

# Programación Orientada a Objetos (POO)

## Polimorfismo

El polimorfismo es un concepto fundamental en la programación orientada a objetos que permite que objetos de diferentes clases puedan ser tratados como si fueran del mismo tipo. Esto significa que diferentes objetos pueden responder de manera diferente a un mismo mensaje o método, dependiendo de su clase específica.

## Polimorfismo de métodos

El polimorfismo de métodos permite que diferentes clases implementen métodos con el mismo nombre pero con comportamientos específicos según la clase.

**El siguiente código muestra un ejemplo de polimorfismo en Python, donde diferentes clases derivadas implementan el mismo método de manera específica. El polimorfismo permite que el mismo método se comporte de manera diferente según el tipo de objeto que lo invoque.**

**Clase Base: `Animal`**

- `def hablar(self)`
  - **Propósito:** Define una interfaz común para todos los animales, especificando que el método `hablar()` debe ser implementado por las subclases.
  - **Implementación:** Lanza una excepción `NotImplementedError` para indicar que las subclases deben proporcionar su propia implementación del método `hablar()`.

**Clase Derivada 1: `Perro`**

- `def hablar(self)`
  - **Propósito:** Implementa el método `hablar()` para la clase `Perro`.
  - **Implementación:** Devuelve la cadena "`¡Guau!`", que es el sonido característico que hace un perro.

### Clase Derivada 2: Gato

- `def hablar(self)`
  - **Propósito:** Implementa el método `hablar()` para la clase `Gato`.
  - **Implementación:** Devuelve la cadena "¡Miau!", que es el sonido característico que hace un gato.

### Clase Derivada 3: Vaca

- `def hablar(self)`
  - **Propósito:** Implementa el método `hablar()` para la clase `Vaca`.
  - **Implementación:** Devuelve la cadena "¡Muu!", que es el sonido característico que hace una vaca.

### Función que Utiliza Polimorfismo

- `def hacer_hablar(animal)`
  - **Propósito:** Acepta un objeto de tipo `Animal` (o una de sus subclases) y llama al método `hablar()` de ese objeto.
  - **Implementación:**
    - \* Llama al método `hablar()` del objeto `animal`.
    - \* Imprime el resultado del método `hablar()`.

```
# Clase base
class Animal:
    def hablar(self):
        raise NotImplementedError("El método hablar() debe \
ser implementado por las subclases")

# Clase derivada 1
class Perro(Animal):
    def hablar(self):
        return "¡Guau!"

# Clase derivada 2
class Gato(Animal):
    def hablar(self):
        return "¡Miau!"

# Clase derivada 3
class Vaca(Animal):
    def hablar(self):
```

```
        return "¡Muu!"

# Función que utiliza polimorfismo
def hacer_hablar(animal):
    print(animal.hablar())

# Crear instancias
mi_perro = Perro()
mi_gato = Gato()
mi_vaca = Vaca()

# Llamar a la función hacer_hablar() con difernetes tipos de animales
hacer_hablar(mi_perro)
hacer_hablar(mi_gato)
hacer_hablar(mi_vaca)
```

```
¡Guau!
¡Miau!
¡Muu!
```