### Almacenar varias clases en un módulo

Puedes almacenar tantas clases como necesites en un solo módulo, aunque cada clase de un módulo debe estar relacionada de alguna manera. Las clases Battery y ElectricCar ayudan a representar coches, así que añadámoslas al módulo *car.py*.

**car.py**

"""A set of classes used to represent gas and electric cars."""  
  
class Car:  
 --snip--  
  
class Battery:  
 """A simple attempt to model a battery for an electric car."""  
  
 def \_\_init\_\_(self, battery\_size=40):  
 """Initialize the battery's attributes."""  
 self.battery\_size = battery\_size  
  
 def describe\_battery(self):  
 """Print a statement describing the battery size."""  
 print(f"This car has a {self.battery\_size}-kWh battery.")  
  
 def get\_range(self):  
 """Print a statement about the range this battery provides."""  
 if self.battery\_size == 40:  
 range = 150  
 elif self.battery\_size == 65:  
 range = 225  
  
 print(f"This car can go about {range} miles on a full charge.")  
  
class ElectricCar(Car):  
 """Models aspects of a car, specific to electric vehicles."""  
  
 def \_\_init\_\_(self, make, model, year):  
 """  
 Initialize attributes of the parent class.  
 Then initialize attributes specific to an electric car.  
 """  
 super().\_\_init\_\_(make, model, year)  
 self.battery = Battery()

Ahora podemos crear un nuevo archivo llamado *my\_electric\_car.py*, importar la clase ElectricCar y crear un coche eléctrico:

**my\_electric\_car.py**

from car import ElectricCar  
  
my\_leaf = ElectricCar('nissan', 'leaf', 2024)  
print(my\_leaf.get\_descriptive\_name())  
my\_leaf.battery.describe\_battery()  
my\_leaf.battery.get\_range()

Esto tiene el mismo resultado que vimos antes, aunque la mayor parte de la lógica esté oculta en un módulo:

2024 Nissan Leaf  
This car has a 40-kWh battery.  
This car can go about 150 miles on a full charge.

[anterior](c09_21.html)[Subtema 22 de 34: (Ver todo)](c09.html)[siguiente](c09_23.html)