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

Object-oriented programming is a programming paradigm that provides a means of structuring programs so that properties and behaviors are bundled into individual objects.

OOPs provide some benefits to programming languages like:

* Modularity for easier troubleshooting
* Reuse of code through inheritance
* Flexibility through polymorphism
* Effective problem solving ,etc

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?

Key Differences between Class and Object:

* A class is a template for creating objects in a program, whereas the object is an instance of a class.
* A class is a logical entity, while an object is a physical entity.
* A class does not allocate memory space; on the other hand, an object allocates memory space.
* You can declare a class only once, but you can create more than one object using a class.
* Classes can’t be manipulated, while objects can be manipulated.
* Classes don’t have any values, whereas objects have their own values.
* You can create a class using “class” keyword, while hand you can create an object using by assigning a variable with a class with specific attributes.

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

In object-oriented programming, whenever we define methods for a class, we use self as the first parameter in each case. The self keyword is used to represent an instance (object) of the given class. If there was no self argument, the same class couldn't hold the information for more than one objects. However, since the class is just a blueprint, self allows access to the attributes and methods of each object in python. This allows each object to have its own attributes and methods. Thus, even long before creating these objects, we reference the objects as self while defining the class.

Generally, when we call a method with some arguments, the corresponding class function is called by placing the method's object before the first argument. So, anything like obj.meth(args) becomes Class.meth(obj, args). 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?

The Default \_\_init\_\_ method is like the Constructor in C++ and Java. 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.

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

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

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.
* 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.
* 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

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

The class from which a class inherits is called the parent or superclass. A class which inherits from a superclass is called a subclass, also called heir class or child class.