

Q1. What is the purpose of Python's OOP?

Ans: Object Oriented Programming (OOP) in Python is aimed at creating modular code which can be reusable and also protected by using classes and objects. The main advantages of OOP are -

1. Abstraction - By employing classes and objects, the program can be made more simple and easier to read. Moreover, certain functionalities/blocks of code can be hidden from other users if needed.
2. Inheritance - Due to this, desired attributes and methods can be used in a child class from an already existing class. This makes the code more reusable.
3. Polymorphism - The same method can be used on different objects to get different results. In the case of inheritance in child classes, the inherited methods can be overridden to meet the requirements. In this way, the code becomes robust and can be easily modified as needed.
4. Encapsulation - It can be used to protect certain attributes or methods for security reasons or to prevent any accidental modifications. This is achieved using private and protected variables.

Q2. Where does an inheritance search look for an attribute?

Ans: An inheritance search looks for attributes from the parent class. In case of multiple inheritance, it looks for attributes from the parent class mentioned first.

Q3. How do you distinguish between a class object and an instance object?

Ans: A class object is one where the attributes and methods of a class are class variables and class methods. That is, they belong to the class as a whole and are not defined by using the pointer (self).

An instance object has attributes and methods that are instantiated/created specifically using the constructor (`__init__()`) and pointer (self). Thus, such attributes and methods are associated only with that instance object and not other instances.

Q4. What makes the first argument in a class's method function special?

Ans: The very first argument (conventionally specified as 'self') is used to represent the instance of a class. It essentially helps the object to access its attributes and methods.

Q5. What is the purpose of the `__init__` method?

Ans: The `__init__` method is used to set the initial state of the object when it is created by assigning the attributes to the object.

Q6. What is the process for creating a class instance?

Ans: A class instance is created by declaring a variable as a function of the class. For example, suppose there is a class named 'student'. Then the instance of that class can be written as -

```
a = student()
```

Q7. What is the process for creating a class?

Ans: A class is created by using the keyword *class* followed by the name of the class. There may be attributes assigned to the class followed by methods (although this is not mandatory). For example,

```
class car:
    def __init__(self, a):
        self.a = a
    def method(self):
        print("...")
```

Q8. How would you define the superclasses of a class?

Ans: The superclass of a class is defined using the `super()` function. When using the `super()` function, the name of the parent class is not required to access the attributes and methods of the parent class. It can be used in single and multiple inheritance. For example,

```
class Parent:
```

```
    def __init__(self, a):
        self.a = a
    def method(self):
        print("This is the method of the parent class")
```

```
class Child(Parent):
```

```
    def __init__(self):
        super().__init__(a)
```