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

Ans:

**Object-oriented programming** (OOP) is a method of structuring a program by bundling related properties and behaviours into individual **objects**.

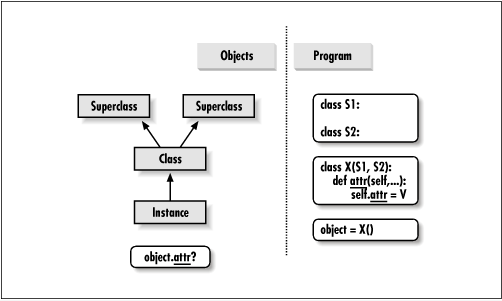
**Object-Oriented Programming(OOP), is all about creating “objects”. An object is a group of interrelated variables and functions. These variables are often referred to as properties of the object and functions are referred to as the behaviour of the objects. These objects provide a better and clear structure for the program.**

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

Ans:

**Every time we use an expression of the form object.attr where object is an instance or class object, Python searches the namespace tree at and above object, for the first attr it can find. Because lower definitions in the tree override higher ones, inheritance forms the basis of specialization.**

**The net result is a tree of attribute namespaces, which grows from an instance, to the class it was generated from, to all the superclasses listed in the class headers. Python searches upward in this tree from instances to superclasses, each time you use qualification to fetch an attribute name from an instance object.**



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

Ans:

|  |  |
| --- | --- |
| **Class** | **Object** |
| **A class is a template for creating objects in program.** | **The object is an instance of a class.** |
| **A class is a logical entity** | **Object is a physical entity** |
| **A class does not allocate memory space when it is created.** | **Object allocates memory space whenever they are created.** |
| **You can declare class only once.** | **You can create more than one object using a class.** |
| **Example: Car.** | **Example: Jaguar, BMW, Tesla, etc.** |
| **Class generates objects** | **Objects provide life to the class.** |
| **Classes can’t be manipulated as they are not available in memory.** | **They can be manipulated.** |
| **It doesn’t have any values which are associated with the fields.** | **Each and every object has its own values,which are associated with the fields.** |
| **You can create class using “class” keyword.** | **You can create object using “new” keyword in Java** |

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

Ans:

**The calling process is automatic while the receiving process is not (its explicit). This is the reason the first parameter of a function in class must be the object itself. Writing this parameter as self is merely a convention. It is not a keyword and has no special meaning in Python.**

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

Ans:

**Whenever you call a class, Python will construct a new instance of that class, and then call that class' \_\_init\_\_ method, passing in the newly constructed instance as the first argument (self).**

**Unlike many programming languages, \_\_init\_\_ isn't called the "constructor method".**

**Python's \_\_init\_\_ method is called the initializer method. The initializer method *initializes* our new class instance. So by the point that the initializer method is called the class instance has already been constructed.**

**When you make a new class in Python the first method you'll likely make is the \_\_init\_\_ method. The \_\_init\_\_ method allows you to accept arguments to your class.**

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

Ans:

**The process of creating an instance is called Instantiation.**

Q7. What is the process for creating a class?

Ans:

**The class statement creates a new class definition.**

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

Ans:

**A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class**