### Importar una sola clase

Creemos un módulo que contenga sólo la clase Car. Esto plantea un sutil problema de nomenclatura: ya tenemos un archivo llamado *car.py* en este capítulo, pero este módulo debería llamarse *car.py* porque contiene código que representa un coche. Resolveremos este problema de nomenclatura almacenando la clase Car en un módulo llamado *car.py*, en sustitución del archivo *car.py* que utilizábamos anteriormente. A partir de ahora, cualquier programa que utilice este módulo necesitará un nombre de archivo más específico, como *my\_car.py*. Aquí tienes *car.py* con sólo el código de la clase Car:

**car.py**

❶ """A class that can be used to represent a car."""  
  
class Car:  
 """A simple attempt to represent a car."""  
  
 def \_\_init\_\_(self, make, model, year):  
 """Initialize attributes to describe a car."""  
 self.make = make  
 self.model = model  
 self.year = year  
 self.odometer\_reading = 0  
  
 def get\_descriptive\_name(self):  
 """Return a neatly formatted descriptive name."""  
 long\_name = f"{self.year} {self.make} {self.model}"  
 return long\_name.title()  
  
 def read\_odometer(self):  
 """Print a statement showing the car's mileage."""  
 print(f"This car has {self.odometer\_reading} miles on it.")  
  
 def update\_odometer(self, mileage):  
 """  
 Set the odometer reading to the given value.  
 Reject the change if it attempts to roll the odometer back.  
 """  
 if mileage >= self.odometer\_reading:  
 self.odometer\_reading = mileage  
 else:  
 print("You can't roll back an odometer!")  
  
 def increment\_odometer(self, miles):  
 """Add the given amount to the odometer reading."""  
 self.odometer\_reading += miles

Incluimos un docstring a nivel de módulo que describe brevemente el contenido de este módulo ❶. Debes escribir un docstring para cada módulo que crees.

Ahora creamos un archivo independiente llamado *my\_car.py*. Este archivo importará la clase Car y luego creará una instancia a partir de esa clase:

**my\_car.py**

❶ from car import Car  
  
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()

La sentencia import ❶ indica a Python que abra el módulo car e importe la clase Car. Ahora podemos utilizar la clase Car como si estuviera definida en este archivo. El resultado es el mismo que vimos antes:

2024 Audi A4  
This car has 23 miles on it.

Importar clases es una forma eficaz de programar. Imagina lo largo que sería este archivo de programa si se incluyera toda la clase Car. En cambio, si trasladas la clase a un módulo e importas el módulo, sigues obteniendo toda la misma funcionalidad, pero mantienes tu archivo de programa principal limpio y fácil de leer. También almacenas la mayor parte de la lógica en archivos separados; una vez que tus clases funcionen como quieres, puedes dejar esos archivos tranquilos y centrarte en la lógica de alto nivel de tu programa principal.

[anterior](c09_20.html)[Subtema 21 de 34: (Ver todo)](c09.html)[siguiente](c09_22.html)