Class_Act25

🧠 Python OOP Activity: Polymorphism – Shape Drawer

📌 Objective:

Understand **polymorphism** by using **the same method name** across different classes to perform **different behaviors**.

🟢 Scenario:

You are designing a shape drawer. All shapes can be drawn using a method called `draw()`, but each shape displays a different message when it’ s drawn.

🔧 Instructions:

1. Create a **base class** called `Shape` with a method `draw()` (it can just print a generic message like `"Drawing a shape..."`).
2. Create two **child classes**:
   - `Circle` – override `draw()` to print `"Drawing a circle."`
   - `Rectangle` – override `draw()` to print `"Drawing a rectangle."`
3. Write a function called `draw_shape(shape)` that takes any object and calls its `draw()` method.
4. Create one object from each class and test them **using the same function** — this is **polymorphism** in action!