Variables de Clases: Estáticas o de Instancia – Python POO

Índice del contenido

Continuando con las lecciones de **Python orientado a objetos** es el momento de ver **variables de clases e instancia y métodos de clase e instancia**.

Como habíamos visto anteriormente los **atributos y métodos** definen las características del objeto y modifican su estado. Son datos que están asociados a ellas. De ellos se desprenden dos tipos de variables, **de clase o de instancia (objeto).**

Antes de aprender esto me gustaría repasar lo siguiente:

- **Las clases** agrupan objetos con propiedades y características en común, por ejemplo (Perros).
- **Instancia** significa crear un objeto a partir de una clase. (Un perro en concreto).
- Los atributos y métodos son elementos **que almacenan datos** (diferenciando los objetos de una misma clase) utilizando ***variables de instancia***.

Las **variables de clase** pertenecen a una clase determinada y son accesibles sólo invocando dicha clase. (Ver más abajo).

Las **variables de instancia** también pertenecen a una clase determinada pero sólo son accesibles mediante la instancia del objeto. (Ver más abajo)

¡Y lo mismo sucede con los métodos!

Veamos con más profundidad y ejemplifiquemos esto.

Variables de clase:

Las variables de clases o estáticas como dijimos más arriba son definidas dentro de una clase, específicamente después de su declaración. Pero nunca dentro de un método, porque eso la convertiría en una variable de instancia.

Sintaxis de variables estáticas o de clase:

Veamos ahora cómo crear una variable de clase. ¡Primeramente, debemos crear una clase por supuesto!.

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.  # pythones.net
4.
5.  class Perros (object):
6.      'Clase para los perros' #Descripción
7.      Collar = True #Variable de clase Estática
8.      def __init__(self, salud, hambre):
9.          self.salud = salud #Variable de Instancia
10.         self.hambre = hambre #Variable de Instancia
```

Puedes ver donde declaramos la variable de clase? **Justo después del encabezado de la clase** y luego de la **Descripción de clase.** Si esto no lo sabías ahora ya lo sabes, es para organizarnos mejor. ¡La creamos utilizando comillas simples!

En este caso la variable del tipo Booleano "collar" es igual a True. Determina que todos nuestros perros tienen collares.

Acceso

Ahora esta variable al ser de tipo "clase", nos permite acceder a ella sin necesidad de instanciar (es decir, de crear un objeto perteneciente a esta clase). Por ende, para acceder a ella basta con colocar la clase y el nombre de la variable:

```
print (Perros.Collar)
```

Resultado:

True

Variables de instancia:

Las variables de instancia son aquellas que se relacionan con una única instancia de una clase.

En el código anterior las variables de instancia son salud y hambre. Porque dependen exclusivamente de la instancia de clase. Es decir que para acceder a ellas debemos si o si instanciar: Crear un objeto de la clase Perros.

Sintaxis de variables de instancia.

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.  # pythones.net
4.
5.  class Perros (object):
6.      'Clase para los perros' #Descripcion
7.      Collar = True #Variable de clase Estática
8.      def __init__(self, salud, hambre):
9.          self.salud = salud #Variable de Instancia
10.         self.hambre = hambre #Variable de Instancia
11.
12.
13. # print (Perros.hambre) #-> No puedes acceder de esta manera porque hambre es una
    variable de instancia
14. #Debes instanciar:
15. Dogo = Perros(100, 50)
16. print(Dogo.hambre)
```

Resultado:

50

Lo mismo sucede con los métodos. Están los métodos de clase o estáticos y los de instancia:

Métodos de clase.

Los métodos de clases son similares a las variables de clase y nos permiten acceder a ellos sin instanciar un objeto. Pero para indicarle a Python que un método es de clase debemos usar un **decorador** que nos permita modificar las funciones utilizando otra función para ampliar su funcionalidad. Estos decoradores los veremos en profundidad en la próxima lección, ahora nos enfocaremos en diferenciar los métodos.

Sintaxis de los métodos de clase:

Para indicarle a Python que se trata de un método de clase debemos utilizar el decorador **@classmethod.**

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.  # pythones.net
4.
5.  class Animal (object):
6.      @classmethod
7.      def correr(self, km):
8.          print ("El animal corre %s kilómetros" % km)                #Aquí estamos
    utilizando format
9.      'Clase para los animales corredores' #Descripción de esta clase
10.
11.
12.  Animal.correr(12)
```

Resultado:

El animal corre 12 kilómetros

Como ves en el ejemplo no necesitamos instanciar para acceder a este método de la clase Animal. Solo llamamos el método mediante la clase, y brindamos los argumentos en este caso los km que corrió el animal.

Sintaxis de los métodos de instancia:

Los métodos de instancia son aquellos que utilizamos normalmente a los que no necesitamos añadir ningún decorador y que dependen estrictamente de que instanciemos para poder llamarlos. Observa:

```
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.  # pythones.net
4.
5.  class Ave (object):
6.      'Clase para las aves' #Descripcion
7.      def __init__(self):
8.          pass
9.
10.     def hablar (self, color):
11.         print ("Soy una jodida ave de color %s" %color) #Usamos format
12.
13.
14.  #Ave.hablar ('verde') #-> No puedes acceder de esta manera porque es un método de
    instancia
15.
16.  #Debes instanciar (crear un objeto a partir de la clase)
17.  Loro = Ave()
18.  Loro.hablar('verde') #Ahora si invocamos al método de instancia
```

Resultado:

Soy una jodida ave de color verde

Sintaxis de los métodos estáticos:

Los métodos estáticos requieren del decorador **@staticmethod** para indicarle a Python que se trata del mismo. Este tipo de método nos permite llamar a una función elegantemente dentro de una clase sin que esté ligada a la clase misma ni a la instancia. Veamos un ejemplo con el mismo código anterior:

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# pythones.net

class Ave (object):
    'Clase para las aves' #Descripción
    def __init__(self):
        pass

    def hablar (self, color):
        print ("Soy una jodida ave de color %s" %color) #Usamos format

    @staticmethod        ### ######decorador staticmethod
    def funcion_volar_ave(tiene_alas, kms): #Definimos el metodo
        #####Condicional####################
        if tiene_alas == True:
            print ("El ave se fue volando %i kilometro" %kms)
        else:
            print ("Este ave no puede volar")


Ave.funcion_volar_ave(True, 1) #Llamamos el método (función) usando la clase
```

Resultado:

El ave se fue volando 1 kilometro

Si le pasamos el argumento False en vez de True imprimirá que no puede volar.

Comprender las variables y métodos de clase nos será beneficioso cuando comencemos a crear nuestros primeros programas o aplicaciones orientados a objetos.