

**Universidad Nacional del Altiplano**  
**Facultad de Ingeniería Estadística e Informática**

**Docente:**

Ing. Coyla Idme Leonel

**Alumno:**

Ticona Miramira Roberto Angel

**Métodos y Sobrecarga**

**» MÉTODOS**

Un método es una función definida dentro de una clase que realiza una acción sobre los datos del objeto. Los métodos representan el comportamiento del objeto.

**Ejemplo**

Crear el objeto persona.

- Clase: Persona.
- Atributo: Nombre.
- Acción: Saludar.
- Objetos: `miPersona = Persona("Roberto")`, `miPersona2 = Persona("Maria")`

**Código**

```
1 class Persona:
2     def __init__(self, nombre):
3         self.nombre = nombre
4
5     def saludar(self):
6         print(f"Hola, soy {self.nombre}")
7
8 def main():
9     miPersona = Persona("Roberto")
10    miPersona2 = Persona("Maria")
11    miPersona.saludar()
12    miPersona2.saludar()
13
14 if __name__ == "__main__":
15     main()
```

**Ejecución**

```
1 Hola, soy Roberto
2 Hola, soy Maria
```

**» PARÁMETROS EN MÉTODOS**

Los métodos pueden recibir parámetros para trabajar con datos externos. El primer parámetro de todo método es de instancia (**self**), que representa el objeto mismo.

**Ejemplo**

Crear el objeto calculadora que permita sumar dos números.

- Clase: Calculadora.
- Atributos: a, b.
- Acción: Sumar.
- Objeto: miCalculadora(1,3)

### Código

```
1 class Calculadora:
2     def sumar(self, a, b):
3         self.a = a
4         self.b = b
5         return a + b
6
7 miCalculadora = Calculadora()
8 suma = miCalculadora.sumar(1,3)
9 print(f"La suma de {miCalculadora.a} y {miCalculadora.b} es {suma}")
```

### Ejecución

```
1 La suma de 1 y 3 es 4
```

### » RETORNO DE MÉTODOS

un método puede devolver un valor usando la instrucción return. Esto permite usar el resultado del método en otras partes del programa.

### Ejemplo

Hallar el área de un rectángulo.

- Clase: Rectángulo.
- Atributos: base, altura.
- Acción: calcular\_area().
- Objeto: rectángulo = Rectangulo(10,6)

### Código

```
1 class Rectangulo:
2     def __init__(self, base, altura):
3         self.base = base
4         self.altura = altura
5
6     def calcular_area(self):
7         return self.base * self.altura
8
9 rectangulo = Rectangulo(10,5)
10 area = rectangulo.calcular_area()
11 print(f"El área del rectángulo es: {area}")
```

### Ejecución

```
1 El área del rectángulo es: 50
```

### » SOBRECARGA DE MÉTODOS

Consiste en definir múltiples versiones de un método con el mismo nombre pero diferentes parámetros.

## Ejemplo 1

Definir el objeto operaciones que puede sumar 2 y 3 números.

- Clase: Operaciones.
- Atributos: a, b, c = None.
- Acción: sumar()
- Objetos: objeto = Operaciones()

### Código

```
1 class Operaciones:
2     def Sumar(self, a, b, c = None):
3         if c is not None:
4             return a + b + c
5         else:
6             return a + b
7
8 objeto = Operaciones()
9 suma = objeto.Sumar(1,2,0)
10 suma2 = objeto.Sumar(1,2)
11 print(suma)
12 print(suma2)
```

### Ejecución

```
1 3
2 3
```

## Ejemplo 2

Calcular el área y perímetro de un círculo.

- Clase: Circulo.
- Atributo: radio.
- Acción: calcular\_area, calcular\_perimetro, mostrar\_informacion.
- Objetos: circulo = Circulo(7)

### código

```
1 import math
2 class Circulo:
3     def __init__(self, radio):
4         self.radio = radio
5
6     def calcular_area(self):
7         area = math.pi * self.radio ** 2
8         return area
9
10    def calcular_perimetro(self):
11        perimetro = 2 * math.pi * self.radio
12        return perimetro
13
14    def mostrar_informacion(self):
15        print(f"El radio del circulo es: {self.radio}")
16        print(f"El área es: {self.calcular_area():.2f}")
17        print(f"El perímetro es: {self.calcular_perimetro():.2f}")
18
19 circulo = Circulo(7)
```

```
20 | circulo.mostrar_informacion()
```

### Ejecución

```
1 | El radio del circulo es: 7
2 | El área es: 153.94
3 | El perímetro es: 43.98
```

### Ejemplo 3

Calcular la hipotenusa de un triángulo rectángulo.

- Clase: Hipotenusa.
- Atributos: cateto1, cateto2.
- Acción: hipotenusa, mostrar.
- Objetos: hipotenusa = Hipotenusa(3,4)

### Código

```
1 | class Hipotenusa:
2 |     def __init__(self, cateto1, cateto2):
3 |         self.cateto1 = cateto1
4 |         self.cateto2 = cateto2
5 |
6 |     def hipotenusa(self):
7 |         return (self.cateto1 ** 2 + self.cateto2 ** 2) ** 0.5
8 |
9 |     def mostrar(self):
10 |         print(f"La hipotenusa de {self.cateto1} y {self.cateto2} es
    |             ↪ {self.hipotenusa()}")
11 |
12 | hipotenusa = Hipotenusa(3,4)
13 | hipotenusa.mostrar()
```

### Ejecución

```
1 | La hipotenusa de 3 y 4 es 5.0
```