Untitled9

February 7, 2023

0.1 Assignment - 6

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

0.1.2 Ans - Class is like blueprint of peroperties (Variables) and methods. object is the instance of class.

0.1.3 Example - Class - Empty id card form. object is form filled by each and every student. here properties are name, age, address and mobile number. which vary based on object but blueprint will be same for everyone.

0.1.4 Q2. Name the four pillars of OOPs.

0.1.5 Ans - Inheritance, Polymorphism, Encapsulation and Abstraction

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

0.1.7 Ans - The init method lets the class initialize the object's attributes and it will be specific to object it is beeing initiated with.

0.1.8 Example -

[14]:
```python
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
```

0.1.9 Q4. Why self is used in OOPs?

0.1.10 Ans - The self keyword is used to represent an instance (object) of the given class.

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

0.1.12 Ans - Inheritance allows us to define a class that inherits all the methods and properties from another class.

0.2 Single Inheritance:

0.2.1 Single inheritance enables a derived class to inherit properties from a single parent class, thus enabling code reusability and the addition of new features to existing code.

0.2.2 Example

[11]:
```python
# Python program to demonstrate
# single inheritance

# Base class
class Parent:
        def func1(self):
                print("This function is in parent class.")

# Derived class


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


# Driver's code
object = Child()
object.func1()
object.func2()
```

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

0.2.3 Multiple Inheritance:

0.2.4 When a class can be derived from more than one base class this type of inheritance is called multiple inheritances. In multiple inheritances, all the features of the base classes are inherited into the derived class.

[15]:
```python
# Python program to demonstrate
# multiple inheritance

# Base class1
class Mother:
        mothername = ""

        def mother(self):
                print(self.mothername)

# Base class2


class Father:
        fathername = ""

        def father(self):
                print(self.fathername)

# Derived class


class Son(Mother, Father):
        def parents(self):
                print("Father :", self.fathername)
                print("Mother :", self.mothername)


# Driver's code
s1 = Son()
s1.fathername = "RAM"
s1.mothername = "SITA"
s1.parents()
```

```
Father : RAM
Mother : SITA
```

0.2.5 Multilevel Inheritance :

0.2.6 In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class. This is similar to a relationship representing a child and a grandfather.

[16]:
```
# Python program to demonstrate
# multilevel inheritance

# Base class


class Grandfather:

        def __init__(self, grandfathername):
                self.grandfathername = grandfathername

# Intermediate class


class Father(Grandfather):
        def __init__(self, fathername, grandfathername):
                self.fathername = fathername

                # invoking constructor of Grandfather class
                Grandfather.__init__(self, grandfathername)

# Derived class


class Son(Father):
        def __init__(self, sonname, fathername, grandfathername):
                self.sonname = sonname

                # invoking constructor of Father class
                Father.__init__(self, fathername, grandfathername)

        def print_name(self):
                print('Grandfather name :', self.grandfathername)
                print("Father name :", self.fathername)
                print("Son name :", self.sonname)


# Driver code
s1 = Son('Prince', 'Rampal', 'Lal mani')
print(s1.grandfathername)
s1.print_name()
```

```
Lal mani
Grandfather name : Lal mani
Father name : Rampal
Son name : Prince
```

0.2.7 Hierarchical Inheritance:

0.2.8 When more than one derived class are created from a single base this type of inheritance is called hierarchical inheritance. In this program, we have a parent (base) class and two child (derived) classes.

[17]:
```python
# Python program to demonstrate
# Hierarchical inheritance


# Base class
class Parent:
        def func1(self):
                print("This function is in parent class.")

# Derived class1


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

# Derivied class2


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


# Driver's code
object1 = Child1()
object2 = Child2()
object1.func1()
object1.func2()
object2.func1()
object2.func3()
```

```
This function is in parent class.
This function is in child 1.
This function is in parent class.
This function is in child 2.
```

0.2.9 Hybrid Inheritance:

0.2.10 Inheritance consisting of multiple types of inheritance is called hybrid inheritance.

[19]:
```python
# Python program to demonstrate
# hybrid inheritance


class School:
        def func1(self):
                print("This function is in school.")


class Student1(School):
        def func2(self):
                print("This function is in student 1. ")


class Student2(School):
        def func3(self):
                print("This function is in student 2.")


class Student3(Student1, School):
        def func4(self):
                print("This function is in student 3.")


# Driver's code
object = Student3()
object.func1()
object.func2()
```

```
This function is in school.
This function is in student 1.
```

[ ]:
```
```