No puedo llamar a la función privada de una clase que no es la main (la que ejecuto).

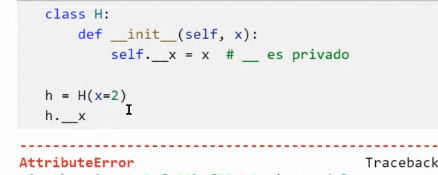
Si llamo a una funcion públics, y ella llama a una función privada de su misma clase, está bien.

Privado solo puedo acceder desde dentro de la propia clase, o sus hijas(si no está en una clase, puedo acceder a ella desde todo el fichero, pero no desde otro). Publicas se pueden usar libremente. Protegidas no pueden ser accedidas desde fuera, pero puedo acceder a todos los q esten dentro del fichero, independientemente de la clase. Cuando hablo de **FICHERO**, me refiero a todo lo que está a la misma altura en mi carpeta, desde la que ejecuto el archivo

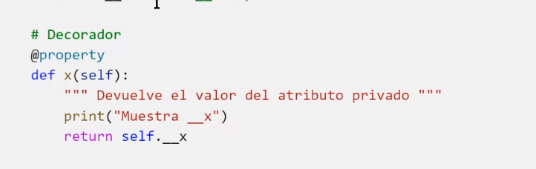
Privada(clase y fichero), protegida(fichero), pública(nada).

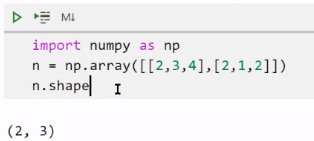
**Propiedades en Python**

Aquí hablaremos de los atributos(propiedades). En las clases, asignamos atributos, a los que podemos llamar para que nos diga el valor q tiene ese atributo en la clase. Ese atributo es modificable, no siempre. Si el atributo es PRIVADO, nos va a dar error, porque intentamos cambiarlo fuera de la clase, y los privados no se pueden tocar fuera de la clase.

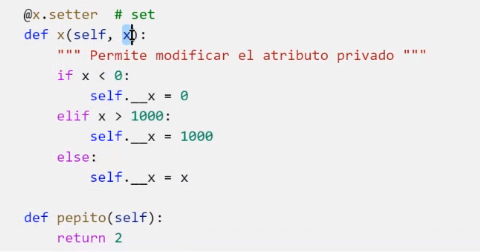


Decorador de función: **@property** (decorador built-in). Cuando vea un @property encima de una función, tengo que entender que al llamar a esa función, no necesitaré poner los parámetros para ejecutarla. Tengo que saber diferenciar entre función y propiedad, porque son cosas muy distintas. Ya que una función hará una acción, y la propiedad me mostrará lo que hace un atributo.



Ejemplo: en el siguiente caso, utilizo shape. Shape, como yo lo conozco, es una función que muestra un array reformado, con shape() para definir sus nuevas dimensiones. No obstante, aquí lo hago sin el paréntesis, es decir, sin parámetros. Con esto, tengo que saber que shape no actúa como función, sino como propiedad. 

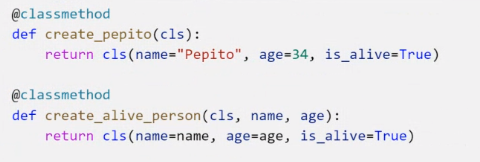
Con **setter**, puedo añadir parámetros a una propiedad.



Cuando trabajo con **setter o con property**?: Cuando trabajo con un atributo que no modifico, uso property. Si uso un atributo que quiero poder modificar, uso setter.



Un **classmethod** es una función que puede construir una nueva instancia. Podemos utilizarlo para crear una instancia con distintos atributos de los que presenta el constructor. Pero tiene que contener dentro todos los parámetros del constructor de alguna manera. Ej: age = birthday\_date etc….



Los **staticmethod** son funciones que vamos a poder utilizar sin crear funciones y que utilizan atributos de la clase.

Ejemplo. En el siguiente ejercicio, creo un staticmethod y una función normal. La diferencia entre ellas, es q la static trabaja con la clase. La otra trabaja con los parámetros que ofrece la instancia creada (seeeeeelf). Para llamar a la instancia, siempre utilizamos el (), en este caso Tablero().check\_boats2(number\_of\_boats=10).

