**Object Oriented Programming**

*Pandas*

*import pandas as pd*

*1º class DataFrame():*

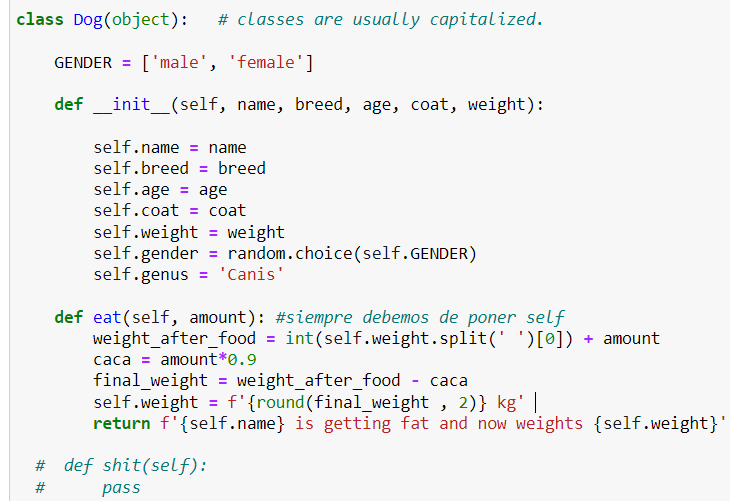
* *atributos → los datos*
* *methods → las acciones*

*2º lo podemos almacenamos en objetos (variables)*

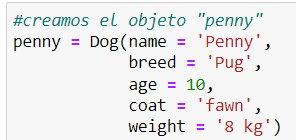
*→ var\_df = pd.DataFrame(x)*

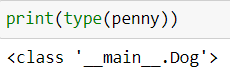
ORDEN:

1. Creamos la CLASE: **class Nombre(object):**
   1. crear **atributos locales** para la clase (opcional)
   2. creamos **\_\_init\_\_**→**def** **\_\_init\_\_(self,** argumentos**) *→ MÉTODO CONSTRUCTOR***
      1. **definimos** elconjunto de ARGUMENTOS
      2. **definimos** el conjunto de ATRIBUTOS (estáticos, dinámicos o lógicos(con una función o method))
   3. creamos resto de **FUNCIONES()** dentro de la clase:

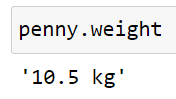


1. Creamos el OBJETO **(instanciamos la clase)**, dando un nombre a la clase y valores a sus ARGUMENTOS (hay que dar valor a TODOS los arg del \_\_init\_\_, sino PETA, ya que es una función)

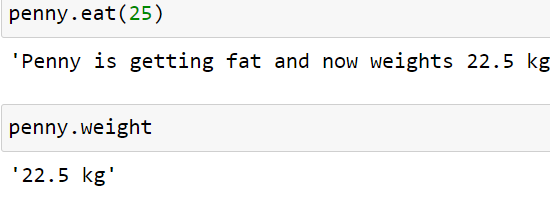


***\* su type es un obj llamado “Dog” (el nombre de la clase)***

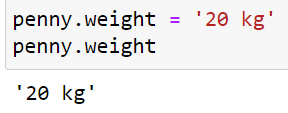
1. EJECUTAMOSfunciones del objeto o simplemente atributos del objeto
2. **llamamos** a un atributo existente del objeto: **nombre\_objeto.atributo**



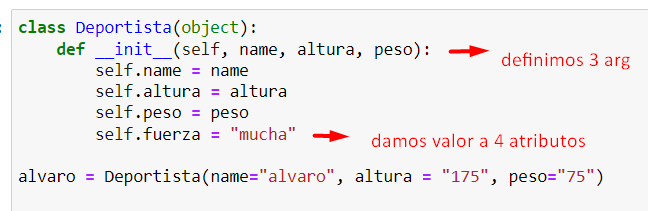
1. **ejecutamos una función del objeto nombre\_objeto.funcion(**args**)** (de esta manera ALTERAMOS el objeto y los valores de sus atributos

****

**→ de esta misma manera podemos cambiar manualmente el valor atributos**

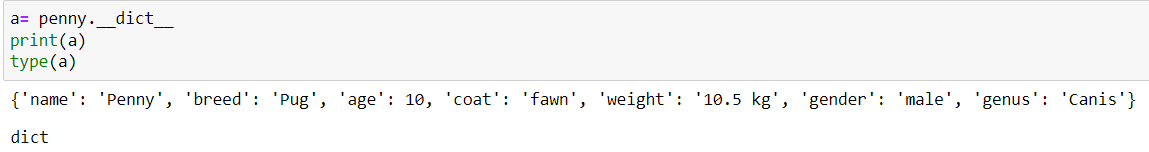
****

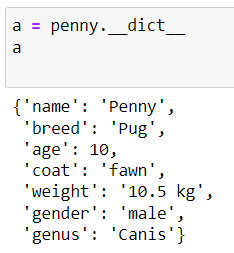
*OTRO EJEMPLO*

****

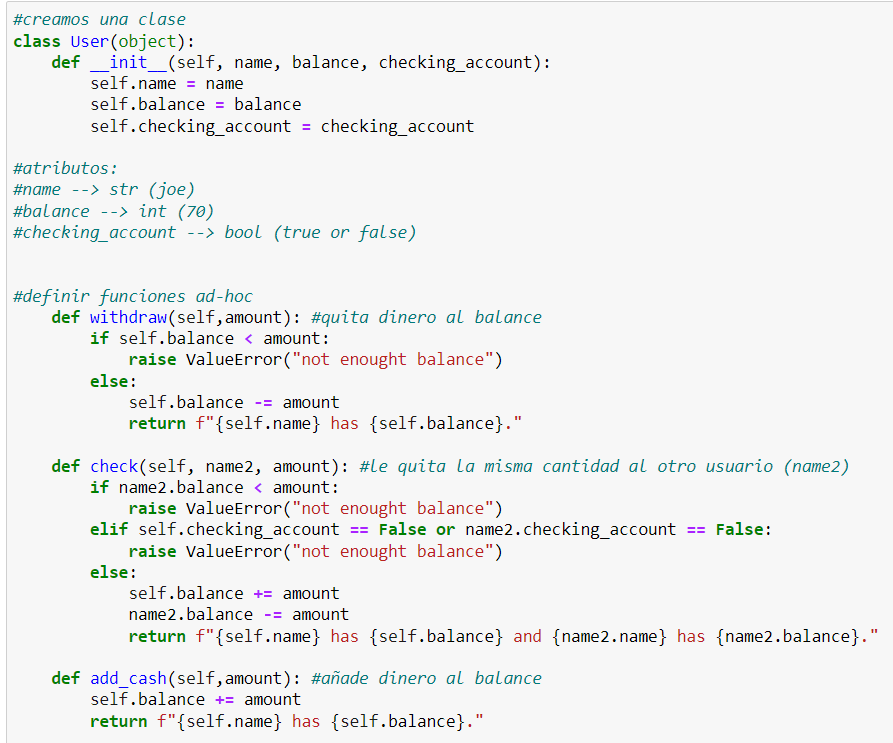
**\* METHODS que se pueden aplicar a estas clases:**

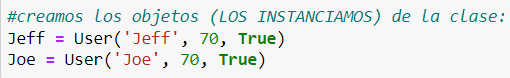
nombre\_objecto**\_\_dict\_\_** → te crea un DIC (key, value) (que realmente es un JSON, (semi-estructurado))



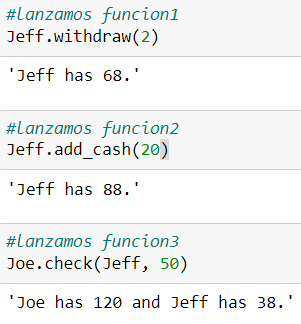


**KATA 4**

* PASO 1: **Creamos la CLASE (\_\_init\_\_ + args** | **atributos** |demás **funciones)**
* PASO 2: **Creamos el OBJETO/S** (los instanciamos con un **nombre** (jeff = …) y **valores** para sus argumentos ClassName(“jeff”, 60, True))



* PASO 3: **Ejecutamos funciones de cada objeto/instancia**

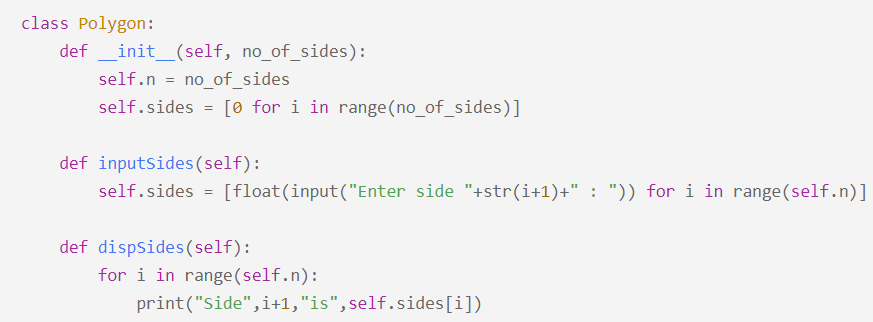
**-->** Joe es **self** y jeff es **name2**

**INHERITANCE**

→ Crear una nueva clase que asumirá los **atributos** y **funciones** de otra clase.

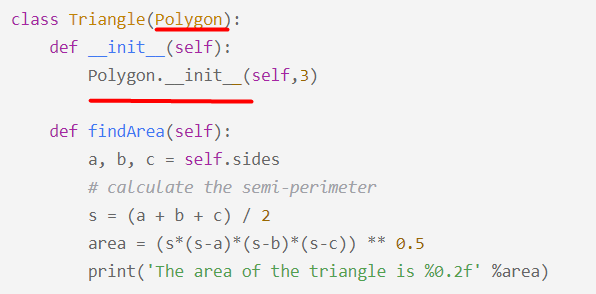
MÉTODO 1: el método de la clase derivada reemplaza al de la clase base (tomando preferencia la nueva sobre la antigua)

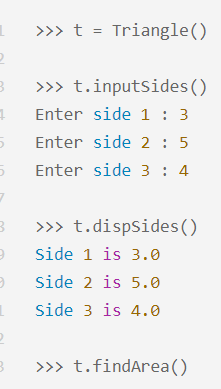
* CLASE 1: *Polygon*



* CLASE 2 “heredada”: *Triangle* → hereda los atributos y funciones

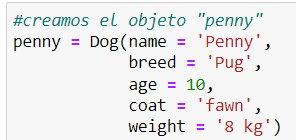
1. Añadiendo la clase antigua dentro de la nueva **NewClass(OldClass)**
2. **OldClass.\_\_init\_\_(self,** args**)** para definir los atributos heredados



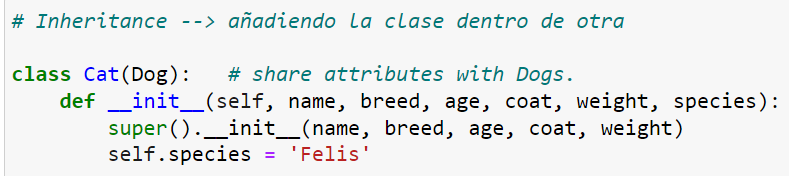


MÉTODO 2: En vez de tomar preferencia, equivale una a la otra.

* CLASE 1: *Dog*



* CLASE 2 “heredada”: *Cat*  → hereda los atributos y funciones



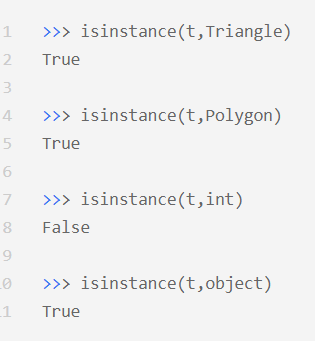
→ hereda los atributos (o forma de crearlos y darles valor) de la clase “superior”:

1. Añadiendo la clase antigua dentro de la nueva **NewClass(OldClass)**
2. utilizando el method **super()** a la hora de definir el conjunto de atributos heredados



**METHODS isinstance() y issubclass()**

* **isinstance(**Objeto**,**Clase**)** → devuelve True si el objeto es una instancia de la clase u otras clases derivadas de él.



* **issubclass(**Clase heredada**,** Clase Superior**)** → se usa para verificar la herencia de clases.

