5th Feb Assignment

February 10, 2023

1 Assignment 6

Q1. Explain Class and Object with respect to Object-Oriented Programming. Give a suitable example.

Ans.Class and Object are fundamental concepts in Object-Oriented Programming (OOP).

A Class is a blueprint or template that defines the properties and behaviors of a certain type of objects. In other words, a class defines the structure and functionality of a group of similar objects.

An Object is an instance of a class, created at runtime. An object represents a specific instance of a class and has its own unique state and behavior.

[2]:
```
#example
#creating test class
class test:
    def welcome_message(self):
        print('Welcome to pwskills')
```

[3]:
```
#object of class test
jp=test()
```

[4]:
```
jp.welcome_message()
```

```
Welcome to pwskills
```

Q2. Name the four pillars of OOPs.

Ans.The four pillars of OOPs are:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Q3. Explain why the init() function is used. Give a suitable example.

Ans.The init method, also known as the constructor, is a special method in Python that is automatically called when an object is created from a class. It is something which help my class to take data while creating an object.

[5]:
```
#example
class test1:
    def __init__(self,name,age,phone_number):
        self.name=name
        self.age=age
        self.phone_number=phone_number
    def student_details(self):
        return self.name,self.age,self.phone_number
```

[6]:
```
jp=test1('Jp',20,7877777777777775)
```

[7]:
```
jp.name
```

[7]:
```
'Jp'
```

[8]:
```
jp.age
```

[8]:
```
20
```

[9]:
```
jp.phone_number
```

[9]:
```
7877777777777775
```

[10]:
```
jp.student_details()
```

[10]:
```
('Jp', 20, 7877777777777775)
```

Q4. Why self is used in OOPs?

Ans.The self parameter is a reference to the instance of the object on which a method is being called. It is used to access the attributes and methods of the object from within the class.

Q5. What is inheritance? Give an example for each type of inheritance.

Ans. Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called "derived class" or "child class" and the existing class is known as the "base class" or "parent class". The derived class now is said to be inherited from the base class.

There are 4 types of inheritance-

- Single inheritance
- Multiple Inheritance

- Multilevel Inheritance
- Hierarchial Inheritance

1.0.1 Single Inheritance

[19]:
```
class parent:
    def welcomeP(self):
        print('This is a base class')

class child(parent):
    def welcomeB(self):
        print('This is a child class')
```

[20]:
```
obj_base=child() ##base class object
```

[21]:
```
obj_base.welcomeB()
```

```
This is a child class
```

[22]:
```
obj_base.welcomeP()
```

```
This is a base class
```

1.0.2 Multilevel Inheritance

[14]:
```
class classP:
    def testP(self):
        print('This is parent class')

class class1(classP):
    def test1(self):
        print('This is child of parent class')

class class2(class1):
    def test2(self):
        print('This is child of class1')
```

[15]:
```
#object of class2
obj_class2=class2()
```

[16]:
```
obj_class2.test2()
```

```
This is child of class1
```

[17]:
```
obj_class2.test1()
```

```
This is child of parent class
```

[18]:
```
obj_class2.testP()
```

```
This is parent class
```

1.0.3 Multiple Inheritance

[23]:
```
class parent1:
    def test3(self):
        print('This is parent1 class')

class parent2:
    def test4(self):
        print('This is parent2 class')

class child1(parent1,parent2):
    def test5(self):
        print('This is child1 class')
```

[24]:
```
#object of child1 class
obj1=child1()
```

[25]:
```
obj1.test5()
```

```
This is child1 class
```

[26]:
```
obj1.test4()
```

```
This is parent2 class
```

[27]:
```
obj1.test3()
```

```
This is parent1 class
```

1.0.4 Hierarchial Inheritance

[29]:
```
class Parent:
    def func1(self):
        print("This function is in parent class.")

class Child1(Parent):
    def func2(self):
        print("This function is in child 1.")

class Child2(Parent):
    def func3(self):
        print("This function is in child 2.")
```

[33]:
```
#objects of both child class
object1=Child1()
object2=Child2()
```

[34]:
```
object1.func1()
object1.func2()
```

```
This function is in parent class.
This function is in child 1.
```

[35]:
```
object2.func1()
object2.func3()
```

```
This function is in parent class.
This function is in child 2.
```