**Herencia y Polimorfismo en Python**

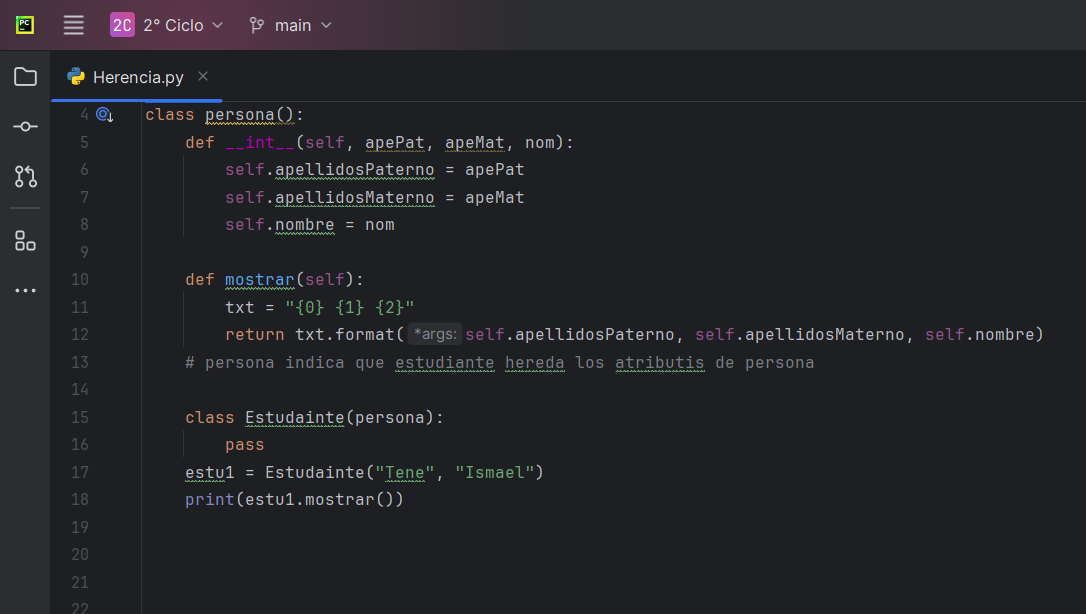
En programación orientada a objetos (POO) es un paradigma de programación que proporciona una forma de estructurar programas de modo que las propiedades y compartimentos se agrupen en objetos individuales.

**Herencia en Python**

Es un concepto fundamental en la programación orientada a objetos en la cual permite crear una nueva clase que es una versión modificada de una clase existente. La clase original se denomina clase base o clase principal, y la nueva clase se denomina clase derivada o clase secundaria. La clase hija hereda todos los atributos y comportamientos de la clase padre, pero también puede agregar nuevos o modificar los existentes.

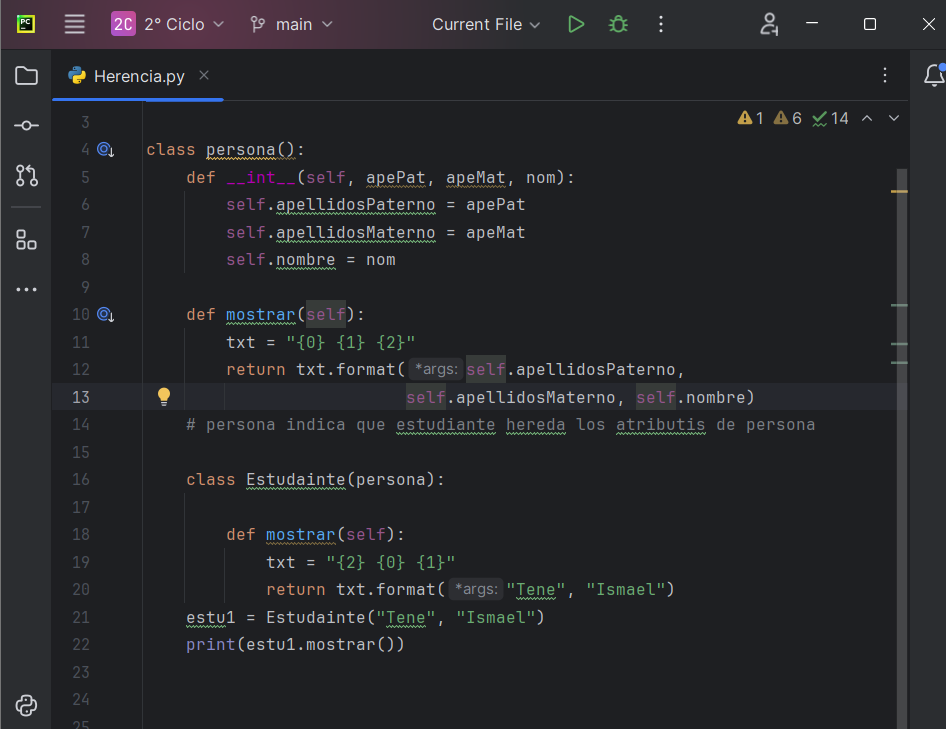
**Beneficios de la Herencia**

1. **Reutilización de código**: Puedes utilizar el código de la clase padre en las clases hijas, evitando la duplicación y facilitando el mantenimiento.
2. **Modularidad**: Facilita la organización del código en módulos más pequeños y manejables.
3. **Extensibilidad**: Permite extender y personalizar el comportamiento de las clases sin modificar el código original.
4. **Polimorfismo**: Permite tratar objetos de diferentes clases de una manera uniforme.



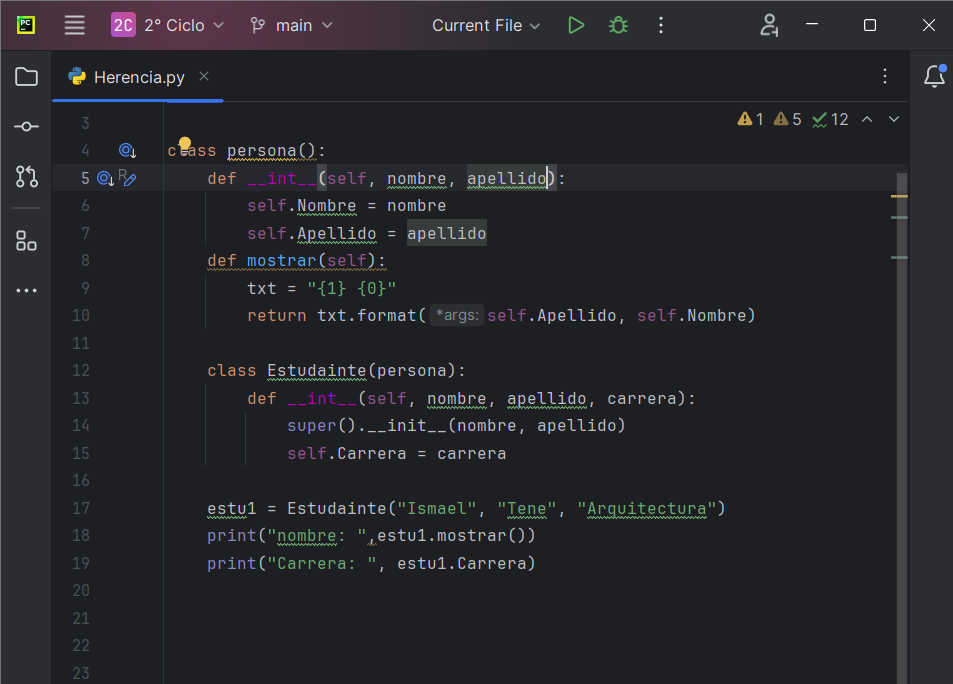
En Python, la herencia se declara pasando la clase principal como parámetro a la clase segundaria.

Sobreescritura de métodos



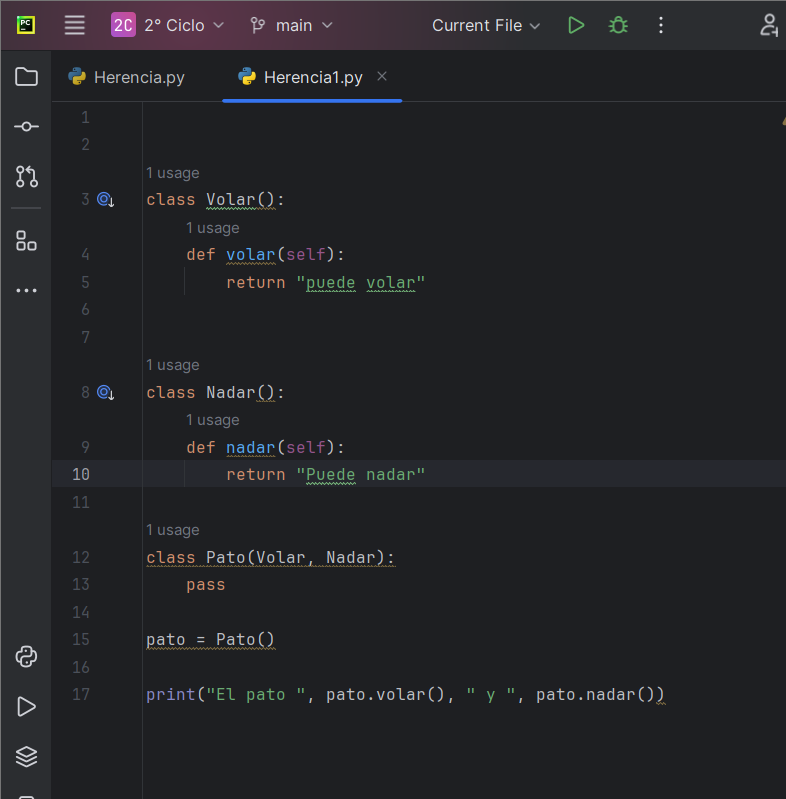
Uso de super()

La función super() se utiliza para llamar a métodos de la clase padre desde la subclase. Esto es útil cuando necesitas extender la funcionalidad de un método heredado.



Herencia Multiple

Python también soporta herencia múltiple, donde una clase puede heredar de múltiples clases. Esto se hace listando todas las clases padres entre paréntesis:



**Polimorfismo en Python**

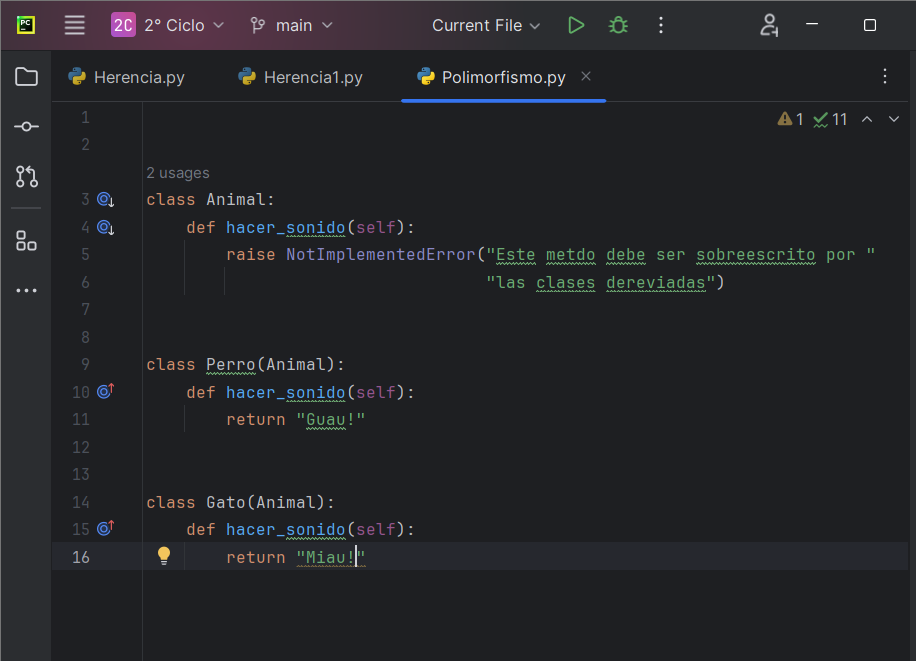
En polimorfismo es otro concepto fundamental de programación orientada a objetos que se refiere a la capacidad de un objeto de adoptar muchas formas. En Python, el polimorfismo se implementa de dos maneras: sobrecarga de métodos y anulación de métodos.

**Interfaz Común**

* Las clases diferentes pueden implementar los mismos métodos.
* Ejemplo: Clases Perro y Gato que heredan de una clase base Animal y sobrescriben el método hacer\_sonido.

La sobrecarga de métodos se refiere a la capacidad de una clase de tener múltiples métodos con el mismo nombre, pero con diferentes parámetros. En Python, la sobrecarga de métodos se realiza utilizando argumentos predeterminados o argumentos variables.

En la anulación de métodos se refiere a la capacidad de una clase secundaria de cambiar la implementación de un método que heredó de su clase principal. Esto se hace simplemente declarando un método en la clase secundaria con el mismo nombre que el método en la clase principal.



En Python el uso de herencia y polimorfismo le permite crear código flexible y reutilizable. En herencia permite crear nuevas clases a partir de clases existentes y el polimorfismo permite que los objetos de diferentes clases sean tratados de la misma manera. Estos conceptos son fundamentales para crear software complejo y son una parte esencial de cualquier curso de Python.

**Descripción del Ejemplo**

1. **Definición de Clases**:
   * Clase base Animal con un método abstracto hacer\_sonido.
   * Clases derivadas Perro y Gato que implementan hacer\_sonido.
2. **Función Polimórfica**:
   * imprimir\_sonido acepta un objeto Animal y llama a hacer\_sonido.
3. **Ejemplo de Uso**:
   * Instancias de Perro y Gato se pasan a imprimir\_sonido.

**Ventajas**

* **Flexibilidad**: Facilita la integración de nuevas clases.
* **Extensibilidad**: Permite agregar nuevas funcionalidades sin modificar el código existente.
* **Mantenibilidad**: Mejora la claridad y la estructura del código.

https://cursa.app/es/pagina/programacion-orientada-a-objetos-en-python-herencia-y-polimorfismo-en-python#:~:text=La%20herencia%20le%20permite%20crear,de%20cualquier%20curso%20de%20Python.