Programación II

Programación Orientada a Objetos (Parte 2)

Universidad Nacional de Rosario.
Facultad de Ciencias Exactas, Ingeniería y Agrimensura.

Atributos y Métodos de una Clase

Atributo de clase: Este atributo es compartido por todas las instancias de la clase. Se puede acceder a él a través de la clase o de cualquier instancia.

Atributo de instancia: Es específico de cada instancia de la clase. Aunque es posible definir atributos de instancia en métodos distintos al constructor, es una buena práctica inicializados en el constructor.

Método de instancia: Este tipo de método se llama en una instancia específica de la clase.

Método de clase: Los métodos de clase se identifican con @classmethod. Se puede llamar en la clase misma y tiene acceso a los atributos de clase. No necesita una instancia de la clase para para ser llamado.

Veamos un ejemplo simple

```
class Ejemplo:
    contador = 0  # Atributo de clase

    def __init__(self):
        Ejemplo.contador += 1  # Incr. el contador
                               # al crear una instancia
    @classmethod
    def obtener_contador(cls):
        return cls.contador  # Acceder al atributo
                             # de clase usando cls
# Crear instancias
objeto1 = Ejemplo()
objeto2 = Ejemplo()

# Llamar al método de clase
print(Ejemplo.obtener_contador())  # Salida: 2
```

Veamos un ejemplo simple

```
class Ejemplo:
    contador = 0  # Atributo de clase

    def __init__(self):
        Ejemplo.contador += 1  # Incr. el contador
                               # al crear una instancia
    @classmethod
    def obtener_contador(cls):
        return cls.contador  # Acceder al atributo
                             # de clase usando cls
# Crear instancias
objeto1 = Ejemplo()
objeto2 = Ejemplo()

# Mostrar el valor del atributo de clase
print(Ejemplo.contador) # Salida: 2
```

Uso de `cls`

En Python, `cls` es una convención que se utiliza como nombre del primer parámetro en un método de clase. Es similar al uso de `self` en los métodos de instancia, que se refiere a la instancia específica de la clase. En el caso de un método de clase, `cls` se refiere a la propia clase y no a una instancia de la misma.

Cuando se define un método de clase, se usa el decorador `@classmethod`, y el primer parámetro de este método es `cls`. Esto permite que el método acceda y modifique atributos de clase, así como llamar a otros métodos de clase.

Un método de clase no puede acceder directamente a un atributo de instancia. Los métodos de clase están diseñados para operar sobre la clase misma y sus atributos de clase, mientras que los atributos de instancia son específicos de cada objeto creado a partir de la clase.

Ejemplo

Desarrolla una clase Cuenta_Bancaria que simule una cuenta bancaria. La clase tendrá la capacidad de gestionar depósitos, retiros y calcular intereses aplicando una tasa de interés.

- **Atributos:**
  - **Atributo de clase:**
    - **tasa_de_interes:** Un valor decimal que representa la tasa de interés anual que se aplica a los saldos de las cuentas. Por defecto, puede ser 0.02 (2 %).
  - **Atributos de instancia:**
    - **nombre_titular:** El nombre del titular de la cuenta.
    - **saldo:** El saldo de la cuenta del cliente.

Ejemplo

Desarrolla una clase Cuenta_Bancaria que simule una cuenta bancaria. La clase tendrá la capacidad de gestionar depósitos, retiros y calcular intereses aplicando una tasa de interés.

- **Métodos:**
  - **Métodos de clase:**
    - **cambiar_tasa_de_interes(nueva_tasa)**: Cambia el valor de tasa_de_interes a nueva_tasa.
  - **Métodos de instancia:**
    - **depositar(cantidad):** Aumenta el saldo de la cuenta en la cantidad especificada.
    - **retirar(cantidad):** Disminuye el saldo de la cuenta en la cantidad especificada, siempre que haya suficiente saldo.
    - **calcular_interes():** Calcula y devuelve el interés en base al saldo actual y la tasa de interés.
    - **mostrar_saldo():** Retorna el saldo actual de la cuenta.

Ejemplo

```
class Cuenta_Bancaria:
    tasa_de_interes = 0.02  # Atributo de clase

    def __init__(self,
                 nombre_titular,
                 saldo_inicial=0):
        self.nombre_titular = nombre_titular
        self.saldo = saldo_inicial

    @classmethod
    def cambiar_tasa_de_interes(cls, nueva_tasa):
        cls.tasa_de_interes = nueva_tasa

    def depositar(self, cantidad):
        self.saldo += cantidad
```

Ejemplo

```
class Cuenta_Bancaria:
    ...

    def retirar(self, cantidad):
        if cantidad <= self.saldo:
            self.saldo -= cantidad

    def calcular_interes(self):
        return (self.saldo *
                Cuenta_Bancaria.tasa_de_interes)

    def mostrar_saldo(self):
        return f'Saldo actual: {self.saldo}'
```

¿PREGUNTAS?