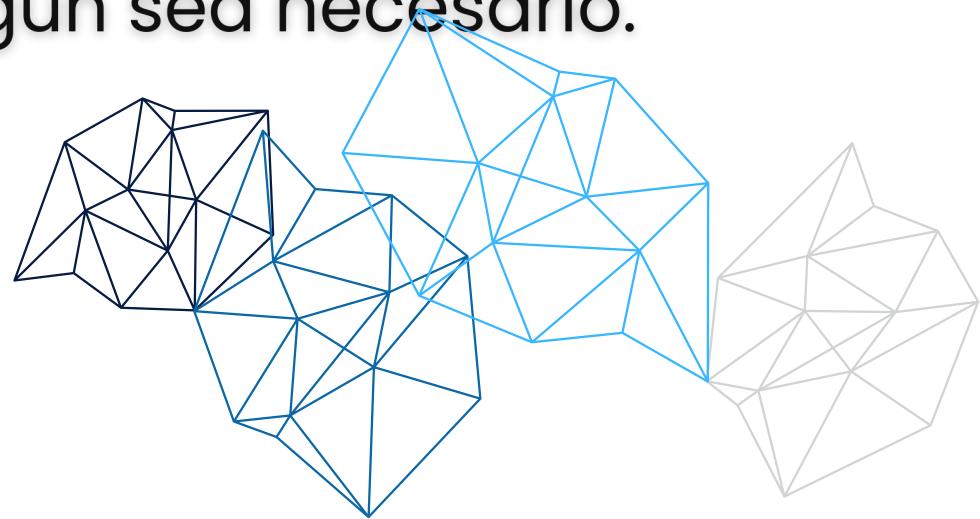


Herencia

Cuando en Python indicamos que una clase extiende (o hereda de) otra clase estamos indicando que es una clase hija de esta y que, por lo tanto, hereda todos sus métodos y variables. Este es un poderoso mecanismo para la reusabilidad del código. Podemos heredar de una clase, por lo cual partimos de su estructura de variables y métodos, y luego añadir lo que necesitemos o modificar lo que no se adapte a nuestros requerimientos.

En la herencia, una clase llamada "clase derivada" o "subclase" hereda atributos y métodos de otra clase llamada "clase padre" o "superclase". La clase derivada puede ampliar o modificar el comportamiento de la clase base al agregar nuevos atributos o métodos, o al sobrescribir los existentes.

Al heredar de una clase base, la clase derivada tiene acceso a todos los atributos y métodos públicos de la clase base. Puede utilizarlos directamente o modificarlos según sea necesario.

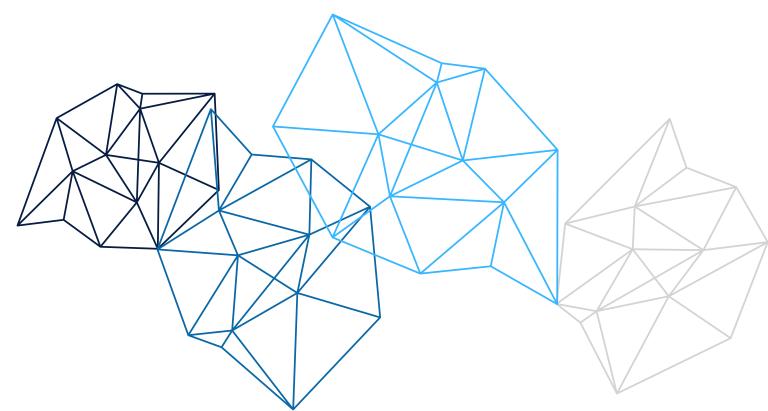


Herencia simple

Si un método no hace lo que nosotros queríamos podemos sobrescribirlo (overriding). Bastará para ello que definamos un método con el mismo nombre y argumentos. Para llamar a cualquier método de la clase Padre se puede:

- Utilizar la sintaxis `super()` y el nombre del método:
`super().<NOMBRE_DEL_METODO>`
- Invocar al nombre de la clase y el nombre del método:
`<NOMBRE_CLASE_PADRE>.<NOMBRE_DEL_METODO>`

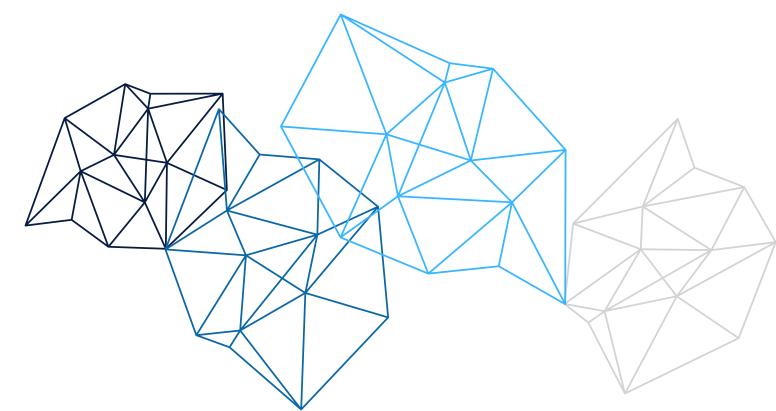
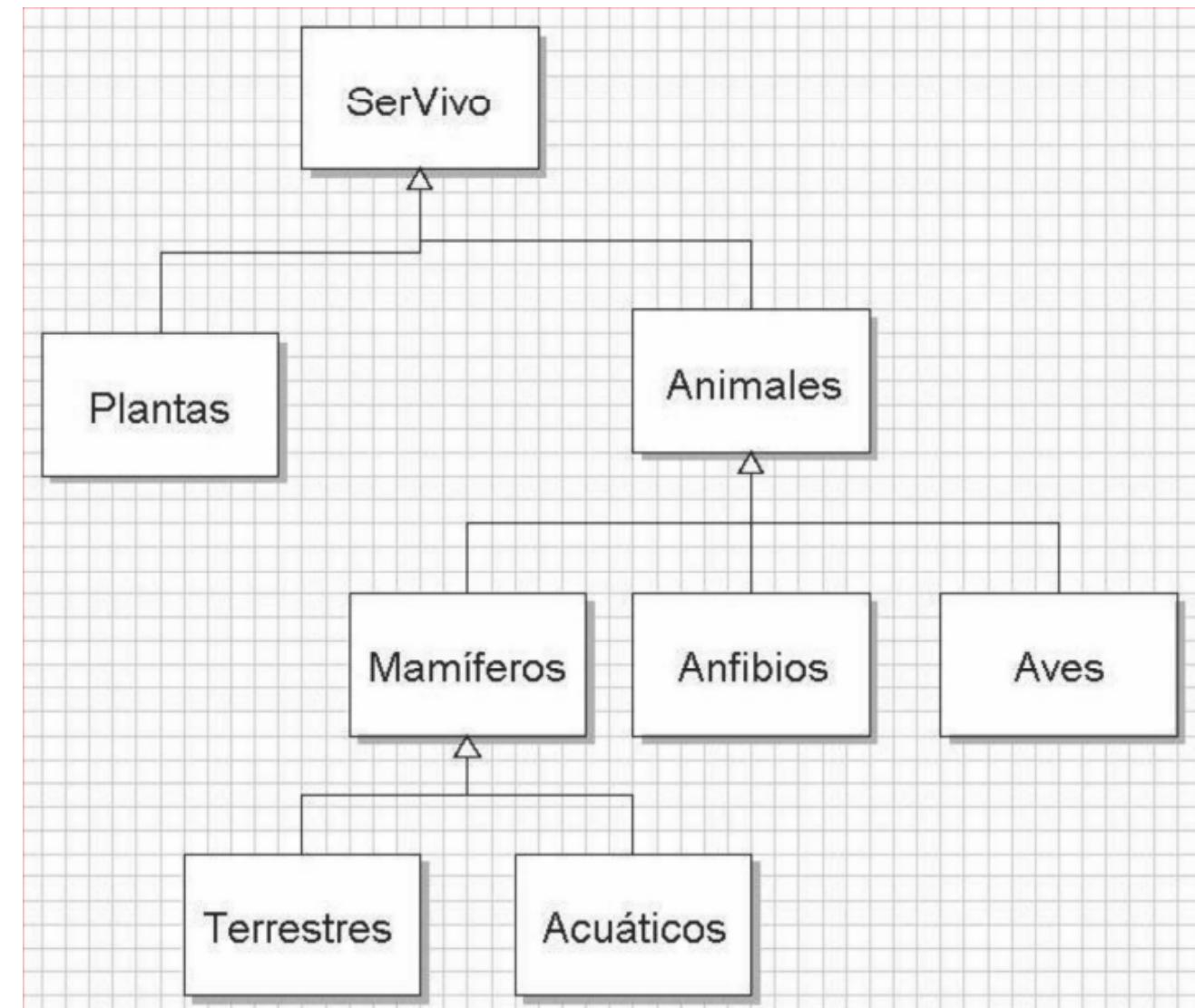
Para sobreescibir un método que no hace lo que nosotros queríamos, podemos sobreescibirlo. Esto se hace mediante el polimorfismo de sustitución. Así, bastará para ello que definamos un método con el mismo nombre y los mismos argumentos que el método definido en la clase padre.



Jerarquía de herencias

Se pueden crear jerarquías completas (como un árbol genealógico) de clases.

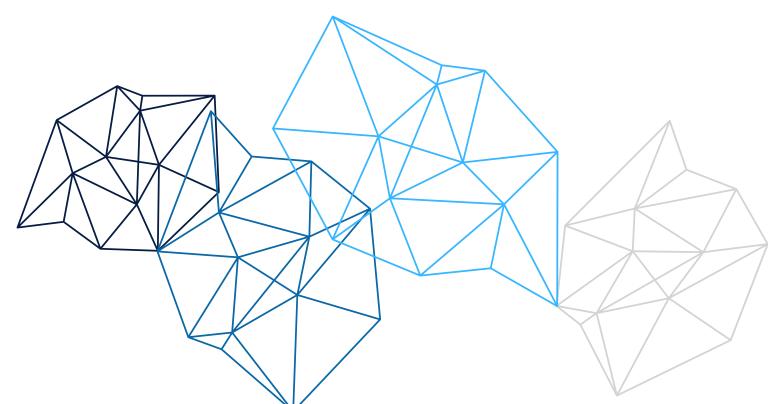
Por ejemplo, un perro tendrá todos los atributos de los mamíferos, y los mamíferos serán una especialización de los animales, que a su vez son una especialización de los seres vivos.



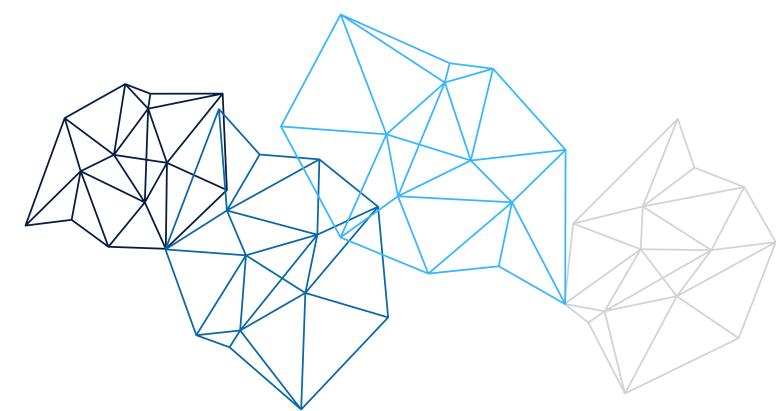
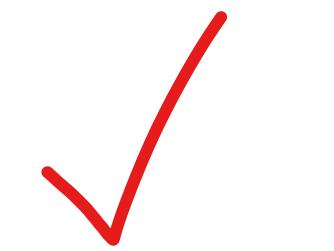
Herencia múltiple

A diferencia de lenguajes como Java y C#, el lenguaje Python permite la herencia múltiple, es decir, se puede heredar de múltiples clases. 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.

Esto puede ocurrir cuando una clase hereda de diferentes abstracciones sin relación entre sí. Para llamar a los métodos o acceder a los atributos de la clase padre, se puede utilizar la misma sintaxis que en la herencia simple, pero en este caso hay que recordar la jerarquía que sigue Python de izquierda a derecha:



- Invocar a super() y el nombre del método: super().<NOMBRE_DEL_METODO>. **En este caso, Python buscará el método o el atributo de izquierda a derecha según la jerarquía de herencias fue establecida. En caso de no encontrarlo en la primera, buscará en la segunda, tercera, y así sucesivamente hasta que lo encuentre.**
- Invocar al nombre de la clase y el nombre del método:<NOMBRE_CLASE>. <NOMBRE_DEL_METODO>.



Polimorfismo

La herencia también facilita el **polimorfismo**, que es la capacidad de los objetos de diferentes clases derivadas de una misma clase base para responder de manera diferente a los mismos mensajes o métodos.

El polimorfismo en Python es un concepto de programación orientada a objetos que permite que un objeto pueda presentar diferentes formas o comportamientos según el contexto en el que se llame.

En Python, el polimorfismo se puede lograr mediante el uso de herencia y la implementación de métodos con el mismo nombre en diferentes clases. Estas clases pueden heredar de una clase base común y proporcionar su propia implementación del método. A esto se le conoce como "sobrescritura o sobrecarga de métodos".

Lo veremos con ejemplos prácticos en la clase :)

