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

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data

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

attribute fetches are simply tree searches. The term inheritance is applied because objects lower in a tree inherit attributes attached to objects higher in that tree. As the search proceeds from the bottom up, in a sense, the objects linked into a tree are the union of all the attributes defined in all their tree parents, all the way up the tree.

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

**Class object** :

A class Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with *actual values*. It’s not an idea anymore, it’s an actual dog, like a dog of breed pug who’s seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.

A class object consists of:

* **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
* **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
* **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

**Instance objects:**

**SELF** is a default variable that contains the memory address of the current object. Instance variables and methods can be referred to by the self variable. When the object of a class is created, the memory location of the object is contained by its object name. This memory location is passed to the SELF internally, as SELF knows the memory address of the object, so the variable and method of an object is accessible. The first argument to any object method is SELF because the first argument is always object reference. This process takes place automatically whether you call it or not.

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

* The first parameter in the class method is the class on which you are calling the method, not (necessarily) the class that defines the method. (Having a variable that always holds the same class would probably not be that useful.)
* Q5. What is the purpose of the \_\_init\_\_ method?

**The Default \_\_init\_\_ 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 a class instance, we need to call the class by its name and pass the arguments to the class, which its **init** method accepts.

**Example:** **name = my\_class("navya","sri")** Here name is an instance of class my\_class with attributes "navya" and "sri".

Q7. What is the process for creating a class?

 keyword **class** is used to created a class in python. The syntax to create a class in python is

**class <classname>:**

**Example:** **class name:** ➞ this creates a class called name

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

A super class of a class is defined by giving super class as an arugment to the child class

**Example:**

Class daughter(mother):

Here child class daughter inherits all the attributes and methods from Superclass/Parent class mother