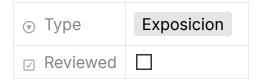
# Polimorfismo en Python



## Polimorfismo y Polimorfismo Parametrico

La capacidad de una función de comportarse de cierta manera con un tipo de dato especifico.

Polimorfimso paramétrico, se puede usar con los diferentes tipos de variables

Herencia que trae los métodos de la clase padre, como ejecutar y hacer el código mas flexible, operaciones que solo funcionan con cierto tipo de datos

Se puede reutilizar muchas veces, extension de funcionalidad, implementarlo en clases

Polimorfismo en sobrecarga de operadores, y el polimorfismo, métodos con clases

Permite la flexibilidad en el código para poder llamarlo cuando sea necesario, se asimila a una herencia publica.

#### Por que es util el polimorfismo parametrico?

Tienes el mismo método en una clase y puedes aplicarlo en otra, un mismo método con diferentes tipos de datos

```
def duplicar(valor):
return valor * 2
```

```
print(duplicar(5)) # Salida 10
print(duplicar(HOLA)) # Salida HOLAHOLA
print(duplicar([1,2,3])) # Salida [1,2,3][1,2,3]
```

Polimorfímso simple, necesita un tipo de dato especifico

Polimorfismo paramétrico no necesariamente se necesita un tipo de dato, se corre con el riesgo de sobrecargar los operadores?

|                        |                                     |   | 1     |
|------------------------|-------------------------------------|---|-------|
| Aspecto                | Polimorfismo Simple                 | Polimorfismo Paramétrico  |       |
| Tipo de Comportamiento | Sobreescritura de métodos           | Genéricos o<br>funciones/métodos que<br>aceptan múltiples tipos |       |
| Implementación         | Herencia y métodos sobrescritos     | Tipado dinámico o genéricos                                     |       |
| Ejemplo                | Métodos hacer_sonido() en subclases | Función duplicar() que acepta<br>int, str, list                 |       |
| Ventaja Principal      | Uniformidad en el uso de métodos    | Generalidad y adaptabilidad                                     |       |
| Uso Común              | Jerarquías de clases en OOP         | Funciones genéricas y<br>estructuras de datos                   | 13 de |

### **Ejemplo Presentado de Polimorfismo**

```
class Notificacion:
    def __init__(self, noti):
        self.noti = noti

def Message(self):
```

```
print(f"alberto: {self.noti}")
class Insta(Notificacion):
    def Message(self):
        print(f"alberto: {self.noti}")
class Whatsapp(Notificacion):
    def Message(self):
        print(f"alberto: {self.noti}")
class Gmail(Norificacion):
    def Message(self):
        print(f"alberto: {self.noti}")
noti = input("Ingrese el mensaje que va a mandar: ")
noti1 = Insta(noti)
noti2 = Whatsapp(noti)
noti3 = Gmail(noti)
noti1.Message()
noti2.Message()
noti3.Message()
```

## **Ejercicios**

 Realizar una clase padre que contenga dos clases hijas, una con cada método, sin atributos

```
class Animal:
    def hacer_sonido(self):
        print("Animal haciendo sonido")

class Animal(Perro):
    def hacer_sonido(self):
        print("Guau Guau")
```

```
class Animal(Gato):
   def hacer_sonido(self):
     print("Miau")
```

2. Realizar una clase transporte y agrega 4 diferentes medios de trasportes con sus respectivos costos

```
# Realizar una clase transporte y agrega 4 diferentes medios de
class Transporte:
    def __init__(self, costo):
        self.costo = costo
    def presentar_costo(self):
        print(f"Este transporte cuesta: {self.costo}")
class Metro(Transporte):
    def presentar_costo(self):
        print(f"Este transporte cuesta: {self.costo}")
class Camion(Transporte):
    def presentar costo(self):
        print(f"Este transporte cuesta: {self.costo}")
class Carro(Transporte):
    def presentar costo(self):
        print(f"Este transporte cuesta: {self.costo}")
class Avion(Transporte):
    def presentar_costo(self):
        print(f"Este transporte cuesta: {self.costo}")
```

```
metro = Metro(5)
camion = Camion(12.50)
carro = Carro(40.50)
avion = Avion(2450)

metro.presentar_costo()
camion.presentar_costo()
carro.presentar_costo()
avion.presentar_costo()
```