1. Write About OOP Concepts?
   Ans: Here is it.

🔑 1. Encapsulation

Encapsulation is the practice of **bundling data and methods** inside a class and **restricting direct access** to some of the object's components.

Why?

- Protects data from unintended modification
- Provides controlled access through **getters/setters**

Example (Python)

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # private

    def deposit(self, amount):
        self.__balance += amount

    def get_balance(self):
        return self.__balance
```

🔑 2. Inheritance

Inheritance allows a class (child) to **acquire the properties and behaviors** of another class (parent).

Why?

- Promotes code reusability
- Supports hierarchical relationships

Example

```
class Animal:
    def speak(self):
        return "Some sound"

class Dog(Animal):
    def speak(self):
        return "Bark"
```

🔑 3. Polymorphism

Polymorphism means **many forms**. It allows methods with the same name to behave differently depending on the object.

Types:

- **Compile-time (overloading)** – same method, different parameters

- **Runtime (overriding)** – child class provides its own version of a method

Example

```
def animal_sound(animal):
    print(animal.speak())
```

🔑 4. Abstraction

Abstraction hides complex implementation details and exposes only essentials.

Why?

- Simplifies usage
- Reduces complexity

Example

```
from abc import ABC, abstractmethod

class Car(ABC):
    @abstractmethod
    def drive(self):
        pass

class Tesla(Car):
    def drive(self):
```

```
        print("Driving autonomously")
```

🔑 Bonus: Other Important OOP Principles

✔ Composition

Building complex objects by combining simpler ones.

✔ Association / Aggregation

Defines relationships between classes.

✔ SOLID Principles

A set of 5 guidelines for good OOP design.

If you want, I can also provide:
✅ UML diagrams
✅ Code examples in any programming language
✅ A full OOP cheat sheet
Just tell me!