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

Ans. OOP promotes the concept of objects, which are instances of classes, and allows developers to model complex systems more intuitively and efficiently. promotes code organization, modularity, reusability, and flexibility, leading to more efficient development and maintenance of software systems.

Q2. Where does an inheritance search look for an attribute?  
Ans. The inheritance search looks for an attribute in the following order:

1. **Instance attributes**: First, Python checks if the attribute is defined directly on the instance itself. If the attribute is found, it is returned, and the search stops.
2. **Class attributes**: If the attribute is not found on the instance, Python checks if it is defined in the class of the instance. If the attribute is found in the class, it is returned, and the search stops.
3. **Parent classes**: If the attribute is not found in the instance or its class, Python continues the search in the parent classes (superclasses) of the class, following the method resolution order (MRO). Python checks each parent class in the order specified by the MRO until it finds the attribute or reaches the end of the inheritance hierarchy.
4. **Object class**: If the attribute is not found in any of the parent classes, Python finally checks if the attribute is defined in the built-in **object** class, which is the ultimate base class for all classes in Python.

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

* Ans. A class object is a blueprint for creating instances (objects) of that class.
* It is created when the **class** statement is executed in Python.
* It defines the attributes (variables) and methods (functions) that are common to all instances of the class.
* It can have class variables and class methods, which are shared among all instances of the class.

Instance Object:

An instance object is a specific realization of a class. It is created from the class blueprint.

It is created when the class is instantiated using the class name followed by parentheses.

It represents a specific entity or object in the program.

Each instance has its own set of instance variables and can have its own state separate from other instances.

Instance objects can call the class methods and access the class variables, but they can also have their own instance methods and instance variables.

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

Ans. the first argument in a class's method function is typically named self, although any valid variable name can be used. This first argument is special because it refers to the instance of the class on which the method is being called.

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

Ans. The \_\_init\_\_ method in Python is a special method, also known as the constructor method, that is automatically called when a new instance of a class is created. Its main purpose is to initialize the newly created object with initial values or state.

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

Ans. First, you need to define the class blueprint by using the class keyword followed by the class name and a colon.

Instantiate the Class

Optional Initialization

Accessing Attributes and Methods

Using the Instance

Q7. What is the process for creating a class?

Ans. Use the class Keyword: Begin by using the class keyword followed by the name of the class you want to create.

Define Attributes: Inside the class definition, you can define attributes, which are variables that store data related to the class.

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

Ans. To define superclasses (also known as parent classes or base classes) for a class in Python, you simply include the superclass name in parentheses after the class name. This indicates that the class inherits attributes and methods from its superclass.

class Animal:

def sound(self):

print("Animal sound")

class Dog(Animal):

def bark(self):

print("Woof")