### Establecer un valor por defecto para un atributo

Cuando se crea una instancia, se pueden definir atributos sin pasarlos como parámetros. Estos atributos pueden definirse en el método \_\_init\_\_(), donde se les asigna un valor por defecto.

Vamos a añadir un atributo llamado odometer\_reading que comience siempre con el valor 0. También añadiremos un método read\_odometer() que nos ayude a leer el cuentakilómetros de cada coche:

class Car:  
  
 def \_\_init\_\_(self, make, model, year):  
 """Initialize attributes to describe a car."""  
 self.make = make  
 self.model = model  
 self.year = year  
❶ self.odometer\_reading = 0  
  
 def get\_descriptive\_name(self):  
 --snip--  
  
❷ def read\_odometer(self):  
 """Print a statement showing the car's mileage."""  
 print(f"This car has {self.odometer\_reading} miles on it.")  
  
my\_new\_car = Car('audi', 'a4', 2024)  
print(my\_new\_car.get\_descriptive\_name())  
my\_new\_car.read\_odometer()

Esta vez, cuando Python llama al método \_\_init\_\_() para crear una nueva instancia, almacena los valores de marca, modelo y año como atributos, como hizo en el ejemplo anterior. A continuación, Python crea un nuevo atributo llamado odometer\_reading y establece su valor inicial en 0 ❶. También tenemos un nuevo método llamado read\_odometer() ❷ que facilita la lectura del kilometraje de un coche.

Nuestro coche empieza con un kilometraje de 0:

2024 Audi A4  
This car has 0 miles on it.

No se venden muchos coches con exactamente 0 kilómetros en el cuentakilómetros, así que necesitamos una forma de cambiar el valor de este atributo.

[anterior](c09_8.html)[Subtema 9 de 34: (Ver todo)](c09.html)[siguiente](c09_10.html)