OOP Fundamentals – Assignment Set

🔹 Topic Coverage:

- Class & Object
- `__init__` constructor
- Instance attributes & methods
- Class attributes & methods
- Default arguments
- Object interactions
- Encapsulation (private variables)
- Real-life modeling problems

Questions

---

1. Create a class `Laptop`

Attributes:

- Brand
- Model
- Price

Methods:

- `show_details()` → print all information

Hint: Use `__init__()` to initialize and `self.brand` etc. to store values.

---

2. Create a class `Circle`

Attributes:

- radius

Methods:

- `area()` → returns area (πr²)
- `circumference()` → returns circumference (2πr)

Hint: Use `math.pi` from the `math` module.

---

3. Create a class `Employee` with class attribute `company_name = "TechSoft"`

Each employee has:

- name, position, salary

Methods:

- `show_info()` → print employee details
- `change_company(cls, new_name)` → class method to update company name

Hint: Use `@classmethod` and `cls.company_name = new_name`.

---

4. Design a class `ShoppingCart`

Attributes:

- customer name
- cart (list of items)

Methods:

- `add_item(item)`
- `remove_item(item)`
- `view_cart()`

Hint: Use `self.cart = []` in `__init__`.

---

5. Create a class `BankAccount` with balance initialized to 0

Methods:

- `deposit(amount)`
- `withdraw(amount)` (only if balance is sufficient)
- `check_balance()`

Hint: Keep `self.balance` private (i.e., `self.__balance`), and use methods to access/modify it.

---

6. Create a class `Movie` with attributes: title, director, rating

- Store all created movies in a class-level list.
- Add a method `is_hit()` that returns True if rating > 8.

Hint: Use a class attribute `all_movies = []` and `Movie.all_movies.append(self)` inside `__init__`.

---

7. Create a class `Book` with method `set_discount(percent)`

- price is an instance attribute
- discount is class-wide, applied on all books

Hint: Use a class attribute `discount_percent = 0` and apply `@classmethod` to update it.

---

8. Create a class `SchoolStudent`

- Attributes: name, class_name, marks (dict)
- Method: `add_marks(subject, score)`
- Method: `average()`

Hint: Use a dictionary to hold subject-mark pairs.

---

9. Create a class `Temperature`

- Accept temperature in Celsius
- Provide methods:
  - `to_fahrenheit()`
  - `to_kelvin()`

Hint:

- °F = (°C × 9/5) + 32
- K = °C + 273.15

---

10. Build a class `FlightBooking`

Attributes:

- passenger_name
- flight_no

- destination

Keep a class attribute `total_bookings`, and increase it every time a new booking is made.

Hint: Use `FlightBooking.total_bookings += 1` in `__init__`.

🧩 Want a Real-Life Inspired Bonus?

11. Class: `TaskManager`

You are building a CLI-based task tracker for your students.

Attributes:

- user_name
- task_list (initially empty)

Methods:

- `add_task(task)`
- `remove_task(task)`
- `view_tasks()`

Hint: Use a list to store tasks. Add/remove using list methods.