Assignment#16

Part-2

Q. How does object-oriented programming differ from procedural programming?

Procedural programing works with Functions, sequences, loops, conditionals while **OOP** key features are Encapsulation, Inheritance, Polymorphism and Abstraction.

Q. What is a constructor in Python? What happens if it is not defined explicitly?

A **constructor** is a special method used to initialize the **object** when it is created from a class. If we don't define a constructor than Python will define it automatically.

Q. Explain the concept of class vs instance variables with example.

Class variables can be accessed by all instances of that class sharing the common value while object variables have their different values because each instance has its own properties.

```
lass Dog:
    species = "Canis familiaris"  # 🟡 Class Variable

    def __init__(self, name, age):
        self.name = name        # 🔵 Instance Variable
        self.age = age
```

Q. What is the purpose of _str() and how is it different from __repr_()?

✅ **__str__() — *User-Friendly Representation***

Purpose: Returns a **readable, human-friendly string** version of the object.
Triggered by: print(object) or str(object)
Should be easy to **read and understand**.

✅ **__repr__() — *Developer-Friendly / Debug Representation***

Purpose: Returns an **unambiguous string** meant for **debugging and development**.
Triggered by: repr(object)
just typing the object name in the Python shell.

Q. Why is it beneficial to use object-oriented programming in larger projects?

OOP is beneficial in large projects because it provides:

Advantage	Why It Matters
Modularity	Breaks code into manageable chunks

Reusability	Reduces redundancy
Encapsulation	Protects data and reduces errors
Scalability	Easier to add features or expand
Maintainability	Simplifies debugging and updates
Collaboration	Supports teamwork and division of labor