### Instancias como atributos

Al modelar algo del mundo real en código, puedes encontrarte con que vas añadiendo cada vez más detalles a una clase. Te darás cuenta de que tienes una lista cada vez mayor de atributos y métodos y que tus archivos son cada vez más largos. En estas situaciones, puedes reconocer que parte de una clase puede escribirse como una clase independiente. Puedes dividir tu clase grande en clases más pequeñas que funcionen juntas; este enfoque se denomina *composition*.

Por ejemplo, si seguimos añadiendo detalles a la clase ElectricCar, podríamos darnos cuenta de que estamos añadiendo muchos atributos y métodos específicos de la batería del coche. Cuando veamos que esto ocurre, podemos detenernos y trasladar esos atributos y métodos a una clase independiente llamada Battery. Entonces podremos utilizar una instancia de Battery como atributo en la clase ElectricCar:

class Car:  
 --snip--  
  
class Battery:  
 """A simple attempt to model a battery for an electric car."""  
  
❶ def \_\_init\_\_(self, battery\_size=40):  
 """Initialize the battery's attributes."""  
 self.battery\_size = battery\_size  
  
❷ def describe\_battery(self):  
 """Print a statement describing the battery size."""  
 print(f"This car has a {self.battery\_size}-kWh battery.")  
  
  
class ElectricCar(Car):  
 """Represent 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)  
❸ self.battery = Battery()  
  
my\_leaf = ElectricCar('nissan', 'leaf', 2024)  
print(my\_leaf.get\_descriptive\_name())  
my\_leaf.battery.describe\_battery()

Definimos una nueva clase llamada Battery que no hereda de ninguna otra clase. El método \_\_init\_\_() ❶ tiene un parámetro, battery\_size, además de self. Se trata de un parámetro opcional que establece el tamaño de la pila en 40 si no se proporciona ningún valor. También se ha trasladado a esta clase el método describe\_battery() ❷.

En la clase ElectricCar, añadimos ahora un atributo llamado self.battery ❸. Esta línea le dice a Python que cree una nueva instancia de Battery (con un tamaño por defecto de 40, porque no estamos especificando un valor) y asigne esa instancia al atributo self.battery. Esto ocurrirá cada vez que se llame al método \_\_init\_\_(); cualquier instancia de ElectricCar tendrá ahora una instancia de Battery creada automáticamente.

Creamos un coche eléctrico y lo asignamos a la variable my\_leaf. Cuando queramos describir la batería, tendremos que trabajar a través del atributo battery del coche:

my\_leaf.battery.describe\_battery()

Esta línea le dice a Python que busque la instancia my\_leaf, encuentre su atributo battery y llame al método describe\_battery() que está asociado a la instancia Battery asignada al atributo.

El resultado es idéntico al que vimos anteriormente:

2024 Nissan Leaf  
This car has a 40-kWh battery.

Esto parece mucho trabajo extra, pero ahora podemos describir la pila con tanto detalle como queramos sin saturar la clase ElectricCar. Añadamos otro método a Battery que informe de la autonomía del coche en función del tamaño de la batería:

class Car:  
 --snip--  
  
class Battery:  
 --snip--  
  
 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):  
 --snip--  
  
my\_leaf = ElectricCar('nissan', 'leaf', 2024)  
print(my\_leaf.get\_descriptive\_name())  
my\_leaf.battery.describe\_battery()  
❶ my\_leaf.battery.get\_range()

El nuevo método get\_range() realiza algunos análisis sencillos. Si la capacidad de la batería es de 40 kWh, get\_range() establece la autonomía en 240 km, y si la capacidad es de 65 kWh, establece la autonomía en 225 km. A continuación, informa de este valor. Cuando queramos utilizar este método, tendremos que llamarlo de nuevo a través del atributo battery ❶ del coche .

La salida nos indica la autonomía del coche en función del tamaño de su batería:

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

[anterior](c09_16.html)[Subtema 17 de 34: (Ver todo)](c09.html)[siguiente](c09_18.html)