Batch: A3   **Roll No.:** 16010121051
Experiment / assignment / tutorial No. 6
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

TITLE: Class, Object , Types of methods and Constructor

AIM: Write a program to create StudentInfo class .Calculate the percentage scored by the student

_____

Expected OUTCOME of Experiment: Apply Object oriented programming concepts in Python

_____

**Resource Needed: Python IDE**

_____

Theory:

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods .A Class is like an object constructor, or a "blueprint" for creating objects. Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Example :
```
class MyClass:
    variable = "hello"
    def function(self):
        print("This is a message inside the class.")
myobjectx = MyClass()
```

The self-parameter is a reference to the current instance of the class, and is used to access variables that belong to the class. It does not have to be named self you can call it whatever you like, but it has to be the first parameter of any function in the class.
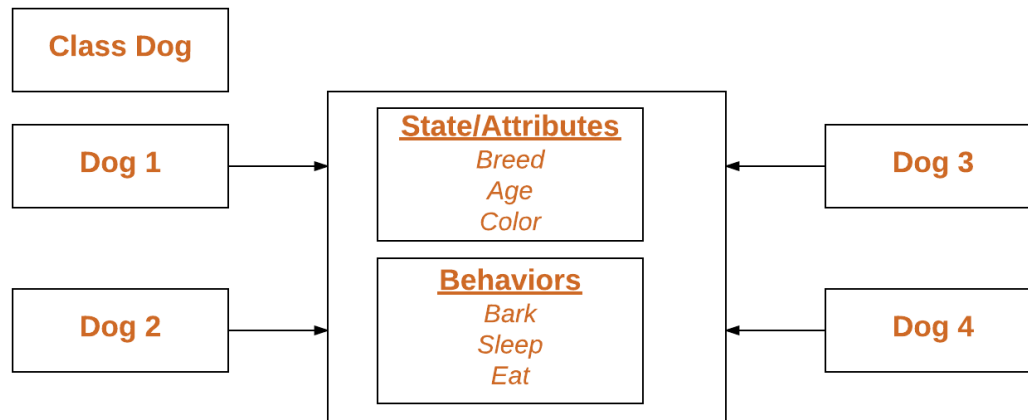
Public Members of a class (data and methods) are accessible from outside the class. Private members are inaccessible from outside the class. Private members by convention start with an underscore, as _name, _age, _salary.

There are three types of methods in Python: instance methods, static methods, and class methods.

Instance methods:

Instance methods are the most common type of methods in Python classes. These are so called because they can access unique data of their instance. Instance methods must have self as a parameter. Inside any instance method, you can use self to access any data or methods that may reside in your class. You won't be able to access them without going through self.

Static methods:

 Static methods are methods that are related to a class in some way, but don't need to access any class-specific data. You don't have to use self, and you don't even need to instantiate an instance

Class methods: They can't access specific instance data, but they can call other static methods. Class methods don't need self as an argument, but they do need a parameter called cls. This stands for class, and like self, gets automatically passed in by Python. Class methods are created using the @classmethod decorator.

Example:

```
class MyClass:
    def method(self):
        return 'instance method called', self

    @classmethod
    def classmethod(cls):
        return 'class method called', cls
```

```
@staticmethod
def staticmethod():
    return 'static method called
```

Constructors in Python

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 class is created. In Python the __init__() method is called the constructor and is always called when an object is created.
Syntax of constructor declaration:

```
def __init__(self):
    # body of the constructor
```

Types of constructors:

• **Default constructor:** The default constructor is simple constructor which doesn't accept any arguments. It's definition has only one argument which is a reference to the instance being constructed.
• **Parameterized constructor**: constructor with parameters is known as parameterized constructor. The parameterized constructor take its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

Python built-in function

The built-in functions defined in the class are described in the following table.

SN	Function	Description
1	getattr(obj,name,default)	It is used to access the attribute of the object.
2	setattr(obj, name,value)	It is used to set a particular value to the specific attribute of an object.
3	delattr(obj, name)	It is used to delete a specific attribute.

| 4 | hasattr(obj, name) | It returns true if the object contains some specific attribute. |

Problem Definition:

1. For given program find output

Sr.No	Program	Output
1	class MyClass: x = 5 p1 = MyClass() print(p1.x)	5

2	class Person:
	  def __init__(self, name, age):
	    self.name = name
	    self.age = age
	
	p1 = Person("John", 36)
	
	print(p1.name)
	print(p1.age)
		John
		
		36

3	class Student:
	  # Constructor - non parameterized
	  def __init__(self):
	    print("This is non parametrized constructor")
	  def show(self,name):
	    print("Hello",name)
	student = Student()
	student.show("John")
		This is non parametrized constructor
		
		Hello John

4	class Student:
	  roll_num = 101
	  name = "Joseph"
	
	  def display(self):
	    print(self.roll_num,self.name)
	
	st = Student()
	st.display()
		101 Joseph

5	class Student:
	  # Constructor - parameterized
	  def __init__(self, name):
		This is parametrized constructor

	print("This is parametrized constructor")
	        self.name = name
	    def show(self):
	        print("Hello",self.name)
	student = Student("John")
	student.show()
		Hello John

2. Write a program to accept Roll Number, Marks Obtained in four subjects, calculate total Marks and percentage scored by the student. Display the roll number, marks obtained, total marks and the percentage scored by the student. Use getter-setter methods.

Implementation details:

```
class student:
    def __init__(name,self,roll,member1,member2,member3,member4):
        self.roll = roll
        self.name = name
        self.m2 = member2
        self.m1 = member1
        self.m3 = member3
        self.m4 = member4
name = input("Enter your full name >>")
roll = int(input("Enter the roll number of the student >>"))
member1 = int(input("Enter marks in subject No.1 >>"))
member2 = int(input("Enter marks in subject No.2 >>"))
member3 = int(input("Enter marks in subject No.3 >> "))
member4 = int(input("Enter marks in subject No.4 >> "))

stu1 = student
setattr(stu1,'name',name)
setattr(stu1,'roll',roll)
setattr(stu1,'m1',member1)
setattr(stu1,'m2',member2)
setattr(stu1,'m3',member3)
setattr(stu1,'m4',member4)
```

```python
print("MARK LIST")
print("Name :",getattr(stu1,'name'))
print("Roll Number : ",getattr(stu1,'roll'))
print("Marks in Subject1 : ",getattr(stu1,'m1'))
print("Marks in Subject2 : ",getattr(stu1,'m2'))
print("Marks in Subject3 : ",getattr(stu1,'m3'))
print("Marks in Subject4 : ",getattr(stu1,'m4'))

totalMarks = 0

for i in range(1,5):
    totalMarks = totalMarks + getattr(stu1,'m'+str(i))
    i = i+1

print("Total marks scored by the student : ",totalMarks)

percentage = totalMarks*0.25

print("Percentage scored by the student : ",percentage,"%")
```

Output(s):

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\PythonProgramming> & "C:/Program Files/Python310/python.exe" c:/PythonProgramming/exp6a.py
Enter your full name >>Meet gala
Enter the roll number of the student >>16010121051
Enter marks in subject No.1 >>77
Enter marks in subject No.2 >>90
Enter marks in subject No.3 >> 67
Enter marks in subject No.4 >> 85
MARK LIST
Name : Meet gala
Roll Number :  16010121051
Marks in Subject1 :  77
Marks in Subject2 :  90
Marks in Subject3 :  67
Marks in Subject4 :  85
Total marks scored by the student :  319
Percentage scored by the student :  79.75 %
PS C:\PythonProgramming>
```

Books/ Journals/ Websites referred:

1. Reema Thareja, *Python Programming: Using Problem Solving Approach*, Oxford University Press, First Edition 2017, India
2. Sheetal Taneja and Naveen Kumar, *Python Programming: A modular Approach*, Pearson India, Second Edition 2018,India

Conclusion: Through this experiment, we understand the use of classes, objects and methods to make user defined datatypes to create a blueprint for the objects.

Post Lab Questions:

1. Write a program that has a class 'store' which keeps a record of code and price of each product. Display a menu of all products to the user and prompt them to enter the quantity of each item required. Generate a bill and display the total amount.

   Implementation details:

```python
class store:
    def __init__(self,code,price):
        self.code = code
        self.price = price


st1 = []

st1.append(store("AB12345",195))
st1.append(store("AB14567",350))
st1.append(store("AB09375",100))
st1.append(store("AB98734",250))
st1.append(store("AB09998",185))
st1.append(store("AB45346",190))


print("Product and Price list")
```

```python
for i in range(len(st1)):
    print(st1[i].code,"   ",st1[i].price)

dict1 = {}

while True:
    pq = input("Enter the product code : ")
    qty = int(input("Enter the quantity : "))
    dict1[pq] = qty

    x = input("Do you want to continue[yes/no] : ")
    if x == "no" or x == "No":
        break
    else:
        continue

print("\nProduct and Quantity purchased")
print(dict1)

totalPrice = 0

for x in dict1.keys():
    for i in range(len(st1)):
        if x == st1[i].code:
            totalPrice = totalPrice + (dict1[x]*st1[i].price)

print("Total cost = ",totalPrice)
```

Output(s):

```
Install the latest PowerShell for new features an

PS C:\PythonProgramming> & "C:/Program Files/Pyth
Product and Price list
AB12345    195
AB14567    350
AB09375    100
AB98734    250
AB09998    185
AB45346    190
Enter the product code : AB12345
Enter the quantity : 4
Do you want to continue[yes/no] : yes
Enter the product code : AB12345
Enter the quantity : 4
Do you want to continue[yes/no] : yes
Enter the product code : AB45346
Enter the quantity : 2
Do you want to continue[yes/no] : no

Product and Quantity purchased
{'AB12345': 4, 'AB45346': 2}
Total cost =  1160
PS C:\PythonProgramming>
```

2. What is the use of getter and setter methods?

Getters and Setters are important in retrieving and updating the value of a variable outside the encapsulating class. They are used to protect our data, particularly when we are creating classes. For each instance variable, a getter method returns its value while a setter method sets or updates its value. They make it easy to debug the code and verify in the case of errors.

Date: _____   **Signature of faculty in-charge**