**Day 4: OOP in Python**

* Classes and objects
* \_init\_\_, self, \_\_str\_\_
* Inheritance and polymorphism
* Class methods and static methods
* Dunder methods

**✅ 1. Creating Classes and Objects**

class Person:

def \_\_init\_\_(self, name, age): # Constructor

self.name = name

self.age = age

def greet(self):

print(f"Hi, I'm {self.name} and I'm {self.age} years old.")

# Creating an object

p1 = Person("Alice", 30)

p1.greet()

**✅ 2. Class Variables vs Instance Variables**

class Dog:

species = "Canis familiaris" # Class variable (shared)

def \_\_init\_\_(self, name):

self.name = name # Instance variable (unique to object)

d1 = Dog("Rex")

d2 = Dog("Buddy")

print(d1.species, d1.name)

print(d2.species, d2.name)

**✅ 3. Inheritance**

class Animal:

def speak(self):

print("Animal speaks")

class Cat(Animal):

def speak(self):

print("Meow")

c = Cat()

c.speak() # Outputs: Meow

**✅ 4. Class Methods and Static Methods**

class Book:

count = 0

def \_\_init\_\_(self, title):

self.title = title

Book.count += 1

@classmethod

def total\_books(cls):

print(f"Total books: {cls.count}")

@staticmethod

def library\_info():

print("Open from 9AM to 5PM")

b1 = Book("Python 101")

b2 = Book("AI Basics")

Book.total\_books()

Book.library\_info()

**✅ 5. Dunder Methods (\_\_str\_\_, \_\_len\_\_, etc.)**

class Car:

def \_\_init\_\_(self, brand):

self.brand = brand

def \_\_str\_\_(self):

return f"This is a {self.brand} car"

c = Car("Toyota")

print(c) # Calls \_\_str\_\_

**🧠 Mini Exercises:**

1. Create a Student class with name, roll, and a method to display info.
2. Create a base class Vehicle and a subclass Bike that overrides a method.
3. Add a \_\_str\_\_ method to a class to print a friendly string when the object is printed.