Programación orientada a objetos: clases, objetos, atributos, métodos

1. Introducción a la programación orientada a objetos (POO)

La Programación Orientada a Objetos (POO) es un paradigma de programación que utiliza objetos y sus interacciones para diseñar y desarrollar aplicaciones. Algunas ventajas de utilizar la POO incluyen la modularidad, reusabilidad de código y facilidad para realizar mantenimiento y depuración.

2. Clases y objetos

Clases

Una clase es una plantilla que define la estructura y comportamiento de un objeto. Contiene atributos y métodos que serán compartidos por todos los objetos de esa clase.

Ejemplo de definición de una clase en Python:

```
class Persona:
    pass
```

Objetos

Un objeto es una instancia de una clase, que representa a una entidad individual con su propio conjunto de atributos y comportamientos.

Ejemplo de creación de objetos en Python:

```
persona1 = Persona()
persona2 = Persona()
```

3. Atributos y métodos

Atributos

Los atributos son variables asociadas a un objeto. Cada objeto de una clase puede tener diferentes valores para sus atributos.

Ejemplo de atributos en Python:

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
```

Métodos

Los métodos son funciones asociadas a un objeto que pueden acceder y modificar sus atributos.

Ejemplo de métodos en Python:

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def presentarse(self):
        print(f"Hola, mi nombre es {self.nombre} y tengo {self.edad} años.")
```

La función `__init__`

El método `__init__` es un método especial en las clases de Python. Se llama automáticamente cuando se crea un objeto de la clase. Usualmente, se utiliza para inicializar los atributos del objeto.

El parámetro `self`

El parámetro `self` es una referencia al objeto que llama al método. Se utiliza para acceder a los atributos y métodos del objeto desde dentro de la clase.

4. Ejemplo práctico

Vamos a crear una clase `Persona` con atributos y métodos, y luego crearemos objetos de esa clase.

Ejemplo de clase Persona:

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def presentarse(self):
        print(f"Hola, mi nombre es {self.nombre} y tengo {self.edad} años.")
```

Creación de objetos `Persona`:

```
persona1 = Persona("Ana", 28)
persona2 = Persona("Carlos", 35)

persona1.presentarse()
persona2.presentarse()
```

Salida:

```
Hola, mi nombre es Ana y tengo 28 años.
Hola, mi nombre es Carlos y tengo 35 años.
```