Métodos y Sobrecarga

Alumno: Puma Huanca Anthony Rusbel
Docente: Ing. Coyla Idme Leonel
Lenguajes de programación II – FINESI
Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

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("Silvana")`, `miPersona2 = Persona("Maria")`.

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

    def saludar(self):
        print(f"Hola, soy {self.nombre}")

def main():
    miPersona = Persona("Roberto")
    miPersona2 = Persona("Maria")
    miPersona.saludar()
    miPersona2.saludar()

if __name__ == "__main__":
    main()
```

Listing 1: Código Ejemplo

Ejecución:

```
Hola, soy Silvana
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 = Calculadora()`.

```
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}")
```

Listing 2: Código Parámetros en métodos

Ejecución:

```
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: `rectangulo = Rectangulo(10, 5)`.

```python
class Rectangulo:
    def __init__(self, base, altura):
        self.base = base
        self.altura = altura

    def calcular_area(self):
        return self.base * self.altura

rectangulo = Rectangulo(10, 5)
area = rectangulo.calcular_area()
print(f"El  rea  del rect ngulo es: {area}")
```

Listing 3: Código Retorno de métodos

Ejecución:

```
El área del rectángulo es: 50
```

Sobrecarga de métodos

Consiste en definir múltiples versiones de un método con el mismo nombre pero diferentes parámetros. En Python, la sobrecarga se simula usando argumentos por defecto o `*args`.

Ejemplo 1

Definir el objeto `Operaciones` que puede sumar 2 y 3 números.
Clase: `Operaciones`.
Atributos: a, b, c = None.
Acción: `Sumar()`.
Objeto: `objeto = Operaciones()`.

```python
class Operaciones:
    def Sumar(self, a, b, c=None):
        if c is not None:
            return a + b + c
        else:
            return a + b

objeto = Operaciones()
suma = objeto.Sumar(1, 2, 0)
suma2 = objeto.Sumar(1, 2)
print(suma)
print(suma2)
```

Listing 4: Código Sobrecarga Ejemplo 1

Ejecución:

```
3
3
```

Ejemplo 2

Calcular el área y perímetro de un círculo.
Clase: `Circulo`.
Atributo: radio.
Acción: `calcular_area`, `calcular_perimetro`, `mostrar_informacion`.
Objeto: `circulo = Circulo(7)`.

```
import math

class Circulo:
    def __init__(self, radio):
        self.radio = radio

    def calcular_area(self):
        area = math.pi * self.radio**2
        return area

    def calcular_perimetro(self):
        perimetro = 2 * math.pi * self.radio
        return perimetro

    def mostrar_informacion(self):
        print(f"El radio del circulo es: {self.radio}")
        print(f"El  rea  es: {self.calcular_area():.2f}")
        print(f"El per metro es: {self.calcular_perimetro():.2f}")

circulo = Circulo(7)
circulo.mostrar_informacion()
```

Listing 5: Código Sobrecarga Ejemplo 2

Ejecución:

```
El radio del circulo es: 7
El área es: 153.94
El perímetro es: 43.98
```

Ejemplo 3

Calcular la hipotenusa de un triángulo rectángulo.
Clase: `Hipotenusa`.
Atributos: cateto1, cateto2.
Acción: `hipotenusa`, `mostrar`.
Objeto: `hipotenusa = Hipotenusa(3, 4)`.

```
class Hipotenusa:
    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()
```

Listing 6: Código Sobrecarga Ejemplo 3

Ejecución:

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