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

🡪 The purpose of Python's Object-Oriented Programming (OOP) is to provide a way to structure and organize code by creating objects that encapsulate data and functionality together.

Benefits of using OOP in Python:

1. Modularity and Reusability:

OOP allows you to break down a complex problem into smaller, manageable objects. Each object can have its own data and methods, making it easier to understand, modify, and reuse code. Objects can be created based on existing classes, promoting code reuse and reducing redundancy.

2. Encapsulation:

OOP encapsulates data and methods within objects. The internal details and implementation of an object are hidden, and only a well-defined interface is exposed to interact with the object. Encapsulation provides data protection, preventing direct manipulation of object state, and helps maintain code integrity.

3. Abstraction:

You can define classes with generalized attributes and behaviours, and then create specific instances of those classes with unique data values. Abstraction simplifies complex systems, allowing you to focus on essential features.

4. Inheritance:

Inheritance enables the creation of hierarchical relationships between classes. A subclass can inherit attributes and behaviors from a superclass, extending or modifying them as needed. Inheritance promotes code reuse, reduces redundancy, and facilitates the creation of specialized classes that inherit common functionality.

5. Polymorphism:

Polymorphism allows objects of different classes to be treated as interchangeable entities. It enables you to write code that can work with objects of multiple types, as long as they support a common interface. Polymorphism enhances code flexibility, extensibility, and modifiability.

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

🡪 Python searches for an attribute in an upward tree of attributes. it first searches for the attribute in its instance and then looks in the class it is generated from, to all super classes listed in its class header

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

|  |  |
| --- | --- |
| **Class Object** | **Instance Object** |
| A class object is the blueprint or template for creating instances of a particular class. | An instance object is a specific object created from a class, based on the blueprint provided by the class object. |
| It is created when the class is defined. | It is created when you instantiate the class using the class object. |
| It defines the structure and behaviour of the instances that will be created from it. | It represents a specific occurrence or individual instance of the class. |
| You can access the class object itself using the class name. | You create an instance object by calling the class object as if it were a function, which invokes the class's constructor. |

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

🡪 Python Class's usually have three types of methods which are:

Instance Methods (object level methods)

Class Methods (class level methods)

Static Methods (general utility methods)

self is the first argument for instance methods. which refers to the object itself

cls is the first argument for class methods which refers to the class itself

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

🡪 \_\_init\_\_ is a reserved method in python classes. It serves the role of a constructor in object-oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class

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.

Q7. What is the process for creating a class?

🡪 class keyword is used to create a class in python. The syntax to create a class in python is class <classname>:

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

🡪 Superclass/Parent class is given as a arugment to the child class