data visualization library that allows you to create a wide range of plots and charts. To install Matplotlib in Python, you can follow these simple steps.
Step 1: Check Your Python Installation
Open your terminal or command prompt and enter the following command to check your Python version: python --version
Step 2: Open a Terminal or Command Prompt
Step 3: pip install Matplotlib in Python
To install Matplotlib, you can use Python's package manager, pip. Enter the following command:
pip install matplotlib
Step 4: Verify the Installation
After the installation, you can verify it by running a simple Python script.

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('Some numbers')
plt.show()
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2 GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Write a Python program to create two matrices and perform addition, subtraction, multiplication and division operation on matrix using NumPy.
2. Write a python program to concatenate two strings using NumPy.
3. Write a NumPy program to generate six random integers between 10 and 30.
4. Describe Numpy and Matplotlib with suitable example.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. 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/
4. https://matplotlib.org/
5. https://www.geeksforgeeks.org/python-introduction-matplotlib/

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 20: Develop a python program to perform following operations: 1. Creating a Class with method 2. Creating Objects of class 3. Accessing method using object

I. Practical Significance

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods for modifying their state. This practical will describe Class and Object Creation in Python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO4 - Apply object-oriented approach to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.20.1 Write python program using classes and objects to solve a given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

Syntax:

```
class ClassName:
<statement-1>
.
.
<statement-N>
```

Example:

Create a class named MyClass, with a property named x:

```
class MyClass:
x = 5
```

Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc. Everything in Python is an object, and almost everything has attributes and methods.

Example:

Create an object named p1, and print the value of x:

```
p1 = MyClass()
print(p1.x)
```

Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

Example:

```
class Person:
def __init__(self, name, age):
self.name = name
self.age = age

def myfunc(self):
print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2 GB)	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages for answer if needed)

1. Write a Python class to implement pow(x, n).
2. Write a Python class to reverse a string word by word.
3. Write a Python class to convert an integer to a roman numeral.
4. Write a Python class that has two methods: get_String and print_String , get_String accept a string from the user and print_String prints the string in upper case.
5. Write a Python class named Rectangle constructed from length and width and a method that will compute the area of a rectangle.

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-classes-and-objects/
2. https://www.javatpoint.com/python-oops-concepts
3. https://realpython.com/python3-object-oriented-programming/
4. https://www.w3schools.com/python/python_classes.asp

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 21: Write a python program to demonstrate the use of constructors: 1. Default 2. Parameterized 3. Constructor Overloading

I. Practical Significance

A constructor in Python is a special method called when an object is created. Its purpose is to assign values to the data members within the class when an object is initialized. In object-oriented programming, constructor overloading is powerful feature that allow developers to define multiple constructors with the same name but with different parameters. This flexibility enhances code readability, and reusability, and provides a more intuitive interface for developers. In this practical, students will learn about types of constructor and constructor overloading in Python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO4 - Apply object-oriented approach to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.21.1 Write a python program to initialize objects of class using various types of constructors.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Constructors are generally used for instantiating an object. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. In Python, the __init__() method is called the constructor and is always called when an object is created.

In Python, there are two types of constructors:

i)Default Constructor: A default constructor is a constructor that takes no arguments. It is used to create an object with default values for its attributes.

Example:

```
class Person:
    def __init__(self):
        self.name = "John"
        self.age = 30

person = Person()
print(person.name)
print(person.age)
```

Output:

```
John
30
```

ii)Parameterized Constructor: A parameterized constructor is a constructor that takes one or more arguments. It is used to create an object with custom values for its attributes.

Example:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

person = Person("Alice", 25)
print(person.name)
print(person.age)
```

Output:

```
Alice
25
```

Constructor Overloading in Python:

Constructor overloading involves defining multiple constructors within a class, each taking a different set of parameters. While Python does not support explicit constructor overloading like some other languages, developers can achieve a similar effect by using default values and optional parameters.

```
class Rectangle:
def __init__(self, length=1, breadth=1):
self.length = length
self.breadth = breadth

# Creating objects with different number of arguments
rect1 = Rectangle()
print(rect1.length, rect1.breadth)
rect2 = Rectangle(5)
print(rect2.length, rect2.breadth)
rect3 = Rectangle(5, 10)
print(rect3.length, rect3.breadth)
```

Output:

```
1 1
5 1
5 10
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. State the purpose of the self parameter in a constructor.
2. State the advantages & disadvantages of constructor overloading in Python.
3. Write a python program to implement parameterized constructor.
4. Write a python program to implement constructor overloading.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/constructors-in-python/
2. https://www.javatpoint.com/python-constructors
3. https://flexiple.com/python/constructor-overloading-python
4. https://www.tutorialspoint.com/python/python_constructors.htm

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 22: Implement a python program to demonstrate 1. Method Overloading 2. Method Overriding

I. Practical Significance

Method Overloading and Method Overriding in Python are the two main object-oriented concepts that allow programmers to write methods that can process a variety of different types of functionalities with the same name. This helps us to implement Polymorphism and achieve consistency in our code. This practical will let learner to develop programs using method method overloading and method overriding concepts of python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO4 - Apply object-oriented approach to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.22.1 Write a python program to implement polymorphism.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant 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. Method Overloading is an example of Compile time polymorphism. 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('Vaibhav')
```

Output

```
Hey
Hey Vaibhav
```

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. Method overriding is an example of run time polymorphism.

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
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. Differentiate between method overloading and overriding in Python.
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.

……

……

……

……

……

……

……

……

……

……

……

XI. 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.geeksforgeeks.org/python-method-overloading/
4. https://www.geeksforgeeks.org/method-overriding-in-python/
5. https://www.tutorialspoint.com/difference-between-method-overloading-and-method-overriding-in-python
6. https://www.youtube.com/watch?v=HIkoX4pVmrI

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 23: Write python program to demonstrate data hiding

I. Practical Significance

Data hiding is a concept which underlines the hiding of data or information from the user. It is one of the key aspects of Object-Oriented programming strategies. Data hiding minimizes system complexity for increase robustness by limiting interdependencies between software requirements. Data hiding is also known as information hiding. In class, if we declare the data members as private so that no other class can access the data members, then it is a process of hiding data. This practical will let learner to develop program to demonstrate data hiding in Python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO4 - Apply object-oriented approach to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.23.1 Write a python program use data hiding concept in python.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

The Python introduces Data Hiding as isolating the user from a part of program implementation. Some objects in the module are kept internal, unseen, and unreachable to the user. Modules in the program are easy enough to understand how to use the application, but the client cannot know how the application functions. Thus, data hiding imparts security, along with discarding dependency. Data hiding in Python is the technique to defend access to specific users in the application. Python is applied in every technical area and has a user-friendly syntax and vast libraries. Data hiding in Python is performed using the __ double underscore before done prefix. This makes the class members non-public and isolated from the other classes.

Example: 1

```
class Solution:
   __privateCounter = 0

   def sum(self):
      self.__privateCounter += 1
      print(self.__privateCounter)

count = Solution()
count.sum()
count.sum()
```

```
# Here it will show error because it unable to access private member
print(count.__privateCounter)
```

Example: 2
To rectify the error, we can access the private member through the class name

```
class Solution:
    __privateCounter = 0

    def sum(self):
        self.__privateCounter += 1
        print(self.__privateCounter)

count = Solution()
count.sum()
count.sum()

# Here we have accessed the private data member through class name.
print(count._Solution__privateCounter)
```

Output:

```
1
2
2
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. State the importance of data hiding in Python.
2. State advantages and disadvantages of data hiding.
3. Write a Python program to implement data hiding.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/data-hiding-in-python/
2. https://www.tutorialspoint.com/data-hiding-in-python
3. https://www.scaler.com/topics/data-hiding-in-python/
4. https://www.javatpoint.com/data-hiding-in-python/

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 24: Write a python program to implement 1. Single inheritance 2. Multiple Inheritance 3. Multilevel 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. This practical will let learner to develop programs using various types of inheritance in python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO4 - Apply object-oriented approach to solve given problem using python.

IV. Laboratory Learning Outcome(s)

LLO.24.1 Select appropriate type of inheritance to solve given problem.

LLO.24.2 Write python program using inheritance to solve given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

a) Single 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 single 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:

```
class A:
# variable of class A
# functions of class A
class B:
# variable of class B
# functions of class B
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
```

c) Multilevel inheritance

- Multilevel Inheritance in Python is a type of Inheritance in which a class inherits from a class, which itself inherits from another class. It allows a class to inherit properties and methods from multiple parent classes, forming a hierarchy similar to a family tree.

Syntax:

```
class A:
# variable of class A
# functions of class A
class B(A):
# class B inheriting property of class A
# add more properties to class B
class C(B):
# class C inheriting property of class A & B
# add more properties to class C
```

Example:

```
# GrandParent class created
class GrandParent:
    grandparentname = ""
    def show_grandparent(self):
        print(self.grandparentname)
# Parent class created
class Parent(GrandParent):
    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.grandparentname = "Vaibhav" # Access GrandParent class attributes
ch1.parentname = "Vijay" # Access Parent class attributes
ch1.childname = "Parth"
ch1.show_grandparent() # Access GrandParent class method
ch1.show_parent() # Access Parent class method
ch1.show_child() # Access Child class method
```

VII. Required Resources:

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical Students may add extra pages if needed)

1. State the use of inheritance. List different types of inheritance.
2. Write Python program to read and print students information using single inheritance..
3. Write a Python program to implement multiple inheritance.
4. Write a Python program to implement multilevel inheritance.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. 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

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 25: Implement Python program to perform following operations using panda package: 1. Create Series from Array 2. Create Series from List 3. Access element of series 4. Create DataFrame using List or dictionary

I. Practical Significance

The Pandas Series can be defined as a one-dimensional array that is capable of storing various data types. One can easily convert the list, tuple, and dictionary into series using "series' method. The row labels of series are called the index. This practical will make students to work on Panda package supported in Python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO5 - Use relevant built-in python package to develop application.

IV. Laboratory Learning Outcome(s)

LLO.25.1 Use panda package and its appropriate functions/methods to solve a given problem.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). It has the following parameters:

- data: It can be any list, dictionary, or scalar value.
- index: The value of the index should be unique and hashable. It must be of the same length as data. If we do not pass any index, default np.arrange(n) will be used.
- dtype: It refers to the data type of series.
- copy: It is used for copying the data.

One can alternatively install the libraries using pip by running the following commands from terminal: pip install pandas

a) Create Series from Array: In order to create a series from array, we have to import a numpy module and have to use array() function.

Example:

```
# import pandas as pd
import pandas as pd

# import numpy as np
import numpy as np
```

```
# simple array
data = np.array(['g','e','e','k','s'])
ser = pd.Series(data)
print(ser)
```

b) Create Series from List: In order to create a series from list, we have to first create a list after that we can create a series from list.

Example:

```
import pandas as pd
# a simple list
list = ['g', 'e', 'e', 'k', 's']
# create series form a list
ser = pd.Series(list)
print(ser)
```

c) Access element of series: There are two ways through which we can access element of series, they are :

i)Accessing Element from Series with Position

ii)Accessing Element Using Label (index)

Example:

```
# import pandas and numpy
import pandas as pd
import numpy as np
# creating simple array
data = np.array(['g','e','e','k','s','f', 'o','r','g','e','e','k','s'])
ser = pd.Series(data)
#retrieve the first element
print(ser[:5])
```

d)Create DataFrame using List or dictionary:

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Example:

```
import pandas as pd # importing the pandas package
Li = [100,200,300,400, 500] # Assigning the value to list(Li)
df = pd.DataFrame(Li) # Creating the DataFrame
print(df) # Printing the values of DataFrame
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. Describe Pandas library of Python.
2. Write a Python program to create series from array using Panda.
3. Write a Python program to create series from list using Panda.
4. Write a Python program to access element of series using Panda.
5. Write a Python program to create DataFrame using list or dictionary using Panda.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-pandas-series/
2. https://www.javatpoint.com/python-pandas-series
3. https://www.w3schools.com/python/pandas/pandas_dataframes.asp
4. https://www.tutorialspoint.com/how-to-create-a-pandas-dataframe-using-a-list

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 26: Implement python program to load a CSV file into a Pandas DataFrame and perform operations.

I. Practical Significance

Python pandas is a widely-used Python library for data science, analysis, and machine learning that offers a flexible and intuitive way to handle data sets of all sizes. One of the most important functionalities of pandas is the tools it provides for reading and writing data. For data available in a tabular format and stored as a CSV file, one can use pandas to read it into memory using the read_csv() function, which returns a pandas dataframe. This practical will enable student to work on Pandas DataFrame and perform various operations..

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO5 - Use relevant built-in python package to develop application.

IV. Laboratory Learning Outcome(s)

LLO.26.1 Write python program to read CSV file using the panda package.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

CSV files are the “comma-separated values”, these values are separated by commas, this file can be viewed like an excel file. In Python, Pandas is the most important library coming to data science. We need to deal with huge datasets while analyzing the data, which usually can get in CSV file format. Creating a pandas data frame using CSV files can be achieved in multiple ways.

Method #1: Using read_csv() method: read_csv() is an important pandas function to read csv files and do operations on it.

Example:

```
# Python program to illustrate
# creating a data frame using CSV files

# import pandas module
import pandas as pd

# creating a data frame
df = pd.read_csv("CardioGoodFitness.csv")
print(df.head())
```

Method #2: Using read_table() method: read_table() is another important pandas function to read csv files and create data frame from it.

Example:

```
    # Python program to illustrate
```

```
# creating a data frame using CSV files

# import pandas module
import pandas as pd

# creating a data frame
df = pd.read_table("CardioGoodFitness.csv", delimiter =", ")
print(df.head())
```

Method #3: Using the csv module: One can directly import the csv files using the csv module and then create a data frame using that csv file.

Example:

```
# Python program to illustrate
# creating a data frame using CSV files

# import pandas module
import pandas as pd
# import csv module
import csv

with open("CardioGoodFitness.csv") as csv_file:
# read the csv file
csv_reader = csv.reader(csv_file)

# now we can use this csv files into the pandas
df = pd.DataFrame([csv_reader], index = None)

# iterating values of first column
for val in list(df[1]):
print(val)
```

Beyond simply exploring the data, one can manipulate it in various ways, such as sorting, filtering, merging, and pivoting.

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. Explain DataFrame in Python pandas.
2. State and explain various operations which can be performed on CSV files in Python using pandas.
3. Write a Python program to read and write with CSV files using pandas.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/creating-a-dataframe-using-csv-files/
2. https://www.datacamp.com/tutorial/pandas-read-csv
3. https://www.w3schools.com/python/pandas/pandas_csv.asp
4. https://www.tutorialspoint.com/creating-a-dataframe-using-csv-files
5. https://www.programiz.com/python-programming/csv

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 27: Write python GUI program to import Tkinter package and create a window and set its title

I. Practical Significance

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run. This practical will enable student to write GUI programs using Tkinter package.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO5 - Use relevant built-in python package to develop application.

IV. Laboratory Learning Outcome(s)

LLO.27.1 Use appropriate packages in a python program to create GUI applications.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python Tkinter is the fastest and easiest way to create GUI applications.

To create a Tkinter Python app, you follow these basic steps:

1. Import the tkinter module: This is done just like importing any other module in Python.
2. Create the main window (container): The main window serves as the container for all the GUI elements you'll add later.
3. Add widgets to the main window: You can add any number of widgets like buttons, labels, entry fields, etc., to the main window to design the interface as desired.
4. Apply event triggers to the widgets: You can attach event triggers to the widgets to define how they respond to user interactions.

Create First Tkinter GUI Application:

There are two main methods used which the user needs to remember while creating the Python application with GUI.

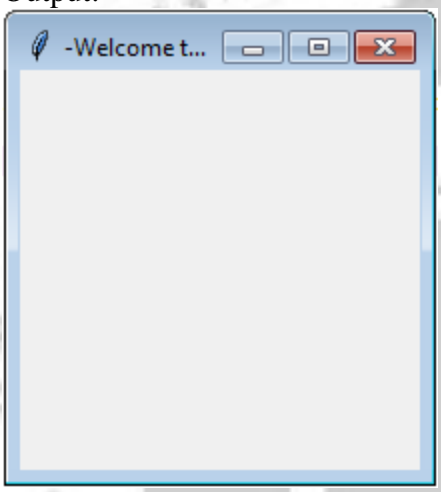
Tk(): To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

mainloop(): There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur, and process the event as long as the window is not closed.

Example:

```
import tkinter as tk
# Create instance
parent = tk.Tk()
# Add a title
parent.title("-Welcome to Python tkinter Basic exercises-")
# Start GUI
parent.mainloop()
```

Output:



VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. Explain various GUI packages available in Python.
2. State the importance of Tkinter package in Python.
3. Write GUI program to import Tkinter package and create a window and set its title.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-gui-tkinter/
2. https://realpython.com/python-gui-tkinter/
3. https://www.w3resource.com/python-exercises/tkinter/python-tkinter-basic-exercise-1.php
4. https://www.topcoder.com/thrive/articles/python-for-user-interface-tkinter-basics

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 28: Write python GUI program that adds labels and buttons to the Tkinter window

I. Practical Significance

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux.Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run. This practical will enable student to write GUI programs to add widgets using Tkinter package.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO5 - Use relevant built-in python package to develop application.

IV. Laboratory Learning Outcome(s)

LLO.28.1 Write python program to create GUI based python applications using appropriate python packages.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

1. Label: It refers to the display box where you can put any text or image which can be updated any time as per the code.

The general syntax is:

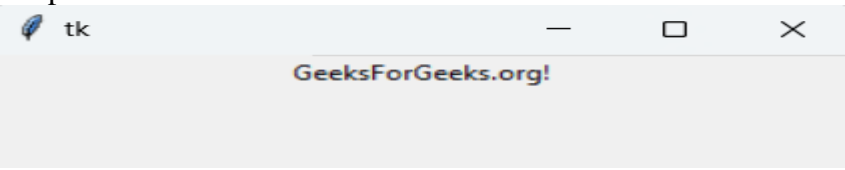
w=Label(master, option=value)

master is the parameter used to represent the parent window.

Example:

```
from tkinter import *
root = Tk()
w = Label(root, text='GeeksForGeeks.org!')
w.pack()
root.mainloop()
```

Output:



2. Button:

To add a button in your application, this widget is used. The general syntax is:

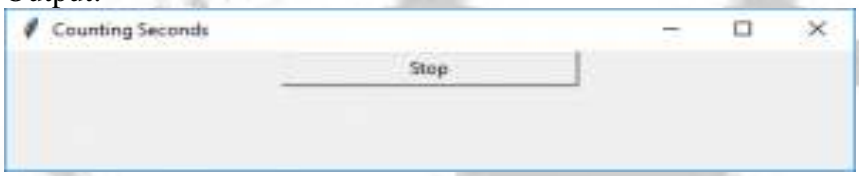
w=Button(master, option=value)

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas.

Example:

```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=25, command=r.destroy)
button.pack()
r.mainloop()
```

Output:



3. Entry:

It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used. The general syntax is:

w=Entry(master, option=value)

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

4. CheckButton:

To select any number of options by displaying a number of options to a user as toggle buttons. The general syntax is:

w = CheckButton(master, option=value)

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas.

5. RadioButton:

It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option. The general syntax is:

w = RadioButton(master, option=value)

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas.

6. Listbox:

It offers a list to the user from which the user can accept any number of options. The general syntax is:

w = Listbox(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

7. Scrollbar:

It refers to the slide controller which will be used to implement listed widgets. The general syntax is:

w = Scrollbar(master, option=value)

master is the parameter used to represent the parent window.
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

8. Menu:

It is used to create all kinds of menus used by the application. The general syntax is:

w = Menu(master, option=value)
master is the parameter used to represent the parent window.
There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas.

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. Explain various Tkinter widgets available in Python.
2. Write a Python program using Tkinter that creates a CheckButton.
3. Write a Python program using Tkinter that creates a RadioButton.
4. Write a Python program using Tkinter that creates a Menu.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.geeksforgeeks.org/python-gui-tkinter/
2. https://realpython.com/python-gui-tkinter/
3. https://www.w3resource.com/python-exercises/tkinter/python-tkinter-basic-exercise-6.php
4. https://www.topcoder.com/thrive/articles/python-for-user-interface-tkinter-basics

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 29: Write program to create a connection between database and python

I. Practical Significance

The database is a well-organized collection of structured information or data stored in a computer system. In database, the data is arranged in the tabular form, and we can access that information or data by querying. Python can be used to connect the Database. MySQL is one of the most popular Databases. In this practical, student will be able to establish a connection to MySQL via Python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO5 - Use relevant built-in python package to develop application.

IV. Laboratory Learning Outcome(s)

LLO.29.1 Write python program to connect database.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

Python is a high-level, general-purpose, and very popular programming language. Basically, it was designed with an emphasis on code readability, and programmers can express their concepts in fewer lines of code. We can also use Python with SQL.

Connecting MySQL with Python:

The following steps are required to connect SQL with Python:

Step 1: Download and Install the free MySQL database

Step 2: After installing the MySQL database, open your Command prompt.

Step 3: Navigate your Command prompt to the location of PIP.

Step 4: Now run the command given below to download and install “MySQL Connector”.

```
pip install mysql-connector-python
```

Step 5: Test MySQL Connector

To check if the installation was successful, or if you already installed “MySQL Connector”, go to your IDE and run the given below code :

```
import mysql.connector
```

if the above code gets executed with no errors, “MySQL Connector” is ready to be used.

Step 6: Create a Connection Object:

The mysql.connector provides the connect() method used to create a connection between the MySQL database and the Python application. The syntax is given below.

Syntax:

Conn_obj= mysql.connector.connect(host = <hostname>, user = <username>, passwd = <password>)
The connect() function accepts the following arguments.
Hostname - It represents the server name or IP address on which MySQL is running.
Username - It represents the name of the user that we use to work with the MySQL server. By default, the username for the MySQL database is root.
Password - The password is provided at the time of installing the MySQL database. We don't need to pass a password if we are using the root.
Database - It specifies the database name which we want to connect. This argument is used when we have multiple databases.

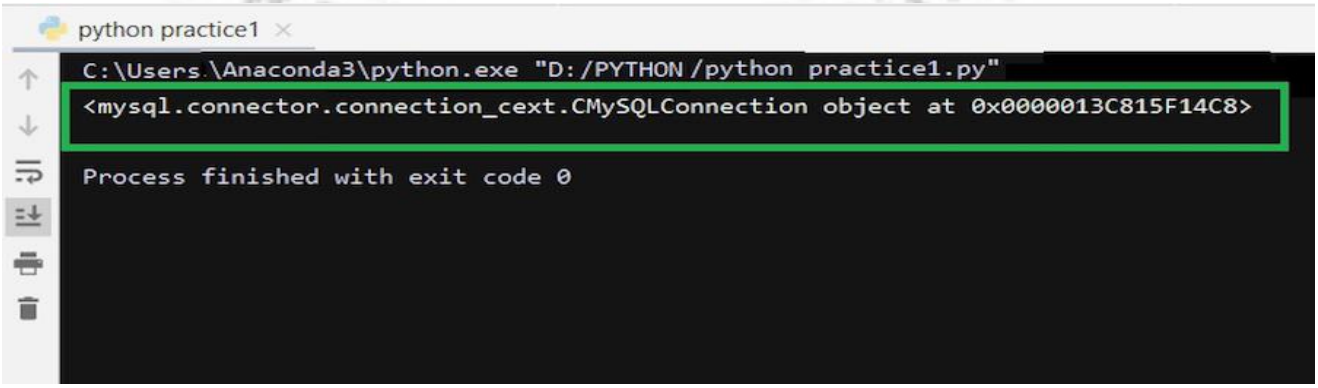
Example:

```
# Importing module
import mysql.connector

# Creating connection object
mydb = mysql.connector.connect(
    host = "localhost",
    user = "yourusername",
    password = "your_password")

# Printing the connection object
print(mydb)
```

Output:



VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. State and explain any five SQL Queries.
2. Explain steps to connect Python with SQL Database.
3. Write program to connect Python with PostgreSQL Database.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.javatpoint.com/how-to-connect-database-in-python
2. https://www.geeksforgeeks.org/how-to-connect-python-with-sql-database/
3. https://www.opensourceforu.com/2019/04/database-programming-python/
4. https://www.w3schools.com/python/python_mysql_getstarted.asp

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	

Practical No. 30: Implement python program to select records from the database table and display the result

I. Practical Significance

The database is a well-organized collection of structured information or data stored in a computer system. In database, the data is arranged in the tabular form, and we can access that information or data by querying. Python can be used to connect the Database. MySQL is one of the most popular Databases. In this practical, student will be able to perform various database operations via Python.

II. Industry / Employer Expected Outcome(s)

1. Develop applications using python to solve given problem.

III. Course Level Learning Outcome(s)

CO5 - Use relevant built-in python package to develop application.

IV. Laboratory Learning Outcome(s)

LLO.30.1 Write python program to display the content from database.

V. Relevant Affective Domain related Outcomes

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

VI. Relevant Theoretical Background

The following steps are required to connect SQL with Python:

1.Install MySQL Driver:

First, we need a MySQL driver in our system. Install the MySQL software and configure the settings. We will use the MySQL connector driver, which is installed using the pip command. Open a command prompt and type the following command.

python -m pip install mysql-connector-python

2.Create a Connection Object:

The mysql.connector provides the connect() method used to create a connection between the MySQL database and the Python application. The syntax is given below.

Syntax:

Conn_obj= mysql.connector.connect(host = <hostname>, user = <username>, passwd = <password>)

The connect() function accepts the following arguments.

Hostname - It represents the server name or IP address on which MySQL is running.

Username - It represents the name of the user that we use to work with the MySQL server. By default, the username for the MySQL database is root.

Password - The password is provided at the time of installing the MySQL database. We don't need to pass a password if we are using the root.

Database - It specifies the database name which we want to connect. This argument is used when we have multiple databases.

3.Create a Cursor Object:

The connection object is necessary to create because it provides the multiple working environments the same connection to the database. The cursor() function is used to create the cursor object. It is important for executing the SQL queries. The syntax is given below.

Syntax:

Con_obj = conn.cursor()

4.Executes the Query:

Various operations can be performed on database by executing the query.

Example:

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword")

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

for x in mycursor:
  print(x)
```

Reading data from a MYSQL table using Python:

READ Operation on any database means to fetch some useful information from the database. You can fetch data from MYSQL using the fetch() method provided by the mysql-connector-python.

The cursor.MySQLCursor class provides three methods namely fetchall(), fetchmany() and, fetchone() where,

The fetchall() method retrieves all the rows in the result set of a query and returns them as list of tuples.

The fetchone() method fetches the next row in the result of a query and returns it as a tuple.

The fetchmany() method is similar to the fetchone() but, it retrieves the next set of rows in the result set of a query, instead of a single row.

Example:

```
import mysql.connector
#establishing the connection
conn = mysql.connector.connect( user='root', password='password', host='127.0.0.1', database='mydb')

#Creating a cursor object using the cursor() method
cursor = conn.cursor()
```

```
#Retrieving single row
sql = '''SELECT * from EMPLOYEE'''
#Executing the query
cursor.execute(sql)
#Fetching 1st row from the table
result = cursor.fetchone();
print(result)
#Fetching remaining rows from the table
result = cursor.fetchall();
print(result)
#Closing the connection
conn.close()
```

VII. Required Resources

Sr. No.	Name of the Resources	Specifications	Qty
1	Computer System	Computer (i3-i5 preferable RAM>2GB) with Windows or Linux OS	01system for each student
2	Python Interpreter / IDE	Any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.	
3	Printer	Laser	01 for a batch
4	Projector / Smart Board and Relevant Charts	-	01 for a batch

VIII. Precautions to be followed

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Start and Shutdown system with proper procedure.
4. Don't forget to save file before execution.

IX. Conclusion

……

……

……

X. Practical related questions (Teacher may assign more questions related to practical. Students may add extra pages if needed)

1. Write Python program to create database in MySQL.
2. Write Python program to create table in MySQL database.

3. Write program to insert record into a table in MySQL database.
4. Write program to delete record from a table in MySQL database.
5. Write program to update record in a table in MySQL database.

……

……

……

……

……

……

……

……

……

……

……

……

……

XI. References/Suggestions for further reading

1. https://www.javatpoint.com/how-to-connect-database-in-python
2. https://www.geeksforgeeks.org/how-to-connect-python-with-sql-database/
3. https://www.opensourceforu.com/2019/04/database-programming-python/
4. https://www.w3schools.com/python/python_mysql_select.asp

XII. Assessment Scheme (50 Marks)

S. No.	Weightage- Process related: 60%	Marks-30
1.	Logic formation: 40%	
2.	Debugging ability: 10%	
3.	Correctness of program code: 10%	
	Weightage- Product related: 40%	**Marks-20**
4.	Expected output: 15%	
5.	Timely completion of practical: 15%	
6.	Answer to sample questions: 10%	
	Total 50	
	Dated Signature of Course Teacher	