Vidyavardhini's
College of Engineering & Technology
Vasai Road (W)

Department of Computer Engineering

Laboratory Manual

Semester	IV	**Class**	**S.E**
Course Code	CSL405		
Course Name	Skill Based Lab Course: Python Programming		

Vidyavardhini's College of Engineering & Technology

Vision

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

Mission

- To provide technologically inspiring environment for learning.
- To promote creativity, innovation and professional activities.
- To inculcate ethical and moral values.
- To cater personal, professional and societal needs through quality education.

Department Vision:

To evolve as a center of excellence in the field of Computer Engineering to cater to industrial and societal needs.

Department Mission:

- To provide quality technical education with the aid of modern resources.
- Inculcate creative thinking through innovative ideas and project development.
- To encourage life-long learning, leadership skills, entrepreneurship skills with ethical & moral values.

Program Education Objectives (PEOs):

PEO1: To facilitate learners with a sound foundation in the mathematical, scientific and engineering fundamentals to accomplish professional excellence and succeed in higher studies in Computer Engineering domain

PEO2: To enable learners to use modern tools effectively to solve real-life problems in the field of Computer Engineering.

PEO3: To equip learners with extensive education necessary to understand the impact of computer technology in a global and social context.

PEO4: To inculcate professional and ethical attitude, leadership qualities, commitment to societal responsibilities and prepare the learners for life-long learning to build up a successful career in Computer Engineering.

Program Specific Outcomes (PSOs):

PSO1: Analyze problems and design applications of database, networking, security, web technology, cloud computing, machine learning using mathematical skills, and computational tools.

PSO2: Develop computer-based systems to provide solutions for organizational, societal problems by working in multidisciplinary teams and pursue a career in the IT industry.

Program Outcomes (POs):

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

- **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Objective

1	To understand Basics of Python programming
2	To work with Decision Making, Data structure and Functions in Python
3	To implement Object Oriented Programming using Python
4	To implement Web framework for developing

Course Outcomes

At the end of the course student will be able to:		**Action verbs**	**Bloom's Level**
CSL405.1	Apply concepts of Input / Output, control statements and object oriented programming in python for performing arithmetic operations	Apply	Apply (level 3)
CSL405.2	Use features of files, directories and regular expression in python for file manipulation	Use	Apply (level 3)
CSL405.3	Implement linked list, stacks, queues and dequeues data structures	Implement	Apply (level 3)
CSL405.4	Develop Graphical User Interface, perform database operations and create web applications with Django web framework	Develop and Create	Create (level 6)
CSL405.5	Implement multi-threading in python	Implement	Apply (level 3)
CSL405.6	Use NumPy and Pandas packages for matrix manipulation and data analysis	Use	Apply (level 3)

Mapping of Experiments with Course Outcomes

Sr. No	Title	CSL405.1	CSL405.2	CSL405.3	CSL405.4	CSL405.5	CSL405.6
1	Program to perform arithmetic operations by accepting values from users	3	-	-	-	-	-
2	To implement Conditional Statements and Loop in python.	3	-	-	-	-	-
3	To explore basic data types of python like strings, list, dictionaries and tuples.	3	-	-	-	-	-
4	Creating functions, classes and objects using python.	3	-	-	-	-	-
5	Exploring Files and directories: Python program to append data to existing file and then display the entire file	-	3	-	-	-	-
6	Program for data structure using built in function for link list, stack and queues.	-	-	3	-	-	-
7	Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes.	-	-	-	3	-	-
8	Program to demonstrate CRUD (create, read, update and delete) operations on database (SQLite/ MySQL) using python	-	-	-	3	-	-
9	Creating web application using Django web framework to demonstrate functionality of user login and registration (also validating user detail using regular expression)	-	-	-	3	-	-
10	Demonstrate the concept of Multi-threading	-	-	-	-	3	-
11	Program to manipulate arrays using NumPy	-	-	-	-	-	3
12	Program to demonstrate data frame creation and Manipulation using Pandas	-	-	-	-	-	3

Enter correlation level 1, 2 or 3 as defined below

1: Slight (Low) 2: Moderate (Medium) 3: Substatial (High)

If there is no correlation put "—".

Index

Sr. No	Title	DOP	DOC	Page No.	Remark
1.	Program to perform arithmetic operations by accepting values from users				
2.	To implement Conditional Statements and Loop in python				
3.	To explore basic data types of python like strings, list, dictionaries and tuples				
4.	Creating functions, classes and objects using python				
5.	Exploring Files and directories: Python program to append data to existing file and then display the entire file				
6.	Program for data structure using built in function for link list, stack and queues				
7.	Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes				
8.	Program to demonstrate CRUD (create, read, update and delete) operations on database (SQLite/ MySQL) using python				
9.	Creating web application using Django web framework to demonstrate functionality of user login and registration (also validating user detail using regular expression)				
10.	Demonstrate the concept of Multi-threading				
11.	Program to manipulate arrays using NumPy				
12.	Program to demonstrate data frame creation and Manipulation using Pandas				

D.O.P: Date of performance

D.O.C : Date of correction

Experiment No.1
Program to perform arithmetic operations by accepting values from users
Date of Performance:
Date of Submission:

Experiment No. 1

Title: Program to perform arithmetic operations by accepting values from users

Aim: To write a program to perform arithmetic operations by accepting values from users

Objective: To introduce basic concepts in Python

Theory:

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.
There are 7 arithmetic operators in Python :

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Modulus
6. Exponentiation
7. Floor division

Code:

```
num1= int(input("Enter 1st number"))
num2 = int(input("Enter 2nd number"))

sum = num1 + num2
subtract=num1-num2
multiply=num1*num2
divide=num1/num2
modulus=num1%num2
exponential=num1**num2
floor_div=num1//num2

print(sum)
print(subtract)
print(multiply)
```

```
print(divide)
print(modulus)
print(exponential)
print(floor_div)
```

Output:

Conclusion: Arithmetic operators have been implemented.

Experiment No. 2
To implement Conditional Statements and Loop in python
Date of Performance:
Date of Submission:

Experiment No. 2

Title: To implement Conditional Statements and Loop in python

Aim: To study, and implement Conditional Statements and Loop in python

Objective: To introduce Conditional Statements and Loop in python

Theory:

1. Conditional Statements

There comes situations in real life when we need to do some specific task and based on some specific conditions and, we decide what should we do next. Similarly there comes a situation in programming where a specific task is to be performed if a specific condition is True. In such cases, conditional statements can be used. The following are the conditional statements provided by Python.

if

if..else

Nested if

if-elif statements.

Let us go through all of them.

if Statement

If the simple code of block is to be performed if the condition holds true than if statement is used. Here the condition mentioned holds true then the code of block runs otherwise not.

if..else Statment

In conditional if Statement the additional block of code is merged as else statement which is performed when if condition is false.

Nested if Statement

if statement can also be checked inside other if statement. This conditional statement is called nested if statement. This means that inner if condition will be checked only if outer if condition is true and by this, we can see multiple conditions to be satisfied.

if-elif Statment

The if-elif statement is shoutcut of if..else chain.While using if-elif statement at the end else block is added which is performed if none of the above if-elif statement is true.

2. Looping in python

Python programming language provides following types of loops to handle looping requirements. Python provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

While Loop:

In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.

for in Loop:

For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is "for in" loop which is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.

Code:

```
print("To find: Even or odd")


num4 = int(input("Enter number "))
```

```
if num4 % 2 == 0:

    print("Even")

else:

    print("Odd")

print("To find: Find greatest of 3")
num1 = int(input("Enter number 1 "))
num2 = int(input("Enter number 2 "))
num3 = int(input("Enter number 3 "))

if num1 > num2:

    if num1 > num3:

        print("Greatest number is ", num1)

    else:

        print("Greatest number is ", num3)
```

```
else:

    if num2 > num1:

        if num2 > num3:
            print("Greatest number is ", num2)

    else:

        print("Greatest number is ", num3)

print("To find: First 10 Natural numbers")

for i in range(1, 11, 1):
    print(i)
```

Output:

```
C:\Users\student\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\student\PycharmProjects\pythonProject1\main.py
To find: Even or odd
Enter number 12
Even
To find: Find greatest of 3
Enter number 1 14
Enter number 2 23
Enter number 3 4
Greatest number is  23
To find: First 10 Natural numbers
1
2
3
4
5
6
7
8
9
10

Process finished with exit code 0
```

Conclusion: Conditional and Looping statements have been implemented.

Experiment No. 3
To explore basic data types of python like strings, list, dictionaries and tuples
Date of Performance:
Date of Submission:

Experiment No. 3

Title: To explore basic data types of python like strings, list, dictionaries and tuples.

Aim: To study and explore basic data types of python like strings, list, dictionaries and tuples.

Objective: To introduce basic data types of python

Theory:

Lists: are just like dynamic sized arrays, declared in other languages (vector in C++ and ArrayList in Java). Lists need not be homogeneous always which makes it a most powerful tool in Python.

Tuple: A Tuple is a collection of Python objects separated by commas. In someways a tuple is similar to a list in terms of indexing, nested objects and repetition but a tuple is immutable unlike lists that are mutable.

Set: A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set.

Dictionary: in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key value is provided in the dictionary to make it more optimized.

List, Tuple, Set, and Dictionary are the data structures in python that are used to store and organize the data in an efficient manner.

List	Tuple	Set	Dictionary
List is a non-homogeneous data structure which stores the elements in single row and multiple rows and columns	Tuple is also a non-homogeneous data structure which stores single row and multiple rows and columns	Set data structure is also non-homogeneous data structure but stores in single row	Dictionary is also a non-homogeneous data structure which stores key value pairs

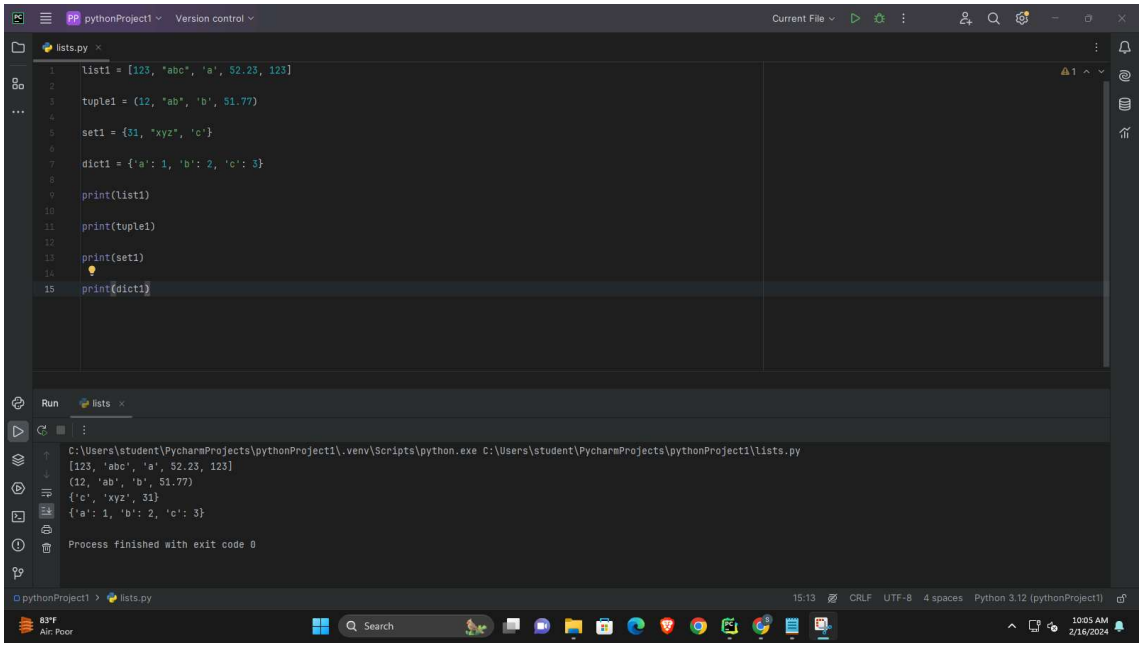
List can be represented by [ ]	Tuple can be represented by ( )	Set can be represented by { }	Dictionary can be represented by { }
List allows duplicate elements	Tuple allows duplicate elements	Set will not allow duplicate elements	Set will not allow duplicate elements but keys are not duplicated
List can use nested among all	Tuple can use nested among all	Set can use nested among all	Dictionary can use nested among all
Example: [1, 2, 3, 4, 5]	Example: (1, 2, 3, 4, 5)	Example: {1, 2, 3, 4, 5}	Example: {1, 2, 3, 4, 5}
List can be created using **list()** function	Tuple can be created using **tuple()** function.	Set can be created using **set()** function	Dictionary can be created using **dict()** function.
List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered
		Creating a set a=set()	
Creating an empty list l=[]	Creating an empty Tuple t=()	b=set(a)	

Code:

```
list1 = [123, "abc", 'a', 52.23, 123]
```

```
tuple1 = (12, "ab", 'b', 51.77)

set1 = {31, "xyz", 'c'}

dict1 = {'a': 1, 'b': 2, 'c': 3}


print(list1)

print(tuple1)

print(set1)

print(dict1)
```

Output:



Conclusion:

Basic data types of python has been studied and implemented.

Experiment No. 4
Creating functions, classes and objects using python
Date of Performance:
Date of Submission:

Experiment No. 4

Title: Creating functions, classes and objects using python

Aim: To study and create functions, classes and objects using python

Objective: To introduce functions, classes and objects in python

Theory:

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

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 (defined by their class) for modifying their state.

To understand the need for creating a class let's consider an example, let's say you wanted to track the number of dogs that may have different attributes like breed, age. If a list is used, the first element could be the dog's breed while the second element could represent its age. Let's suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it's the exact need for classes.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

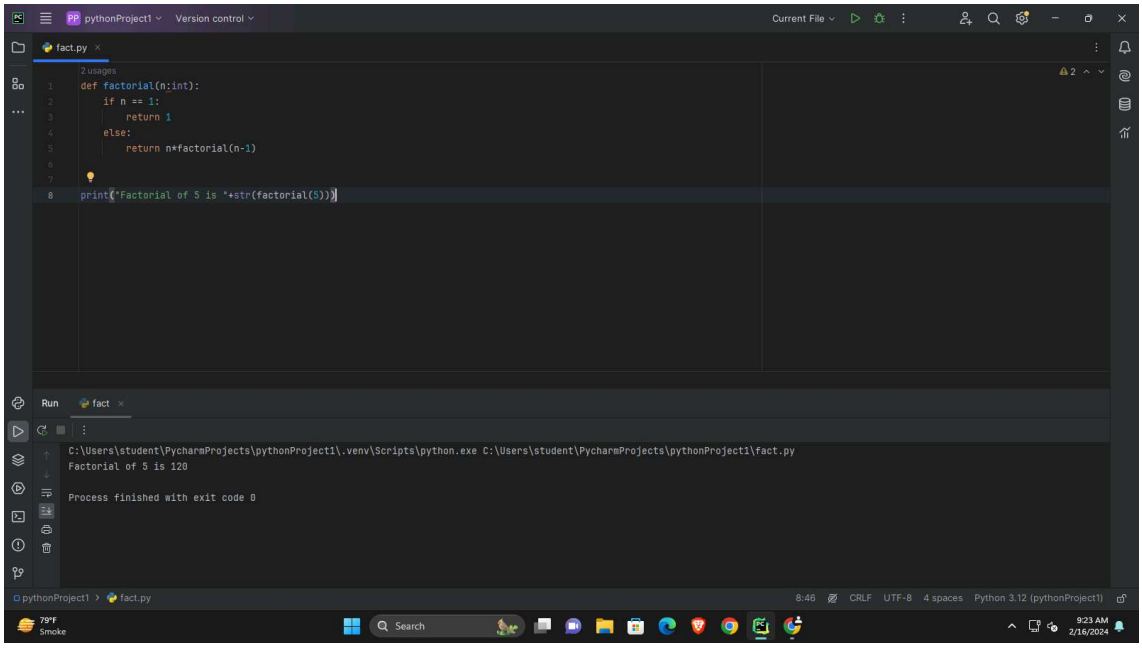
Code:

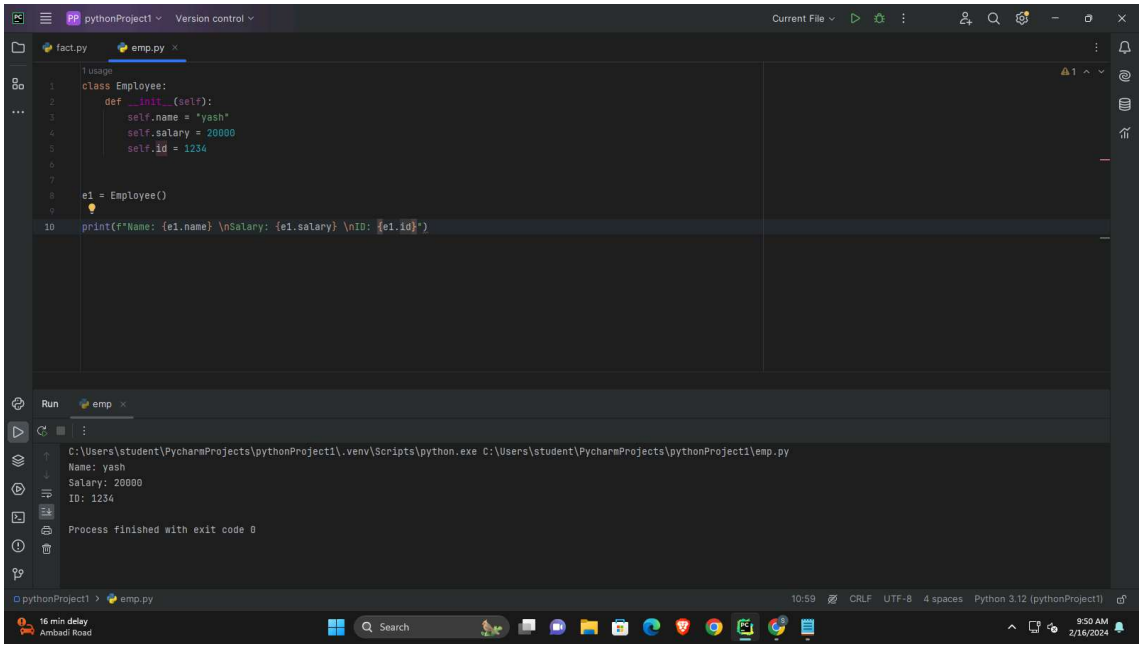
```
def factorial(n:int):
```

```
if n == 1:
    return 1
else:
    return n*factorial(n-1)
```

```
print("Factorial of 5 is "+str(factorial(5)))
```

Output:





Conclusion:

Classes object and functions have been implemented.

Experiment No. 5
Exploring Files and directories: Python program to append data to existing file and then display the entire file
Date of Performance:
Date of Submission:

Experiment No. 5

Title: Exploring Files and directories: Python program to append data to existing file and then display the entire file

Aim: To Exploring Files and directories: Python program to append data to existing file and then display the entire file

Objective: To Exploring Files and directories

Theory:

Directory also sometimes known as a folder are unit organizational structure in computer's file system for storing and locating files or more folders. Python now supports a number of APIs to list the directory contents. For instance, we can use the Path.iterdir, os.scandir, os.walk, Path.rglob, or os.listdir functions.

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

Working of open() function

We use open () function in Python to open a file in read or write mode. As explained above, open ( ) will return a file object. To return a file object we use open() function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: open(filename, mode). There are three kinds of mode, that Python provides and how files can be opened:

" r ", for reading.

" w ", for writing.

" a ", for appending.

" r+ ", for both reading and writing

```
fs = open('test.txt','w')

fs.write('hello world')

fs.close()

fs1 = open('test.txt', 'r')

text = fs1.read()

print(text)

fs1.close()

fs = open('test.txt','a')

fs.write(', how are you?')

fs.close()

fs1 = open('test.txt', 'r')

text = fs1.read()

print(text)

fs1.close()
```

Conclusion: Directories and files have been explored.

Experiment No. 6
Program for data structure using built in function for link list, stack and queues
Date of Performance:
Date of Submission:

Experiment No. 6

Title: Program for data structure using built in function for link list, stack and queues

Aim: To study and implement data structure using built in function for link list, stack and queues

Objective: To introduce data structures in python

Theory:

Stacks -the simplest of all data structures, but also the most important. A stack is a collection of objects that are inserted and removed using the LIFO principle. LIFO stands for "Last In First Out". Because of the way stacks are structured, the last item added is the first to be removed, and vice-versa: the first item added is the last to be removed.

Queues – essentially a modified stack. It is a collection of objects that are inserted and removed according to the FIFO (First In First Out) principle. Queues are analogous to a line

at the grocery store: people are added to the line from the back, and the first in line is the first that gets checked out – BOOM, FIFO!

Linked Lists

The Stack and Queue representations I just shared with you employ the python-based list to store their elements. A python list is nothing more than a dynamic array, which has some disadvantages.

The length of the dynamic array may be longer than the number of elements it stores, taking up precious free space.

Insertion and deletion from arrays are expensive since you must move the items next to them over

Using Linked Lists to implement a stack and a queue (instead of a dynamic array) solve both of these issues; addition and removal from both of these data structures (when implemented with a linked list) can be accomplished in constant O(1) time. This is a HUGE advantage when dealing with lists of millions of items.

Linked Lists – comprised of 'Nodes'. Each node stores a piece of data and a reference to its next and/or previous node. This builds a linear sequence of nodes. All Linked Lists store a head, which is a reference to the first node. Some Linked Lists also store a tail, a reference to the last node in the list.

Code:

```
ll=[]


ll.append(4)


ll.append(3)
```

```
print(l1)

l1.insert(1,2)

print(l1)

l1.remove(2)

l1.insert(1,5)

print(l1)

print(l1.index(5))

print(len(l1))
```

Output:

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

Conclusion: Data structures python has been studied and implemented.

Experiment No. 7
Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes
Date of Performance:
Date of Submission:

Experiment No. 7

Title: Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes

Aim: To study and create GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes

Objective: To introduce GUI, TKinter in python

Theory:

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 with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

Importing the module – tkinter

Create the main window (container)

Add any number of widgets to the main window

Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

Conclusion:

GUI package TKinter has been studied and implemented.

Experiment No. 8
Program to demonstrate CRUD (create, read, update and delete) operations on database (SQLite/ MySQL) using python
Date of Performance:
Date of Submission:

Experiment No. 8

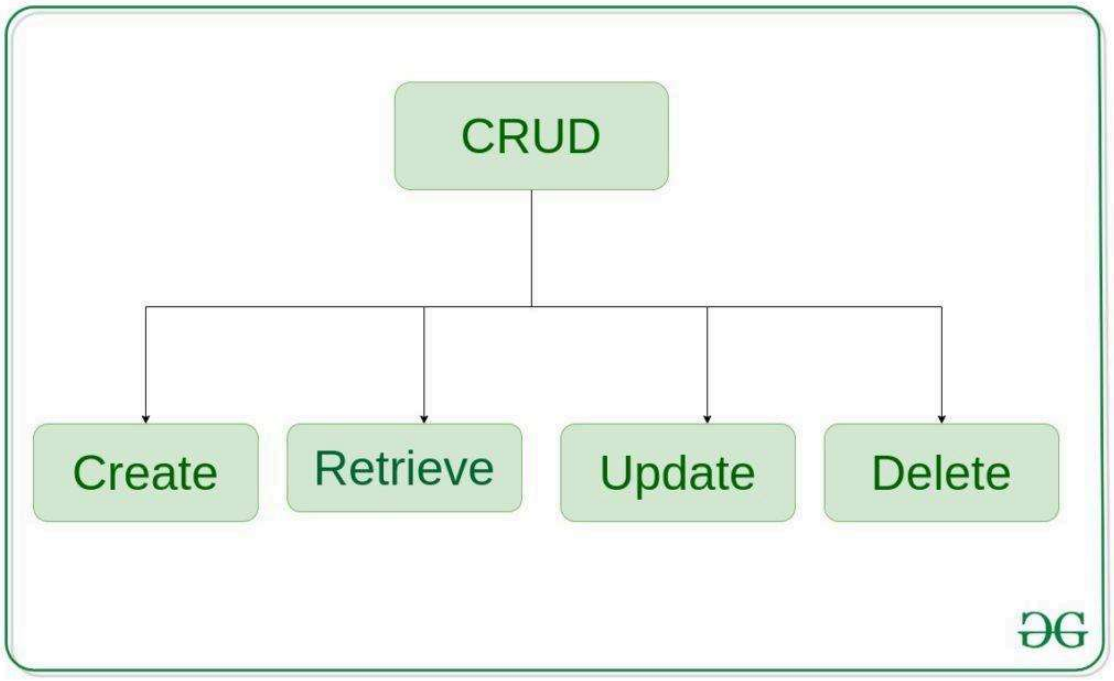
Title: Program to demonstrate CRUD (create, read, update and delete) operations on database (SQLite/ MySQL) using python

Aim: To study and implement CRUD (create, read, update and delete) operations on database (SQLite/ MySQL) using python

Objective: To introduce database connectivity with python

Theory:

In general CRUD means performing Create, Retrieve, Update and Delete operations on a table in a database. Let's discuss what actually CRUD means,



Create – create or add new entries in a table in the database.
Retrieve – read, retrieve, search, or view existing entries as a list(List View) or retrieve a particular entry in detail (Detail View)
Update – update or edit existing entries in a table in the database
Delete – delete, deactivate, or remove existing entries in a table in the database

Conclusion: CRUD operations has been studied and implemented.

Experiment No. 9
Creating web application using Django web framework to demonstrate functionality of user login and registration (also validating user detail using regular expression)
Date of Performance:
Date of Submission:

Experiment No. 9

Title: Creating web application using Django web framework to demonstrate functionality of user login and registration (also validating user detail using regular expression)

Aim: To study and implement web application using Django web framework to demonstrate functionality of user login and registration (also validating user detail using regular expression)

Objective: To introduce Django web framework

Theory:

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web

development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.

Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes.

Exceedingly scalable.

Some of the busiest sites on the web leverage Django's ability to quickly and flexibly scale.

Conclusion: Django web frameworks has been studied and

Experiment No. 10
Demonstrate the concept of Multi-threading
Date of Performance:
Date of Submission:

Experiment No. 10

Title: Demonstrate the concept of Multi-threading

Aim: To study and implement the concept of Multi-threading

Objective: To introduce the concept of Multi-threading in python

Theory:

Thread

In computing, a **process** is an instance of a computer program that is being executed. Any process has 3 basic components:

- An executable program.
- The associated data needed by the program (variables, work space, buffers, etc.)
- The execution context of the program (State of process)

A **thread** is an entity within a process that can be scheduled for execution. Also, it is the smallest unit of processing that can be performed in an OS (Operating System).

In simple words, a **thread** is a sequence of such instructions within a program that can be executed independently of other code. For simplicity, you can assume that a thread is simply a subset of a process!
A thread contains all this information in a **Thread Control Block (TCB)**:

- **Thread Identifier:** Unique id (TID) is assigned to every new thread
- **Stack pointer:** Points to thread's stack in the process. Stack contains the local variables under thread's scope.
- **Program counter:** a register which stores the address of the instruction currently being executed by thread.
- **Thread state:** can be running, ready, waiting, start or done.
- **Thread's register set:** registers assigned to thread for computations.
- **Parent process Pointer:** A pointer to the Process control block (PCB) of the process that the thread lives on.

Conclusion: Multithreading has been successfully implemented in python.

Experiment No. 11
Program to manipulate arrays using NumPy
Date of Performance:
Date of Submission:

Experiment No. 11

Title: Program to manipulate arrays using NumPy

Aim: To study and implement arrays manipulation using NumPy

Objective: To introduce NumPy package

Theory:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

Arrays in Numpy

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array.A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as **ndarray**. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

Creating a Numpy Array

Arrays in Numpy can be created by multiple ways, with various number of Ranks, defining the size of the Array. Arrays can also be created with the use of various data types such as lists, tuples, etc. The type of the resultant array is deduced from the type of the elements in the sequences.

Note: Type of array can be explicitly defined while creating the array.

Conclusion: NumPy package has been studied and arrays have been implemented and manipulated.

Experiment No. 12
Program to demonstrate data frame creation and Manipulation using Pandas
Date of Performance:
Date of Submission:

Experiment No. 12

Title: Program to demonstrate data frame creation and Manipulation using Pandas

Aim: To study and implement data frame creation and Manipulation using Pandas

Objective: To introduce Pandas package for python

Theory:

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

Conclusion: Dataframes have been created and manipulated using Pandas