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

Solutions : A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

To understand the need for creating a class in Python let's consider an example, let's say you wanted to track the number of dogs that may have different attributes like breed, and age. If a list is used, the first element could be the dog's breed while the second element could represent its age. Let's suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it's the exact need for classes.

Q2. Name the four pillars of OOPs.

Solution: 1. Inheritance

2.polymorphism

3. Encapsulation

4.Abstraction

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

Solution: this is a type of constructor .Constructors are used to initializing the object's state. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation. It is run as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.

Example:

```
Class pwskills :
  Def __init__(self, name):
    self.name=name
```

Def greeting (self

print(“hello bachoooooooooooo”,self.name)

jk=pwskills(‘jyotish’)

jk.greeting()

4. Why self is used in OOPs?

Solution : The self variable is used to represent the instance of the class which is often used in object-oriented programming. It works as a reference to the object. Python uses the self parameter to refer to instance attributes and methods of the class. Self is not a reserved keyword we can also use any variable .

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

Solution : Inheritance is a mechanism of acquiring the features and behaviors of a class by another class. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class. Inheritance implements the IS-A relationship.

For example, mammal IS-A animal, dog IS-A mammal; Hence dog IS-A animal as well.

OOPs support the six different types of inheritance as given below :

- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Multipath inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

1. Single inheritance

In this inheritance, a derived class is created from a single base class.

example, Class A is the parent class and Class B is the child class since Class B inherits the features and behavior of the parent class A.

Class A —--> class B

2. Multi-level inheritance

In this inheritance, a derived class is created from another derived class.

In the given example, class c inherits the properties and behavior of class B and class B inherits the properties and behavior of class B. So, here A is the parent class of B and class B is the parent class of C. So, here class C implicitly inherits the properties and behavior of class A along with Class B i.e there is a multilevel of inheritance

Class A —--> class B—--->class c

3. Multiple inheritance

In this inheritance, a derived class is created from more than one base class.

In the given example, class c inherits the properties and behavior of class B and class A at the same level. So, here A and Class B both are the parent classes for Class C.

Class A —>

Class B—> class C

4.Multipath inheritance

In this inheritance, a derived class is created from other derived classes and the same base class of other derived classes.

In the given example, class D inherits the properties and behavior of class C and class B as well as Class A. Both class C and class B inherit the Class A. So, Class A is the parent for Class B and Class C as well as Class D. So it's making it a Multipath inheritance.

5.Hierarchical Inheritance

In this inheritance, more than one derived class is created from a single base class and further child classes act as parent classes for more than one child class.

In the given example, class A has two children class B and class D. Further, class B and class C both are having two children - class D and E; class F and G respectively.

6.Hybrid inheritance

This is a combination of more than one inheritance. Hence, it may be a combination of Multilevel and Multiple inheritance or Hierarchical and Multilevel inheritance Hierarchical and Multipath inheritance, or Hierarchical, Multilevel and Multiple inheritances.