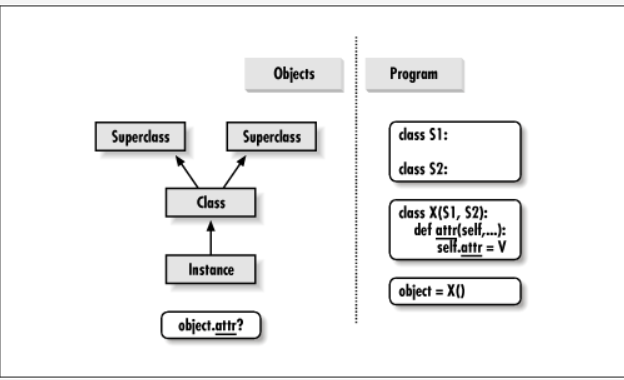
Q1. **What is the purpose of Python's OOP?**

Ans : It makes code more reusable and easier to work with larger programs. OOP programs prevent you from repeating code because a class can be defined once and reused many times.

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

Ans : Python searches upward in this tree from instances to superclasses, each time we use qualification to fetch an attribute name from an instance object.



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

 The Object is an actual thing that is built based on the 'blue print' (like the house). An instance is a virtual copy (but not a real copy) of the object

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

the first argument in a class's method function is typically named self, although we can technically name it whatever we want (though it's considered best practice to use self. This argument refers to the instance of the class itself. When you call a method on an instance of a class, Python automatically passes the instance itself as the first argument to the method. This allows the method to access and modify the object's attributes and perform operations on the object itself.

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

The \_\_init\_\_ method in Python is a special method (also known as a constructor) that is automatically called when a new instance of a class is created. Its purpose is to initialize the newly created object by setting up its initial state.

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

Creating a class instance in Python involves a few simple steps:

Define the Class: First, you need to define the class blueprint by using the class keyword followed by the class name and a colon. Inside the class definition, you specify the attributes and methods that characterize the class.

Instantiate the Class: To create an instance of the class, you call the class name followed by parentheses. This is similar to calling a function. If the class has an \_\_init\_\_ method, you need to provide the required arguments specified in the \_\_init\_\_ method signature.

Initialization: When you create an instance of the class, Python automatically invokes the \_\_init\_\_ method (if present) and passes any specified arguments to it. The \_\_init\_\_ method initializes the instance attributes based on the provided arguments.

Q7. **What is the process for creating a class?**

Creating a class in Python involves several steps:

Use the class Keyword: To define a class, you use the class keyword followed by the name of the class you want to create. Class names are typically written in CamelCase convention, starting with an uppercase letter.

Define Class Attributes and Methods: Inside the class definition, you define attributes and methods that characterize the class. Attributes represent the data associated with the class, while methods define the behavior of the class.

Optional: Define a Constructor (\_\_init\_\_ method): You can define a special method called \_\_init\_\_, which serves as the class constructor. This method is called automatically when a new instance of the class is created and is typically used to initialize instance attributes.

Optional: Define Other Methods: Apart from the \_\_init\_\_ method, you can define other methods to provide additional functionality to the class. These methods can perform various operations on the class attributes or implement specific behaviors.

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

In Python, you define superclasses (also known as parent classes or base classes) by specifying them inside parentheses after the class name in the class definition. This is known as class inheritance, where a class inherits attributes and methods from its superclasses.