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?

A1. The purpose of Python's OOP (Object-Oriented Programming) is to provide a way of organizing and structuring code so that it is more modular, reusable, and maintainable. OOP allows you to define classes, which are templates or blueprints for creating objects. These classes can contain attributes (data) and methods (functions), and can be used to create instances of objects that can interact with each other and with the program as a whole.

A2. Inheritance search in Python looks for an attribute in the following order: First, it looks for the attribute in the instance object itself. If it is not found, it looks for the attribute in the class of the instance. If the attribute is still not found, it searches through the class hierarchy in a depth-first manner, starting with the first superclass, and looking through all the superclasses until the attribute is found or the search is exhausted.

A3. A class object is a blueprint or template for creating instances of objects, while an instance object is a specific object created from a class. In other words, a class object defines the attributes and methods that all instances of the class will have, while an instance object has its own set of unique attributes and can perform its own operations using the methods defined in the class.

A4. The first argument in a class's method function is always the "self" parameter, which refers to the instance object that the method is being called on. This parameter is special because it allows the method to access and manipulate the instance's attributes and call other methods defined in the class.

A5. The "**init**" method is a special method in Python classes that is used to initialize the instance object when it is created. It is called automatically when a new instance of the class is created, and allows you to specify the initial values of the instance's attributes.

A6. To create a class instance in Python, you first need to define the class and then create an instance of the class using the class name followed by parentheses. For example, if you have a class named "Person", you can create an instance of the class using the following code:

scss
```
person1 = Person()
```

A7. To create a class in Python, you define a new class using the "class" keyword followed by the class name and a colon, and then define the class attributes and methods within the class block. For example, the following code defines a simple class named "Person" with a single attribute and method:

ruby
```
class Person:
    name = ""

    def say_hello(self):
        print("Hello, my name is", self.name)
```

A8. The superclasses of a class are the classes that the current class inherits from, either directly or indirectly. In Python, you can define a class that inherits from one or more superclasses by including the superclass names in parentheses after the class name in the class definition. For example, the following code defines a class named "Student" that inherits from the "Person" class:

ruby
```
class Student(Person):
    student_id = ""

    def enroll(self, course):
        print(self.name, "is enrolled in", course)
```