UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR, JALOZAI CAMPUS

Lab 4: Classes, objects, and instances

Lab Title: EE-271 "OOP & Data Structures Lab"

Time: 10 min/ Task

Lab report task:

1. Consider the point class below.
   a. Make an object and print its x and y coordinates.

```
import math
class Point:
    """Represents a point in two-dimensional geometric
coordinates
        Parameters
          ----------
          x : float
          y : float
          """
    def __init__(self, x: float, y: float)->None:
        self.x = x
        self.y = y
    def distance(self, p2)->float:
        return math.sqrt((self.x-p2.x)**2 + (self.y-p2.y)**2)
```

   a. Define point1 and pass two numbers.
   b. Make another instance of the point class, say its name is p2.
   c. Print p1 and p2, for this use the print command and pass the point as input.
   d. Print the coordinate by using the object and dot operator.
   e. Add a new method to the point class that can effectively print the points. Print both the point using that function.
   f. Calculate the distance between these two pints.

Lab work tasks:

1. Motivation

   Interest is to make a list that can store employee data.

   `[name, age, position, the year they started working]`

   For example

```
kirk = ["James Kirk", 34, "Captain", 2265]
spock = ["Spock", 35, "Science Officer", 2254]
mccoy = ["Leonard McCoy", "Chief Medical Officer", 2266]
```

   a. `Print(kirk[0])`

Hints: First, when you reference kirk[0] several lines away from where the kirk list is declared, will you remember that the 0th element of the list is the employee's name?

b. `Print(mccoy[1])`

Hints: What if not every employee has the same number of elements in the list? In the mccoy list above, the age is missing, so mccoy[1] will return "Chief Medical Officer" instead of Dr. McCoy's age.

A great way to make this type of code more manageable and more maintainable is to use **classes**.

In the following section a brief introduction to classes, how to make instances/ objects of those classes, and how to use them.

2. Dog class

```
# dog.py


class Dog:
    pass
```

Creating a new object from a class is called **instantiating** a class. You can create a new object by typing the name of the class, followed by opening and closing parentheses:

a. `Dog()`

This can be assigned to a variable.

b. `Inst=Dog()`

c. Create two new Dog objects and assign them to the variables a and b.

```
a = Dog()
b = Dog()
```

d. When you compare a and b using the == operator, the result is False. Even though a and b are both instances of the Dog class, they represent two distinct objects in memory.

```
a == b
```

e. Use the print command and print both instances/ objects.

3. Following is a class named MyFirstClass.

```
class MyFirstClass:
    pass
```

a. Make two instances named FC1 and FC2.

b. Check whether FC1 and FC2 represent the same object or not.

c. Use the print command and print both instances.

Note:

- The pass keyword on the second line indicates that no further action needs to be taken.

- When printed, the two objects tell us which class they are and what memory address they live at. Memory addresses aren't used much in Python code, but here, they demonstrate that there are two distinct objects involved.

4. Consider the following class definition.

```
class Point:
    pass
```

   a. Make two instances named p1 and p2.

   b. Print both objects p1 and p2.

   c. Try different names for objects, instead of p1 and p2. Also, use different attribute names.

   Note:

   - The value assigned to attribute can be anything: a Python primitive, a built-in data type, or another object. It can even be a function or another class!

5. Update the Dog class with an __init__() method that creates .name and .age attributes:

```
# dog.py

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

   a. Run: `miles = Dog("Miles", 4)`

   b. Print `miles`

6. Create a new Dog class with a class attribute called. species and two instance attributes called .name and .age:

   Class attribute: You define class attributes directly beneath the first line of the class name and indent them by four spaces.

```
# dog.p

class Dog:
    species = "Canis familiaris"

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

   a. To instantiate this Dog class, you need to provide values for name and age. If you don't, then Python raises a TypeError:

- Run: `Dog()`

b. To pass arguments to the name and age parameters, put values into the parentheses after the class name:

```
miles = Dog("Miles", 4)
buddy = Dog("Buddy", 9)
```

c. After you create the Dog instances, you can access their instance attributes using **dot notation**:

```
miles.name
miles.age

buddy.name
buddy.age
```

d. Although the attributes are guaranteed to exist, their values *can* change dynamically:

```
buddy.age = 10
buddy.age

miles.species = "Felis silvestris"
miles.species
```

7. Define a class point in 2 dimension coordinate system. Your class must have the appropriate __init__ method. In addition, add two other instance methods for the distance between points and also show or display the point properly (note: return tuple like (x, y)). Add another method with the name locate to display the coordinate in which the point is located. Add proper annotation and doc string to every class and instance method.
    a. Define point1 and pass two numbers.
    b. Make another instance of the point class, say its name is p2.
    c. Print p1 and p2, for this use the print command and pass the point as input.
    d. Also print the location of both points.
    e. Print the coordinate by using the object and dot operator.
    f. Add a new method to the point class to effectively print the points. Print both the point using that function.
    g. Calculate the distance between these two pints.
    h. Call the __dict__ by the class name.
    i. Call the __dict__ on the object of the class.
    j. Pass the class name to help.
    k. Print the doc-string and annotations of both the class and each instance method.

8. Modify the above class by making its parameters default. The parameters will initialize the point with no parameters to the origin.
    a. Define point_1 by passing two arguments and print their values by using the dot operator and also by calling the show function.
    b. Define another object with the name point_2 and pass one argument. Display its value by calling the show method.
    c. Define another object with the name point_3 and pass no argument. Display its value by calling the show method.