Aprende a programar con Minecraft

### **Atributos de clase**

A veces puedes querer establecer atributos que tengan el mismo valor para cada instancia de objeto de una clase. Sería redundante pasar el mismo argumento a la clase cada vez que se crea un objeto. En su lugar, puedes crear un atributo preestablecido en la clase, y todas las instancias de objetos de esa clase compartirán esos atributos.

Cuando varios objetos comparten el mismo atributo, se denomina *atributo de* clase. Por ejemplo, todos los objetos gato que hemos creado pertenecen a Craig (yo). Puedo volver a visitar la clase Gato en el archivo *catClass.py*, crear un atributo de clase llamado propietario y establecerlo como "Craig":

*catClass.py*

class Cat(object):  
owner = "Craig"  
  
def \_\_init\_\_(self, name, weight):  
self.name = name  
self.weight = weight

Como puedes ver, los atributos de clase no utilizan self antes de su nombre. En este ejemplo, owner es un atributo de clase y self.name es un atributo. Observa que defines los atributos de clase fuera de la función \_\_init\_\_().

Los atributos de clase funcionan igual que cualquier otro atributo de un objeto. Por ejemplo, puedes acceder al valor de un atributo de clase como lo harías con un atributo normal. En este caso, para encontrar al propietario de Fluff, podemos imprimir el atributo de clase propietario del objeto Fluff:

*catClass.py*

fluff = Cat("Fluff", 4.5)  
print(fluff.owner)

El valor impreso debería ser "Craig". Si imprimiéramos el propietario de Stella, el valor sería el mismo porque los atributos de clase son iguales para todos los objetos de esa clase:

*catClass.py*

stella = Cat("Stella", 3.9)  
print(stella.owner)

El valor impreso aquí también es " Craig".

Puedes cambiar el valor de los atributos de clase para objetos individuales. Esto cambiará el valor del atributo para ese objeto, pero no para otros objetos de la clase. Por ejemplo, Stella ha sido adoptada por mi amigo Matthew, así que tenemos que cambiar el propietario de Stella a "Matthew":

*catClass.py*

stella.owner = "Matthew"  
print(stella.owner)  
print(fluff.owner)

Cuando se imprime el atributo propietario de stella, aparece "Matthew", pero el propietario de fluffsigue siendo "Craig".

Después de todos los cambios que hemos hecho en *catClass.py*, el programa final tiene el siguiente aspecto. También está disponible en los recursos del libro en [*https://www.nostarch.com/programwithminecraft/.*](https://www.nostarch.com/programwithminecraft/)

*catClass.py*

class Cat(object):  
owner = "Craig"  
  
def \_\_init\_\_(self, name, weight):  
self.name = name  
self.weight = weight  
  
def eat(self, food):  
self.weight = self.weight + 0.05  
print(self.name + " is eating " + food)  
  
def eatAndSleep(self, food):  
self.eat(food)  
print(self.name + " is now sleeping...")  
  
def getWeightInGrams(self):  
return self.weight \* 1000  
  
fluff = Cat("Fluff", 4.5)  
print(fluff.owner)  
stella = Cat("Stella", 3.9)  
print(stella.owner)  
  
print(fluff.weight)  
fluff.eat("tuna")  
fluff.eatAndSleep("tuna")  
  
print(fluff.getWeightInGrams())  
print(fluff.name)  
print(stella.name)  
  
fluff.eat("tuna")  
stella.eat("cake")  
  
stella.owner = "Matthew"  
print(stella.owner)  
print(fluff.owner)

Ahora que ya has visto cómo utilizar objetos, vamos a ver cómo hacerlos aún más potentes con la herencia.

[anterior](ch12_6.html)[Subtema 7 de 10: (Ver todo)](ch12.html)[siguiente](ch12_8.html)