Module 9: OOP

Class:

- Class is a blueprint for the object.
- It encapsulates data(variables) & methods (functions) together.

Object

- Object is called the instance of class.

Encapsulation & Instantiation:

- Process of creating class is called encapsulation and process of creating object is called instantiation.

Initializer:

- Initializer is a special method that is used to initialize the instance variables of a class ➔ __init__()
- We create instance variables and initialize them in initializer.
- The first parameter is self which contains the memory address of the instance.
- Class can either have: Default Initializer OR Parameterized Initializer

---

Inheritance:

- Inheritance is the mechanism of deriving one class from another such that the new derived class can inherit all the members of the existing base class.
- The syntax of inheritance is:
  - class SubClass(SuperClass):
- Advantage of inheritance is reusability and hence more code less time.

Types of Inheritance:

- Python supports following type on inheritance :
  - Single
  - Multilevel
  - Hierarchical
  - Hybrid
  - Multiple

Polymorphism:

- If a variable, object or method exhibits different behavior in different contexts then it is called polymorphism.

Method Overriding:

- In Method overriding we override the method of super class in the subclass.
- We do this coz we want to change the functionality for the same method in the subclass.

Operator Overloading:

- + operator internally calls __add__() method.
- By overriding this method we can make + operator to work with user defined objects also.

+	object.__add__(self, other)
-	object.__sub__(self, other)
*	object.__mul__(self, other)
//	object.__floordiv__(self, other)
/	object.__div__(self, other)
%	object.__mod__(self, other)
**	object.__pow__(self, other[, modulo])
<<	object.__lshift__(self, other)
>>	object.__rshift__(self, other)
&	object.__and__(self, other)
^	object.__xor__(self, other)
|	object.__or__(self, other)

<	object.__lt__(self, other)
<=	object.__le__(self, other)
==	object.__eq__(self, other)
!=	object.__ne__(self, other)
>=	object.__ge__(self, other)
>	object.__gt__(self, other)

Module 9: OOP - Test

Q1) Which of the following keyword mark the beginning of the class definition?

Options:

a) def
b) define
c) class
d) CLASS

Solution:

Q2) What is the output of the following code:

```
class Animal:
    pass
a = new Animal()
print(a)
```

Options:

a) Error
b) <__main__.Animal object at 0x031A03F0>
c) <__main__.Animal object>
d) <0x031A03F0>

Solution:

Q3) What is the output of the following code:

```
class Employee:
    def __init__(self, id, name):
        self.id = id
        self.name = name
e1 = Employee(10)
print(e1.id , e1.name)
```

Options:

a) 10
b) 10, None
c) 10, null
d) Error

Solution:

Q4) What is the result of the following?

```
class Employee:
    def comm(self):
        return 10
class SalesPerson(Employee):
    def comm(self):
        return 20
s = SalesPerson()
print(s.comm())
```

Options:

a) 10
b) 20
c) Error
d) None

Solution:

Q5) What is the result of the following?

```
class Book:
    def __init__(self, pages=10):
        pass
    def __add__(self, other):
        a = self.pages + other.pages
        return a

b1 = Book()
b2 = Book()
r1 = b2 + b1
print(r1)
```

Options:

a) 10
b) 20
c) Error
d) None

Solution:
