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

It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

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

An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from left to right (by default). The search stops at the first place the attribute is found.

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

Classes are a kind of factory for creating multiple instances. Classes also support operator overloading methods, which instances inherit, and treat any functions nested in the class as methods for processing instances.

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

It always receives the instance object that is the implied subject of the method call. It’s usually called ‘self’ by convention.

Q5. What is the purpose of the \_\_init\_\_ method?

If the \\_\\_init\\_\\_ method is coded or inherited in a class, Python calls it automatically each time an instance of that class is created.

class New():

def \\_\\_init\\_\\_(self, arg1, arg2):

self.first\_var = arg1

self.second\_var = arg2

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

The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon as follows −

class ClassName:

'Optional class documentation string'

class\_suite

The class has a documentation string, which can be accessed via ClassName.\_\_doc\_\_.

The class\_suite consists of all the component statements defining class members, data attributes and functions.

Example

Following is the example of a simple Python class −

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print "Total Employee %d" % Employee.empCount

def displayEmployee(self):

print "Name : ", self.name, ", Salary: ", self.salary

The variable empCount is a class variable whose value is shared among all instances of a this class. This can be accessed as Employee.empCount from inside the class or outside the class.

The first method \_\_init\_\_() is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.

You declare other class methods like normal functions with the exception that the first argument to each method is self. Python adds the self argument to the list for you; you do not need to include it when you call the methods.

Q7. What is the process for creating a class?

The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon as follows −

class ClassName:

'Optional class documentation string'

class\_suite

The class has a documentation string, which can be accessed via ClassName.\_\_doc\_\_.

The class\_suite consists of all the component statements defining class members, data attributes and functions.

Example

Following is the example of a simple Python class −

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print "Total Employee %d" % Employee.empCount

def displayEmployee(self):

print "Name : ", self.name, ", Salary: ", self.salary

The variable empCount is a class variable whose value is shared among all instances of a this class. This can be accessed as Employee.empCount from inside the class or outside the class.

The first method \_\_init\_\_() is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.

You declare other class methods like normal functions with the exception that the first argument to each method is self. Python adds the self argument to the list for you; you do not need to include it when you call the methods.

Creating Instance Objects

To create instances of a class, you call the class using class name and pass in whatever arguments its \_\_init\_\_ method accepts.

"This would create first object of Employee class"

emp1 = Employee("Zara", 2000)

"This would create second object of Employee class"

emp2 = Employee("Manni", 5000)

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

They are classes which are used to inherit from.

class Son(Father, Mother): …

In this case Father and Mother are superclasses for Son subclass.