Namal University, Mianwali
Department of Electrical Engineering

Course Title: EE-253 Data Structures and Algorithms

LAB MANUAL

Lab 3 Python Programming Basics - 3

In previous lab, we have studied the loops (for and while), functions with and without index.

1. Lab Objectives

The objective of this lab is to introduce students to the concept that Python is an object-oriented programming language. In this lab students will enable the concepts of Defining a class, constructing object, accessing members of objects, and self-parameter. Another objective of this lab is to familiarize students to the concept of files in python.
Students will be provided with examples for each objective, followed by performing Lab Tasks and Home Tasks.

2. Lab Outcomes

- CLO:1 The students will enable the concepts of defining a class.
- CLO:1 The students will be able to construct object.
- CLO:1 The students will learn how to access members of objects.
- CLO:1 The students will learn the concept of self-parameter.
- CLO:1 The students will learn about Files in python.
- CLO: 3 A report on the basis of lab tasks and home tasks.

3. Examples

3.1 Defining Classes

A class defines the properties and behavior for objects. Objects are created from classes. A Python class uses variables to store data fields and defines methods to perform actions. A class provides a template which tell what object's data fields and methods will be.

The syntax for a defining a class in Python is as follows:

class ClassName:
initializer
methods

Initializer is used to initialize a new objects state when it is created. A method is designed to perform specific task.

Following example demonstrates the use of class and objects in python.

Example 1: Write a Python class called "Circle" that implements methods to calculate the perimeter and area of a circle, based on a given radius. Also, implement a method to update the radius value. Finally, demonstrate the use of the class by creating an object with a radius of 5 and printing its properties (radius, perimeter, and area).

Source code:

```
import math

class Circle:

    def __init__(self,radius=1):
        self.radius=radius
    def getPerimeter(self):
        return 2*self.radius*math.pi
    def getArea(self):
        return self.radius*self.radius*math.pi
    def setRadius(self,radius):
        self.radius=radius

c=Circle(5)
print("The value of radius is",c.radius)
perimeter = c.getPerimeter()
```

```
print("The perimeter of circle is is",perimeter)
area=c.getArea()
print("The area of circle is is",area)
```

Output:

```
The value of radius is 5
The perimeter of circle is 31.41592653589793
The area of circle is 78.53981633974483
```

Explanation of code:

This code defines a class called Circle that represents a circle object. The import math statement at the beginning of the code imports the Python math library, which contains mathematical functions like pi. Then a constructor method for the Circle class is created. It is called when a new circle object is created. The self-parameter refers to the object being created. The radius parameter is optional and has a default value of 1 if it's not provided. The constructor initializes the radius attribute of the object to the provided radius or the default value. Next a method of Circle class is created that calculates and returns the perimeter of the circle object. The self-parameter refers to the circle object that the method is being called on. The method uses the radius attribute of the object and the pi constant from the math library to calculate the perimeter. Another method of the Circle class that calculates and returns the area of the circle object is created. The self-parameter refers to the circle object that the method is being called on. The method uses the radius attribute of the object and the pi constant from the math library to calculate the area. Another method Circle class that allows you to change the radius attribute of a circle object after it has been created. The self-parameter refers to the circle object that the method is being called on. The radius parameter is the new value for the radius attribute. After creating the class, a new circle object c with a radius of 5 has been created. The constructor method is called with an argument of 5, so the radius attribute of the c object is set to 5. After wards this line gets print. Then the perimeter of the c circle object using the getPerimeter() method has been calculated and printed. The perimeter variable is assigned the result of the method call, and then the perimeter value is printed. The area of the c circle object using the getArea() method has been calculated and printed. The area variable is assigned the result of the method call, and then the area value is printed.

Task 1: Write a Python program that creates a Student class with an __init__ method that takes a name and marks argument and initializes corresponding instance variables.
The average method should calculate and return the average of the marks list.
The program should prompt the user to enter the name of a student and their marks for three subjects.
It should then create a Student object with the given name and marks, and print the student's name and their average marks.

Task 2: Write a class called Person which has a constructor to assign initial values to name and age. Then call a method to display person's name and age.

3.2 Files in Python

Files are typically accessed in Python beginning with a call to a built-in function, named open, that returns a proxy for interactions with the underlying file. For example, the command, *fp = open(sample.txt)*, attempts to open a file named sample.txt, returning a proxy that allows read-only access to the text file. The open function accepts an optional second parameter that determines the access mode. The default mode is *r* for reading. Other common modes are *w* for writing to the file (causing any existing file with that name to be overwritten), or a for appending to the end of an existing file. Although we focus on use of text files, it is possible to work with binary files, using access modes such as *rb* or *wb*.

Example 2: Initially you have a file file_in_class.txt that contains initial content as "DSA Lab". Then you are writing "Timing 1:00 to 2:15 pm" into the file. It will overwrite the file. Then a message "'Now it is appending after previous line" is appended to the file.

Source Code:

```
fp = open('file_in_class.txt')
print(fp.read())

fp = open('file_in_class.txt', 'w')
fp.write('Timing 1:00 to 2:15 pm')
fp.close()
print("Message after write")
fp = open('file_in_class.txt', 'r')
print(fp.read())
fp.close()


fp = open('file_in_class.txt', 'a')
fp.write('Now it is appending after previous line')
fp.close()
print("Message after append")
fp = open('file_in_class.txt', 'r')
print(fp.read())
fp.close()
```

Output:
DSA Lab
Message after write
Timing 1:00 to 2:15 pm
Message after append
Timing 1:00 to 2:15 pm Now it is appending after previous line

Content of file:
Timing 1:00 to 2:15 pm Now it is appending after previous line.

Task 3 Create a Python code that reads a file containing some initial text. Then, write to the file and append additional text to it. Observe the changes that occur in the file.

Lab Evaluation Rubrics							
Domain	**Rubric**	**Performance Indicator**	**Unsatisfactory 0-5**	**Marginal 5-10**	**Satisfactoy 11-15**	**Exemplary 16-20**	**Allocated Marks**
Psychomotor	**R1**	Softwre Setup **(P)**	Does not know how to use Software tool relevant to the lab.	Is able to use some of the functions of Software tool required in an experiment with errors and repeated guidance.	Is able to use all the functions of Software tool required in an experiment with errors and repeated guidance.	Able to use all the functions of Software tool required in an experiment with little to no guidanceand without any error.	10
	R2	Implementation with Results **(P)**	Does not try to solve problems. Many mistakes in code and difficult to comprehend for the instructor. There is not result of the problem.	Does not suggests or refine solutions but is willing to try out solutions suggested by others. Few mistakes in code, but done along with comments, and easy to comprehend for the instructor. Few mistake in result.	Refines solutions suggested by others. Complete and error-free code is done. No comments in the code, but easy to comprehend for the instructor. Results are correctly produced.	Actively looks for and suggests solution to problems. Complete and error free code is done, easy to comprehend for the instructor. Results are correctly produced. Student incorporated comments in the code.	30
Affective	**R3**	Lab Report **(A)**	Code of the problem is not given. Outputs are not provided. Explanation of the solution is not stated.	Code of the problem is not given. Output is not complete. Explanation of the solution is not satisfactory.	Code of the problem is not given. Output is completely given. Explanation of the solution is not satisfactory.	Code of the problem is not given. Output is completely given. Explanation of the solution is satisfactory.	20
	R5	Discipline and Behavior **(A)**	Got and wandered around. More than two incidents of talking non-lab related stuff in laband/or any talk with other groups, voice level exceeding the appropriate level, use of cell phones and involvement in any non lab activity.	Got out of seat and wander around for some time. No more than two incidents of talking non-lab related stuff in lab. Voice level exceeding the appropriate level, use of cell phones and involvement in any non-lab related activity.	Stayed in seat and got up for a specific lab related reason, but took more time than required to do the job. No more than one incidents of talking non-lab related stuff in lab. Voice level exceedingthe appropriate level, use of cell phones and involvementin any non-lab related activity.	Stayed in seat and got up for a specific lab related reason. Tookcare of lab related business and sat down right away. Voice level kept appropriate. Not used cell phones or involved in any non- lab related activity.	10