En Python, la **Programación Orientada a Objetos (POO)** es fundamental porque, en este lenguaje, ¡absolutamente todo es un objeto!

Aquí tienes tu guía desde cero:

## 1. Conceptos Básicos

### La Clase vs. El Objeto

- **Clase:** Es el plano o molde (ejemplo: el plano de una casa).
- **Objeto:** Es la construcción real basada en el plano (ejemplo: tu casa física).

### La Anatomía de una Clase en Python

A diferencia de C#, en Python usamos la palabra clave `class` y un método especial llamado `__init__` (el constructor).

Python

```python
class Perro:
    # El Constructor (donde definimos los atributos)
    def __init__(self, nombre, raza):
        self.nombre = nombre  # Atributo
        self.raza = raza      # Atributo

    # Un Método (lo que el objeto puede hacer)
    def ladrar(self):
        return f"¡Guau! Me llamo {self.nombre}"

# Crear un objeto (instanciar)
mi_perro = Perro("Rex", "Pastor Alemán")
print(mi_perro.ladrar())
```

## 2. Los 4 Pilares de la POO

### A. Encapsulamiento

Sirve para "esconder" datos sensibles. En Python, usamos guiones bajos _ o __ para indicar que algo es privado.

Python

```python
class CuentaBancaria:
    def __init__(self, saldo):
        self.__saldo = saldo  # Privado (con __)

    def mostrar_saldo(self):
        return f"Saldo actual: ${self.__saldo}"
```

### B. Herencia

Permite que una clase hija herede atributos y métodos de una clase padre.

Python

```python
class Animal:
    def respirar(self):
        print("Respirando...")

class Gato(Animal): # Gato hereda de Animal
```

```
    def maullar(self):
        print("¡Miau!")
```

## C. Polimorfismo

Diferentes clases pueden tener métodos con el mismo nombre pero comportamientos distintos.

Python
```
# Ambos tienen el método 'hablar', pero hacen cosas diferentes
class Pato:
    def hablar(self): return "Cuack!"

class Humano:
    def hablar(self): return "Hola!"
```

## D. Abstracción

Enfocarse en lo que hace el objeto en lugar de cómo lo hace (usando clases base).