1. Programación orientada a objetos

La Programación Orientada a Objetos (POO) es un paradigma que organiza el código alrededor de `objetos`, que combinan datos y funcionalidad. En Python, todo es un objeto.

2. Conceptos Básicos

- **Objetos**: Instancias de clases que encapsulan datos y funciones.
- **Clases**: Plantillas para crear objetos.

3. Clase

Una **clase** define las propiedades y los comportamientos de un conjunto de objetos. Una clase es una plantilla para crear objetos.

La clase puede tener:

- **Atributos**: Características de un objeto.
- **Métodos**: Funciones asociadas a una clase.

4. Crear una clase

Para crear una clase, utiliza la palabra clave `class` seguida del nombre de la clase. Los métodos o funciones se identifican con `def`.

```
class Coche:
    def __init__(self, marca, modelo):
        self.marca = marca
        self.modelo = modelo
        self.encendido = False

    def encender_apagar(self):
        self.encendido = not self.encendido
```

El método `__init__` es un método especial llamado **constructor** que inicializa los atributos de la clase.

Los métodos en una clase se incluyen dentro de la definición de la clase y pueden acceder a los atributos del objeto a través del parámetro `self`.

5. Objetos

Un **objeto** es una instancia de una clase. Cada objeto que creemos tendrá todas las propiedades y comportamientos definidos para su clase.

Los objetos se crean dándole un identificador y asignándole la clase correspondiente. Al indicar la clase, también pasamos los atributos que queremos que tenga.

```
mi_coche = Coche("Toyota", "Corolla")
```

6. Ejemplo con pokemon

Aquí podemos ver una **clase** pokemon que tiene:

- Tres **atributos**: nombre, tipo y nivel
- Tres **métodos**: init, atacar y subir_nivel

Todo lo que está dentro de class no se ejecuta, simplemente indica como es esta clase.

```
class Pokemon:
    def __init__(self, nombre, tipo, nivel):
        self.nombre = nombre
        self.tipo = tipo
        self.nivel = nivel

    def atacar(self):
        print(f"{self.nombre} ha usado un ataque de {self.tipo}")

    def subir_nivel(self):
        self.nivel += 1
        print(f"{self.nombre} ha subido al nivel {self.nivel}")
```

Para crear un objeto:

```
pikachu = Pokemon("Pikachu", "Trueno", 5)
```

Si queremos llamar a la función `atacar()`. Recuerda que `pikachu` es un objeto de la clase `pokemon`, y esta clase tiene definido un método `atacar()`.

```
pikachu.atacar()
```

Para acceder al atributo `tipo`:

```
print(pikachu.tipo)
```