### Modificar los valores de los atributos

Puedes cambiar el valor de un atributo de tres formas: puedes cambiar el valor directamente a través de una instancia, establecer el valor a través de un método o incrementar el valor (añadirle una determinada cantidad) a través de un método. Veamos cada uno de estos enfoques.

#### Modificar directamente el valor de un atributo

La forma más sencilla de modificar el valor de un atributo es acceder a él directamente a través de una instancia. Aquí fijamos directamente la lectura del cuentakilómetros en 23:

class Car:  
 --snip--  
  
my\_new\_car = Car('audi', 'a4', 2024)  
print(my\_new\_car.get\_descriptive\_name())  
  
my\_new\_car.odometer\_reading = 23  
my\_new\_car.read\_odometer()

Utilizamos la notación con puntos para acceder al atributo odometer\_reading del coche, y establecemos su valor directamente. Esta línea le dice a Python que tome la instancia my\_new\_car, busque el atributo odometer\_reading asociado a ella y establezca el valor de ese atributo en 23:

2024 Audi A4  
This car has 23 miles on it.

A veces querrás acceder directamente a los atributos de esta forma, pero otras veces querrás escribir un método que actualice el valor por ti.

#### Modificar el valor de un atributo mediante un método

Puede ser útil disponer de métodos que actualicen determinados atributos por ti. En lugar de acceder directamente al atributo, pasas el nuevo valor a un método que se encarga internamente de la actualización.

Aquí tienes un ejemplo de un método llamado update\_odometer():

class Car:  
 --snip--  
  
 def update\_odometer(self, mileage):  
 """Set the odometer reading to the given value."""  
 self.odometer\_reading = mileage  
  
my\_new\_car = Car('audi', 'a4', 2024)  
print(my\_new\_car.get\_descriptive\_name())  
  
❶ my\_new\_car.update\_odometer(23)  
my\_new\_car.read\_odometer()

La única modificación de Car es la adición de update\_odometer(). Este método toma un valor de kilometraje y lo asigna a self.odometer\_reading. Utilizando la instancia my\_new\_car, llamamos a update\_odometer() con 23 como argumento ❶. Esto establece la lectura del cuentakilómetros en 23, y read\_odometer() imprime la lectura:

2024 Audi A4  
This car has 23 miles on it.

Podemos ampliar el método update\_odometer() para que realice un trabajo adicional cada vez que se modifique la lectura del cuentakilómetros. Añadamos un poco de lógica para asegurarnos de que nadie intenta hacer retroceder la lectura del cuentakilómetros:

class Car:  
 --snip--  
  
 def update\_odometer(self, mileage):  
 """  
 Set the odometer reading to the given value.  
 Reject the change if it attempts to roll the odometer back.  
 """  
❶ if mileage >= self.odometer\_reading:  
 self.odometer\_reading = mileage  
 else:  
❷ print("You can't roll back an odometer!")

Ahora update\_odometer() comprueba que la nueva lectura tiene sentido antes de modificar el atributo. Si el valor proporcionado para mileage es mayor o igual que el kilometraje existente, self.odometer\_reading, puedes actualizar la lectura del cuentakilómetros al nuevo kilometraje ❶. Si el nuevo kilometraje es menor que el kilometraje existente, recibirás un aviso de que no puedes retroceder un cuentakilómetros ❷.

#### Incrementar el valor de un atributo mediante un método

A veces querrás incrementar el valor de un atributo en una cantidad determinada, en lugar de establecer un valor completamente nuevo. Supongamos que compramos un coche usado y le hacemos 160 km entre el momento en que lo compramos y el momento en que lo matriculamos. Aquí tenemos un método que nos permite pasar esta cantidad incremental y añadir ese valor a la lectura del cuentakilómetros:

class Car:  
 --snip--  
  
 def update\_odometer(self, mileage):  
 --snip--  
  
 def increment\_odometer(self, miles):  
 """Add the given amount to the odometer reading."""  
 self.odometer\_reading += miles  
  
❶ my\_used\_car = Car('subaru', 'outback', 2019)  
print(my\_used\_car.get\_descriptive\_name())  
  
❷ my\_used\_car.update\_odometer(23\_500)  
my\_used\_car.read\_odometer()  
  
my\_used\_car.increment\_odometer(100)  
my\_used\_car.read\_odometer()

El nuevo método increment\_odometer() toma un número de kilómetros, y añade este valor a self.odometer\_reading. Primero, creamos un coche usado, my\_used\_car ❶. Ponemos su cuentakilómetros en 23.500 llamando a update\_odometer() y pasándole 23\_500 ❷. Por último, llamamos a increment\_odometer() y le pasamos 100 para añadir los 160 km que recorrimos entre la compra del coche y su matriculación:

2019 Subaru Outback  
This car has 23500 miles on it.  
This car has 23600 miles on it.

Puedes modificar este método para que rechace incrementos negativos, de modo que nadie utilice también esta función para hacer retroceder un cuentakilómetros.

[anterior](c09_9.html)[Subtema 10 de 34: (Ver todo)](c09.html)[siguiente](c09_11.html)