Class_Act29

🧠 Python OOP: 4 Pillars – Task-Based Activities

---

1. Encapsulation Activity – Personal Contact Info Manager

Objective:
Protect sensitive data like phone numbers using private attributes and provide access only through getter and setter methods.

Instructions:

- Create a class `Contact`.
- Attributes:
  - `name` (public)
  - `__phone_number` (private)
- Add methods:
  - `get_phone_number()` – returns the phone number.
  - `set_phone_number(new_number)` – updates the phone only if it's 10 digits.
- Test with:
  - Valid and invalid numbers.

---

2. Inheritance Activity – Transportation System

Objective:
Use inheritance to create specialized vehicle types based on a general base class.

Instructions:

- Create a base class `Transport` with:
  - Attributes: `brand`, `capacity`
  - Method: `display_info()`
- Create child classes:
  - `Bus` → add method `route()`: prints "Bus is on route."
  - `Train` → add method `track()`: prints "Train is on track."

- Create objects of both and test.

---

3. Polymorphism Activity – Device Startup

Objective:
Use the same method name in multiple classes to perform different behaviors (method overriding).

Instructions:

- Create a base class `Device` with method `start()`.
- Create two subclasses:
  - `Laptop` → `start()` prints "Laptop is booting up."
  - `Smartphone` → `start()` prints "Smartphone is starting."
- Create a function `turn_on(device)` that calls `device.start()`.
- Call `turn_on()` on both objects to demonstrate polymorphism.

---

4. Abstraction Activity – Online Payment System

Objective:
Use abstract classes and force child classes to implement their own version of a method.

Instructions:

- Import `ABC` and `abstractmethod`.
- Create an abstract class `Payment` with:
  - Attributes: `user`, `amount`
  - Abstract method: `process()`
- Create subclasses:
  - `GcashPayment` → prints "Processing GCash payment for [user]"
  - `PaypalPayment` → prints "Processing PayPal payment for [user]"
- Create objects and call `process()` for each.