### Crear una instancia a partir de una clase

Piensa en una clase como un conjunto de instrucciones sobre cómo crear una instancia. La clase Dog es un conjunto de instrucciones que indica a Python cómo crear instancias individuales que representen a perros concretos.

Hagamos una instancia que represente a un perro concreto:

class Dog:  
 --snip--  
  
❶ my\_dog = Dog('Willie', 6)  
  
❷ print(f"My dog's name is {my\_dog.name}.")  
❸ print(f"My dog is {my\_dog.age} years old.")

La clase Dog que utilizamos aquí es la que acabamos de escribir en el ejemplo anterior. Aquí, le decimos a Python que cree un perro cuyo nombre sea 'Willie' y cuya edad sea 6 ❶. Cuando Python lee esta línea, llama al método \_\_init\_\_() en Dog con los argumentos 'Willie' y 6. El método \_\_init\_\_() crea una instancia que representa a este perro concreto y establece los atributos name y age utilizando los valores que le hemos proporcionado. Python devuelve entonces una instancia que representa a este perro. Asignamos esa instancia a la variable my\_dog. La convención de nomenclatura es útil aquí; normalmente podemos asumir que un nombre en mayúsculas como Dog se refiere a una clase, y un nombre en minúsculas como my\_dog se refiere a una única instancia creada a partir de una clase.

#### Acceder a los atributos

Para acceder a los atributos de una instancia, se utiliza la notación con puntos. Accedemos al valor del atributo my\_dog' name ❷ escribiendo:

my\_dog.name

La notación con puntos se utiliza a menudo en Python. Esta sintaxis muestra cómo Python encuentra el valor de un atributo. Aquí, Python busca en la instancia my\_dog y luego encuentra el atributo name asociado a my\_dog. Se trata del mismo atributo denominado self.name en la clase Dog. Utilizamos el mismo enfoque para trabajar con el atributo age ❸.

El resultado es un resumen de lo que sabemos sobre my\_dog:

My dog's name is Willie.  
My dog is 6 years old.

#### Llamada a métodos

Después de crear una instancia de la clase Dog, podemos utilizar la notación punto para llamar a cualquier método definido en Dog. Hagamos que nuestro perro se siente y se dé la vuelta:

class Dog:  
 --snip--  
  
my\_dog = Dog('Willie', 6)  
my\_dog.sit()  
my\_dog.roll\_over()

Para llamar a un método, indica el nombre de la instancia (en este caso, my\_dog) y el método al que quieres llamar, separados por un punto. Cuando Python lee my\_dog.sit(), busca el método sit() en la clase Dog y ejecuta ese código. Python interpreta la línea my\_dog.roll\_over() del mismo modo.

Ahora Willie hace lo que le decimos:

Willie is now sitting.  
Willie rolled over!

Esta sintaxis es bastante útil. Cuando a los atributos y métodos se les han dado nombres adecuadamente descriptivos como name, age, sit(), y roll\_over(), podemos deducir fácilmente lo que se supone que debe hacer un bloque de código, incluso uno que nunca hemos visto antes.

#### Crear varias instancias

Puedes crear tantas instancias de una clase como necesites. Vamos a crear un segundo perro llamado your\_dog:

class Dog:  
 --snip--  
  
my\_dog = Dog('Willie', 6)  
your\_dog = Dog('Lucy', 3)  
  
print(f"My dog's name is {my\_dog.name}.")  
print(f"My dog is {my\_dog.age} years old.")  
my\_dog.sit()  
  
print(f"\nYour dog's name is {your\_dog.name}.")  
print(f"Your dog is {your\_dog.age} years old.")  
your\_dog.sit()

En este ejemplo creamos un perro llamado Willie y un perro llamado Lucy. Cada perro es una instancia independiente con su propio conjunto de atributos, capaz del mismo conjunto de acciones:

My dog's name is Willie.  
My dog is 6 years old.  
Willie is now sitting.  
  
Your dog's name is Lucy.  
Your dog is 3 years old.  
Lucy is now sitting.

Aunque utilizáramos el mismo nombre y la misma edad para el segundo perro, Python seguiría creando una instancia independiente a partir de la clase Dog. Puedes crear tantas instancias de una clase como necesites, siempre que des a cada instancia un nombre de variable único o que ocupe un lugar único en una lista o diccionario.

[anterior](c09_4.html)[Subtema 5 de 34: (Ver todo)](c09.html)[siguiente](c09_6.html)