Q1. What is Object-Oriented Programming (OOP)?

Object-Oriented Programming is a way to design software using objects that represent real-world things. These objects have attribute properties and methods actions they can perform. The four main ideas in OOP are;

Encapsulation. It restricts direct access to some of an object's components, which means the internal state of the object is protected from unintended modifications. This enhances security and reduces system complexity.

Abstraction. This concept hides complex implementation details and only exposes the necessary functionalities, allowing users to work with higher-level interfaces without worrying about the lower-level workings.

Inheritance .This allows one class to inherit attributes and methods from another class, promoting code reuse and reducing redundancy.

Polymorphism. It allows objects of different classes to be treated as objects of a common base class. The same operation can behave differently on different classes, providing flexibility in the code.

Q2. What is a constructor in Python?

A constructor in python is a special function called (init) that sets up a new object when it's created. It initializes the object's attributes with specific values.

Q3. What's the difference between class variables and instance variables?

Class Variables are variables shared by all instances of a class. They are defined at the class level and have the same value for every instance, unless explicitly modified, while Instance Variables are variables that belong to an individual object (instance) of a class. Each object has its own copy, and changes to one object's instance variables do not affect others.

Q4. What's the difference between class methods, static methods, and instance methods?

Class Method: A method that is tied to the class rather than the instance is called a class method. It requires {cls}, or the actual class, as the first parameter and is defined using the (@classmethod) decorator.

Static Method: A static method is not instance- or class-bound. It does not take {self` or `cls` as a parameter and is specified using the (@staticmethod) decorator.

Instance Methods: They require self`, the instance, as the initial parameter and are tied to examples of the class.

Q5. When should you use class variables?

A real-world scenario would be tracking the total number of students in a school. Since this is a piece of information shared across all instances (students), it makes sense to use a class variable.