**Q1. What is the purpose of Python&#39;s OOP?**

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

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

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

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

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

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

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

**SOLUTIONS**

1. The purpose of Python's OOP (Object-Oriented Programming) is to organize code into reusable, modular structures called classes that encapsulate data and behavior, allowing for more efficient and flexible programming.

2. In Python, an inheritance search looks for an attribute first in the instance itself, then in its class, then in the first parent class, then in the parent's parent class, and so on up the inheritance hierarchy until it is found or there are no more parent classes.

3. A class object is a blueprint that defines the attributes and methods of a class, while an instance object is a specific realization of that class, created by calling the class's constructor.

4. The first argument in a class's method function, conventionally named **self**, refers to the instance object calling the method and allows the method to access and modify the instance's attributes.

5. The purpose of the **\_\_init\_\_** method is to initialize the instance object's attributes with default or user-specified values when the object is created.

6. To create a class instance, you call the class's constructor by using the class name followed by parentheses, optionally passing arguments to the **\_\_init\_\_** method.

A7. To create a class in Python, you define a class using the **class** keyword followed by the class name, optionally followed by a pair of parentheses containing the superclass or super classes, and a body containing the class's attributes and methods.

8. The super classes of a class are the classes that it inherits from, either directly using the **class** statement's superclass argument or indirectly through a chain of parent classes. These super classes are also known as base classes or parent classes.