

Programación Orientada a Objetos (POO)

Métodos

Los métodos son funciones que pertenecen a una clase y se utilizan para definir los comportamientos de los objetos.

- Métodos de instancia
- Métodos de clase
- Métodos estáticos

Los métodos se definen en la definición de la clase.

Métodos de instancia

Son métodos que operan sobre una instancia específica de la clase. * Los métodos de instancia deben recibir *self* como parámetro. * *self* se utiliza para acceder a los atributos y otros métodos de la instancia.

Ejemplo de Método en una Clase en Python

1. Definición de la Clase Persona con un Método respirar:

- La clase `Persona` incluye un método llamado `respirar`, que imprime un mensaje en la consola.

2. Creación de una Instancia y Llamada al Método:

- Se crea una instancia de la clase `Persona` llamada `persona1`.
- Se llama al método `respirar` de la instancia `persona1`.

Explicación

- Método `respirar`:

- El método `respirar` es un método de instancia de la clase `Persona`. Los métodos de instancia requieren el primer parámetro `self`, que es una referencia al objeto actual.
- Cuando `persona1.respirar()` se llama, el método `respirar` se ejecuta, imprimiendo el mensaje “La persona respira” en la consola.

```
class Persona:  
    def respirar(self):  
        print("La persona respira")
```

```
persona1 = Persona()  
persona1.respirar()
```

La persona respira

Ejemplo de Encapsulación y Métodos en una Clase en Python

1. Definición de la Clase `Persona`:

- La clase `Persona` incluye un constructor `__init__` que inicializa un atributo privado `__nombre` y un método `respirar` que imprime un mensaje utilizando este atributo.

2. Creación de una Instancia y Llamada al Método:

- Se crea una instancia de la clase `Persona` llamada `persona1` con el nombre “José”.
- Se llama al método `respirar` de la instancia `persona1`.

Explicación

• Atributo Privado `__nombre`:

- El atributo `__nombre` es privado debido a la doble subrayado (`__`) al inicio del nombre del atributo. Esto significa que no se puede acceder directamente desde fuera de la clase.

• Método `respirar`:

- El método `respirar` accede al atributo privado `__nombre` y utiliza este valor para imprimir un mensaje indicando que la persona respira.

```
class Persona:
    def __init__(self, nombre):
        self.__nombre = nombre
    def respirar(self):
        print(f"{self.__nombre} respira")

persona1 = Persona("José")
persona1.respirar()
```

José respira