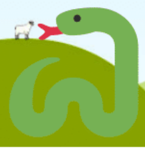

What is OOP?

Object-Oriented Programming (OOP) is a programming style that structures code using **objects**, which combine **data** (attributes) and **behavior** (methods).
It helps make code more **modular**, **reusable**, and **easier to maintain**.

🧠 Key Terms

Class	A blueprint for creating objects
Object	An instance of a class
Attribute	A variable stored in an object
Method	A function defined inside of a class
__init_	Special method that runs when an object is created
self	Refers to the current instance of the class

🔑 The Four Pillars of OOP

Encapsulation

Hides an object's internal state and provides public methods to interact with it safely.
Uses **getter** and **setter** methods to access or update private attributes.

Inheritance

Allows one class to inherit attributes and methods from another.
Promotes code reuse and logical hierarchy.

Polymorphism

Lets different classes use the same method name with different behavior.
Improves flexibility when working with objects.

Abstraction

Hides internal details and shows only what's necessary.
Uses the abc module to define abstract classes and methods.

💻 Code Examples

Encapsulation

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

    def get_balance(self):  # Getter
        return self.__balance

    def set_balance(self, amount):  # Setter
        if amount >= 0:
            self.__balance = amount
```

Inheritance

```python
class Animal:
    def speak(self):
        return "Sound"

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

Polymorphism

```python
animals = [Dog(), Cat()]
for a in animals:
    print(a.speak())  # Woof / Meow
```

Abstraction

```python
from abc import ABC, abstractmethod

class Vehicle(ABC):
    @abstractmethod
    def drive(self):
        pass
```