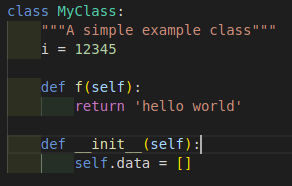
**Clases**

Las clases proveen una forma de empaquetar datos y funcionalidad juntos. Al crear una nueva clase, se crea un nuevo tipo de objeto, permitiendo crear nuevas instancias de ese tipo. Cada instancia de clase puede tener atributos adjuntos para mantener su estado. Las instancias de clase también pueden tener métodos (definidos por su clase) para modificar su estado.

Definicion clases:

Las definiciones de clases, al igual que las definiciones de funciones (instrucciones [def](https://docs.python.org/es/3/reference/compound_stmts.html" \l "def)) deben ejecutarse antes de que tengan efecto alguno. (Es concebible poner una definición de clase dentro de una rama de un [if](https://docs.python.org/es/3/reference/compound_stmts.html" \l "if), o dentro de una función.)



Parahacer referencia a atributosse usa la sintaxis estándar de todas las referencias a atributos en Python: objeto.nombre. Los nombres de atributo válidos son todos los nombres que estaban en el espacio de nombres de la clase cuando ésta se creó.

**Herencia**

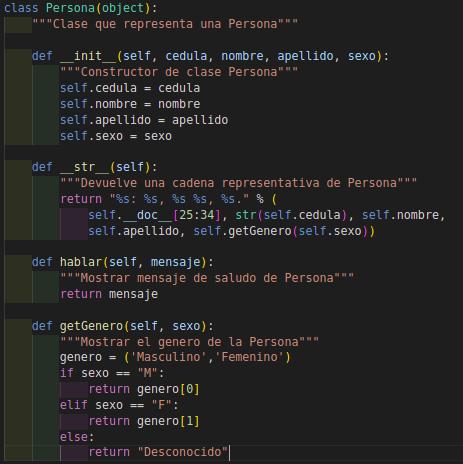
La herencia es una de las premisas y técnicas de la POO la cual permite a los programadores crear una clase general primero y luego más tarde crear clases más especializadas que re-utilicen código de la clase general. La herencia también le permite escribir un código más limpio y legible.

El nombre ClaseBase debe estar definido en un ámbito que contenga a la definición de la clase derivada. En el lugar del nombre de la clase base se permiten otras expresiones arbitrarias.

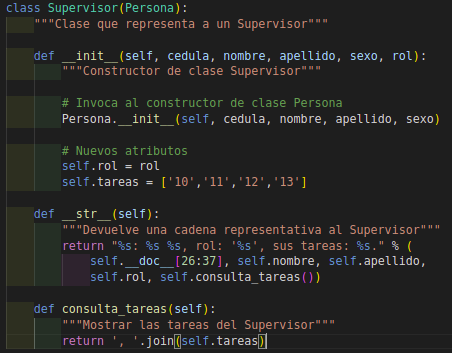
Las clases derivadas pueden redefinir métodos de su clase base. Como los métodos no tienen privilegios especiales cuando llaman a otros métodos del mismo objeto, un método de la clase base que llame a otro método definido en la misma clase base puede terminar llamando a un método de la clase derivada que lo haya redefinido. (Para los programadores de C++: en Python todos los métodos son en efecto virtuales.)

Ejemplo:

Clase Base



Clase que hereda



**Herencia Multiple**

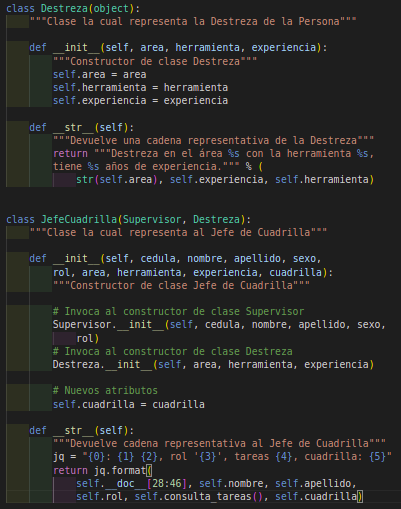
La herencia múltiple es la capacidad de una subclase de heredar de múltiples súper clases.

Esto conlleva un problema, y es que si varias súper clases tienen los mismos atributos o métodos, la subclase sólo podrá heredar de una de ellas.

En estos casos Python dará prioridad a las clases más a la izquierda en el momento de la declaración de la subclase.

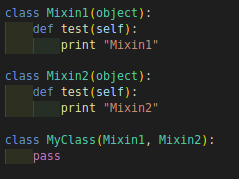
Ejemplo

La clase JefeCuadrilla usa la herencia multiple, puesto que hereda de la clase Supervisor y Destreza



**Mixins**

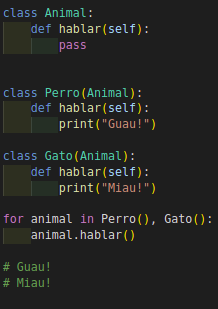
Un Mixin es una clase que hereda de la clase por defecto, object en el caso de Python, y define un conjunto de métodos para agregar comportamiento a alguna otra entidad. Un mixin por sí solo puede resultar inútil, pero al combinarlos y agregarlos a una clase resultan ser una herramienta muy potente que ayuda a la reutilización de código sin ambigüedades y sin generar efectos colaterales.



**Polimorfismo**

En lenguajes de programación como Python, que tiene tipado dinámico, el polimorfismo va muy relacionado con el duck typing.

Al ser un lenguaje con tipado dinámico y permitir duck typing, en Python no es necesario que los objetos compartan un interfaz, simplemente basta con que tengan los métodos que se quieren llamar.



En este ejemplo se observa que la variable animal cambia su forma, de perro o gato.

**Self en Python**

El parámetro self se refiere al objeto instanciado de esa clase sobre el cual se está invocando dicho método. Es decir, el objeto que se uso para llamar al método.

Python, dentro de los métodos definidos de una clase, establece que el primer parámetro definido en un método recibirá el objeto con el cual se invoca dicho método.

Este parámetro es usado dentro de la implementación del método para modificar el contenido o atributos de dicho objeto como se desee.

Por lo tanto, es una condición necesaria que todos los métodos de una clase que puedan ser llamados a través de un objeto tengan al menos un parámetro, el cual se asignará automáticamente al objeto usado en la invocación.

Aunque en la definición del método, self es el primer parámetro, a la hora de llamar a dicho método no hace falta pasarle el propio objeto como primer parámetro explícitamente ya que Python lo hace de manera implícita sin necesidad de hacerlo manualmente. Es decir, el primer parámetro self del método se asigna automáticamente al propio objeto y el resto de parámetros a los argumentos con que llames al método.