Python - Clases y programación orientada a objetos

Introducción

Una clase es una plantilla para crear objetos (como los planos para construir una casa).

> Una clase define variables y todas las distintas funcionalidades asociadas al objeto que describen.
> Se trata de un objeto creado mediante la instrucción *class* y son la base de la Programación Orientada a Objetos (OOP).
> Se recomienda definirlas utilizando etiquetas que comienzen por una letra mayúscula.

La definición más básica de una clase es la siguiente.

```
PS> python.exe
>>>> class Foo:
...    pass
...
>>>> Foo
<class '__main__.Foo'>
```

Las clases se crean en tiempo de ejecución y el código contenido dentro de ella se ejecuta inmediatamente en el momento de su definición.

```
PS> python.exe
>>> class Foo:
...     print('python')
...     import sys
...     print(sys.platform)
...
python
win32
>>>
```

Las clases tienen una serie de atributos denominados **atributos especiales** o **atributos mágicos**, que empiezan y acaban con dos guiones bajos.

```
# __name__ : Se le asigna el nombre de la etiqueta utilizada en la definición de la clase.
>>> Foo.__name__
'Foo'
# __module__ : Se refiere al módulo en el cual se define la clase.
>>> Foo.__module__
'__main__'
>>>
```

Las clases se utilizan para definir tipos de datos.

> Los tipos de datos vistos hasta ahora *int*, *float*, *complex*, *str*, *list*, *tuple*, *dict* y *set* son clases.

Los objetos creados mediante una clase se denominan **instancias de clase**.

> Podemos conocer el tipo de un objeto tanto mediante su atributo *__class__* como mediante la clase integrada *type*.

```
>>> lista1 = list()
>>> lista1.__class__
<class 'list'>
>>> type(lista1)
<class 'list'>
>>>
```

> También podemos utilizar la función integrada *isinstance()* para saber si un objeto es una instancia de una clase.

```
>>> isinstance(lista1, list)
True
>>> isinstance(lista1, int)
False
>>>
```

El parámetro *self*

En Python casi todo es un objeto, incluso una clase es también un objeto.

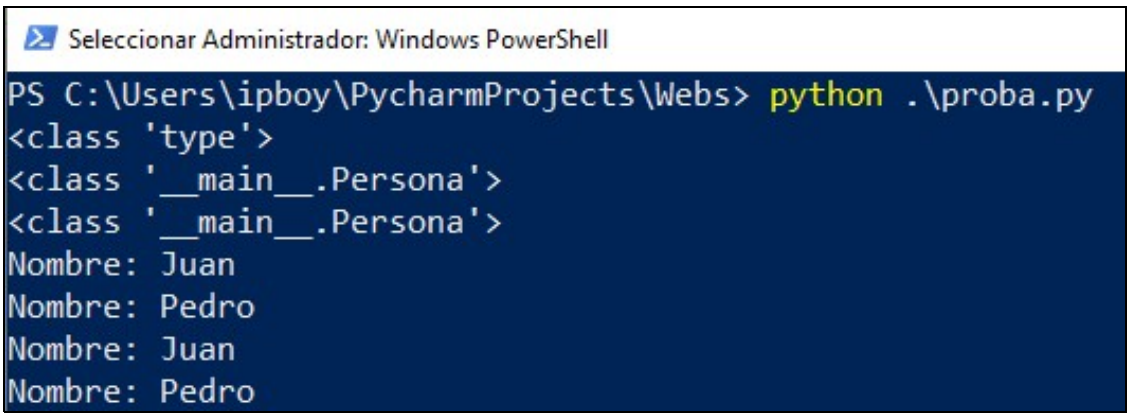
Lo podemos ver en el siguiente ejemplo, donde vemos que Persona es del tipo *<class "type">* y el objeto instanciado es del tipo *<class "Persona">*.

```
#Definimos una clase Persona
class Persona:
    def __init__(self,nom):
        self.nombre=nom

    def imprimir(self):
        print("Nombre: {0}".format(self.nombre))

#Creamos dos instancias p1 y p2
p1 = Persona("Juan")
p2 = Persona("Pedro")
#Vemos los tipos de la Clase
print(type(Persona))
#y los de las instancias
print(type(p1))
print(type(p2))
#Llamamos al método 'imprimir()' de la clase Persona
p1.imprimir()
p2.imprimir()
#Y lo llamamos de otro modo
Persona.imprimir(p1)
Persona.imprimir(p2)
```

Ahora se ve claro que el parámetro *self* es necesario para hacer referencia a la propia instancia del objeto.



Atributos de clase y de instancia

En cuanto a su ámbito de visibilidad, los atributos pueden ser de clase o de instancia.

Las asignaciones realizadas fuera de los métodos, son atributos de clase y son compartidos por todas las instancias.
Los métodos también son atributos de clase.
Las asignaciones realizadas a etiquetas calificadas mediante la instancia crean atributos de instancia.
Si dentro de un método queremos crear un atributo de instancia, necesitamos calificar el atributo mediante el parámetro que hace referencia a la instancia:

```
#Definimos la Clase Cosa
class Cosa:
    a = 33 # Atributo de clase
    def cosa_metodo(self, x):
        self.b = x
#### Fin Clase Cosa

# Creamos las instancias f y g
f = Cosa()
g = Cosa()
print(hasattr(Cosa, 'b'), hasattr(f, 'b'), hasattr(g, 'b'))
## Salida:
```

```
# (False, False, False)

f.cosa_metodo('ffffff')
print(hasattr(Cosa, 'b'), hasattr(f, 'b'), hasattr(g, 'b'))
## Salida:
# (False, True, False)
# Vemos que 'b' es un atributo de instancia
print(f.b)
## Salida:
# ffffff
```

Del mismo modo, si queremos que un método se refiera a un atributo de instancia, necesitamos calificar el atributo con la referencia a la instancia:

```
#Definimos la Clase Cosa
class Cosa:
    def metodoA(self, valor):
        self.valor = valor
    def metodoB(self):
        print(self.valor)
#### Fin Clase Cosa

# Creamos la instancia a
a = Cosa()

a.metodoA('Manuel')
a.metodoB()
## Salida:
# Manuel
```

Referencia principal: "El gran libro de Python" - Capítulo 5 - Marco Buttu.

-- Volver