**Part 1: Theoretical Questions**

**Q1:** Explain the concept of Object-Oriented Programming (OOP). What are the four main pillars

of OOP, and how do they contribute to better software design?

**Q2:** What is the purpose of a constructor in Python? Explain how the \_\_init\_\_ method is used to

initialize an object’s attributes with an example.

**Q3:** Differentiate between class variables and instance variables. Provide an example that

illustrates how class variables are shared among all instances of a class, while instance

variables are unique to each object.

**Q4:** Define and provide an example of a class method and a static method. How do they differ

from instance methods?

**Q5:** Describe a real-world scenario where using class variables would be more appropriate than

using instance variables. Explain why class variables are suitable in this scenario.

**Q1**

**Object-Oriented Programming (OOP)** is a way of programming where we use "objects" (things created from classes) to represent real-world items or concepts. These objects can have their own data (attributes) and actions (methods).

The four main ideas behind OOP are:

**Encapsulation**: This means keeping the details inside an object hidden and only allowing controlled access to it. It's like having a TV remote where you can change the channel, but you don’t need to know how the electronics inside work.

**Abstraction**: This hides unnecessary details and shows only what's important. It's like driving a car—you know how to use the steering wheel, gas, and brake, but you don’t need to know how the engine works.

**Inheritance**: This lets one class (child class) get traits (attributes) and actions (methods) from another class (parent class). For example, if you have a general class "Animal," you can create a "Dog" class that inherits everything from "Animal" but adds dog-specific traits.

**Polymorphism**: This allows one function or method to work in different ways depending on the object. For example, both a "Dog" and "Cat" class can have a speak() method, but they may behave differently (bark or meow) when called.

These ideas help make software easier to maintain, reuse, and update.

### Q2.

A **constructor** in Python is a special function used to create and set up an object when it’s made. In Python, the constructor is the \_\_init\_\_ method. It runs automatically when you make a new object and helps to set its initial values (attributes).

### Q3:

* **Instance Variables** are specific to each object. Each object has its own version of them.
* **Class Variables** are shared among all objects of the same class. They belong to the class itself, not individual objects.

**Q4**

**Instance Methods** work on a specific object and can use its data.

* **Class Methods** work on the class itself and can change class-level data. They are defined with @classmethod.
* **Static Methods** are just regular functions inside the class that don’t use object data or class data. They are defined with @staticmethod.
* Class methods deal with the class itself, static methods don’t deal with the class or object, and instance methods are tied to individual objects.

### :

### Q5:

You should use a **class variable** when the value should be the same for all objects of a class. For example, you might want to track how many objects of a class have been created. For example counting the total number of vehicles created. Since the count should be shared among all vehicles, it’s better as a class variable.