Python Constructor

July 13, 2022

We'll now look at a special Python function that is part of most classes: the Python constructor. A constructor is a function that is called automatically when an object is created. A constructor can optionally accept arguments as well, just like a regular function.

The default constructor

When creating an object from a class, it looks like we are calling a function:

```
car = Car()
```

Well… it doesn't just look like we are calling a function, we are in fact calling a function! This method, which we did not have to define, is called the constructor. It constructs and initializes the object. Every class has one by default, called `__init__`, even if we don't define it ourselves. This has to do with inheritance, which you'll learn about shortly.

Have you ever used the `str()` function to convert a number into a string? Or perhaps the `int()` function to convert a string into a number?

```
>>> 'a' + str(1)
'a1'
>>> int('2') + 2
4
```

What you are doing here, is creating new objects of type `str` and `int` by calling the constructors of the classes `str` and `int`.

Creating your own Python constructor

We can override the `__init__` method, to give it extra abilities by accepting arguments. Let's redefine the `Car` class using a custom constructor:

```
class Car:
    def __init__(self, started = False, speed = 0):
        self.started = started
        self.speed = speed
    def start(self):
        self.started = True
        print("Car started, let's ride!")
    def increase_speed(self, delta):
        if self.started:
            self.speed = self.speed + delta
            print("Vrooooom!")
        else:
            print("You need to start the car first")
    def stop(self):
        self.speed = 0
```

Our custom Python constructor has named parameters with default values, so we can create instances of the class `Car` in multiple ways:

```
>>> c1 = Car()
>>> c2 = Car(True)
>>> c3 = Car(True, 50)
>>> c4 = Car(started=True, speed=40)
```

You may have noticed a flaw: we can now create a new car that is not started but set its speed anyway. For now, let's leave it at that.

‹ Previous: Classes and Objects in Python

Next: Python Inheritance