Pillars of OOP (Object-Oriented Programming)

1. **Encapsulation**

Encapsulation means bundling data (variables) and the functions (methods) that operate on that data into a single unit, i.e., the object.

Key Ideas

- Protect internal state by hiding direct access to data.
- Provide controlled access through public methods only.
- Typical access pattern:
  - Private fields
  - Public getter & setter methods

Types of Public Methods

Method Type	Purpose
Getter	Retrieves private data
Setter	Updates private data

Code Example – Encapsulation with `Product`

```
class Product {
  // Private field
  #price;

  constructor(price) {
    this.#price = price;
  }

  // Getter – returns the data
  getPrice() {
    return this.#price;
  }

  // Setter – updates the data with validation
  setPrice(value) {
    if (value > 0) {
      this.#price = value;
    }
  }
}

// Usage
const item = new Product(100);
console.log(item.getPrice()); // 100
```

```
item.setPrice(150);
console.log(item.getPrice()); // 150
// Direct access like item.price is NOT allowed
```

---

2. **Abstraction**

Abstraction shows only the essential features of an object while hiding unnecessary details.

- Focus on what an object does rather than how it does it.
- Users interact with a simple interface without needing to understand the internal logic.

> **Analogy**: A car driver only needs to know the steering, brake, and accelerator; the engine internals are hidden.

- Implemented via private / protected members, exposing only public methods intended for external use.

---

3. **Polymorphism**

Polymorphism literally means “many forms.”

- Same interface, different behaviors.
- Achieved in two ways:

Type	Description
Compile-time (Method Overloading)	Multiple methods with the same name but different parameters.
Run-time (Method Overriding)	Subclass provides a specific implementation of a method already defined in its superclass.

> **Example**: A Person can behave differently in different roles:
>
> - Father at home
> - Manager at work
> - Customer at a restaurant

---

4. **Inheritance**

Inheritance allows a child (subclass) to inherit properties and behaviors from a parent (superclass).

Key Points

Aspect	Benefit
Reusability	Common functionality lives in the base class.
Extensibility	Child classes can extend/override parent behavior.

Aspect	Benefit
Hierarchy	Organizes classes in a clear tree-like structure.

Types of Inheritance

Type	Description
Single	One child ← one parent
Multiple	One child ← multiple parents
Multilevel	A ← B ← C
Hierarchical	Many children ← one parent

Code Example – Single Inheritance in Python

```
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog(Animal):  # Dog inherits from Animal
    def speak(self):
        print("Dog barks")

dog = Dog()
dog.speak()  # Output: Dog barks
```

Summary Table – All Four Pillars

Pillar	Core Principle	Keyword / Technique
Encapsulation	Hide & protect data	`private`, getters, setters
Abstraction	Show essential parts only	`public` interface
Polymorphism	One interface, many forms	Overloading & overriding
Inheritance	Reuse & extend code	`extends` / `class Sub(Parent)`