

Programación Orientada a Objetos (POO)

Herencia

La herencia es un concepto fundamental en la programación orientada a objetos que permite a una clase derivada (o subclase) heredar atributos y métodos de una clase base (o superclase). Esto permite reutilizar código y tener más organización.

Conceptos Clave de Herencia

- **Clase Base (Superclase)**
- **Clase Derivada (Subclase)**
- **Sobreescripción de Métodos**
- **Uso de la función *super()***
- **Herencia Múltiple**

Superclase y subclase

- **Clase Base (Superclase):** La clase que se está extendiendo o de la que se heredan atributos y métodos. Define los atributos y métodos comunes que serán compartidos por las clases derivadas.
- **Clase Derivada (Subclase):** La clase que hereda de la clase base. Puede utilizar los atributos y métodos de la clase base y también definir sus propios atributos y métodos adicionales o sobrescribir los existentes.

Ejemplo utilizando Herencia

Este código define una jerarquía de clases para modelar vehículos utilizando la herencia en Python. Se presentan dos subclases, **Coche** y **Motocicleta**, que heredan de una superclase común, **Vehiculo**.

Superclase: Vehiculo

- `__init__(self, marca, modelo)`

- **Propósito:** Constructor de la clase `Vehiculo`. Inicializa los atributos `__marca` y `__modelo`.
 - **Parámetros:**
 - * `marca`: Marca del vehículo.
 - * `modelo`: Modelo del vehículo.
 - **Comportamiento:** Establece los valores de `__marca` y `__modelo` como atributos privados de la instancia.
- `def informacion(self)`
 - **Propósito:** Imprime la información del vehículo.
 - **Comportamiento:** Muestra la marca y el modelo del vehículo utilizando los atributos privados `__marca` y `__modelo`.

Subclase: Coche

- `__init__(self, marca, modelo, numero_puertas)`
 - **Propósito:** Constructor de la clase `Coche`, que hereda de `Vehiculo`.
 - **Parámetros:**
 - * `marca`: Marca del coche.
 - * `modelo`: Modelo del coche.
 - * `numero_puertas`: Número de puertas del coche.
 - **Comportamiento:** Llama al constructor de la superclase `Vehiculo` para inicializar `marca` y `modelo`. Inicializa el atributo `__numero_puertas` como privado.
- `def mostrar_puertas(self)`
 - **Propósito:** Imprime el número de puertas del coche.
 - **Comportamiento:** Muestra el valor del atributo privado `__numero_puertas`.

Subclase: Motocicleta

- `__init__(self, marca, modelo, tipo)`
 - **Propósito:** Constructor de la clase `Motocicleta`, que hereda de `Vehiculo`.
 - **Parámetros:**
 - * `marca`: Marca de la motocicleta.
 - * `modelo`: Modelo de la motocicleta.
 - * `tipo`: Tipo de motocicleta (por ejemplo, Cruiser).
 - **Comportamiento:** Llama al constructor de la superclase `Vehiculo` para inicializar `marca` y `modelo`. Inicializa el atributo `__tipo` como privado.
- `def mostrar_tipo(self)`
 - **Propósito:** Imprime el tipo de motocicleta.

- **Comportamiento:** Muestra el valor del atributo privado `__tipo`.

```
# Superclase
class Vehiculo:
    def __init__(self, marca, modelo):
        self.__marca = marca
        self.__modelo = modelo

    def informacion(self):
        print(f"Marca: {self.__marca}, Modelo: {self.__modelo}")

# Subclase
class Coche(Vehiculo):
    def __init__(self, marca, modelo, numero_puertas):
        Vehiculo.__init__(self, marca, modelo)
        self.__numero_puertas = numero_puertas

    def mostrar_puertas(self):
        print(f"Número de puertas: {self.__numero_puertas}")

class Motocicleta(Vehiculo):
    def __init__(self, marca, modelo, tipo):
        Vehiculo.__init__(self, marca, modelo)
        self.__tipo = tipo

    def mostrar_tipo(self):
        print(f"El tipo de la motocicleta es: {self.__tipo}")

# Crear instancias
mi_coche = Coche("Toyota", "Corolla", 4)
mi_motocicleta = Motocicleta("Harley", "Sportster", "Cruister")

mi_coche.mostrar_puertas()
mi_coche.informacion()
mi_motocicleta.mostrar_tipo()
mi_motocicleta.informacion()
```

```
Número de puertas: 4
Marca: Toyota, Modelo: Corolla
El tipo de la motocicleta es: Cruister
Marca: Harley, Modelo: Sportster
```