Augmenting Python with Modules

Python modules are separate files that contain classes, functions, and other data that allow us to import and make use of these methods and classes in our own code. Python comes with a lot of modules out of the box. These modules are referred to as the Python Standard Library. You can make use of these modules by using the **import** keyword, followed by the module name. For example, we'll import the **random** module, and then call the **randint** function within this module:

```
>>> import random
>>> random.randint(1,10)
8
>>> random.randint(1,10)
7
>>> random.randint(1,10)
1
```

This function takes two integer parameters and returns a random integer between the values we pass it; in this case, 1 and 10. You might notice that calling functions in a module is very similar to calling methods in a class. We use dot notation here too, with a period between the module and function names.

Let's take a look at another module: **datetime**. This module is super helpful when working with dates and times.

```
>>> import datetime
>>> now = datetime.datetime.now()
>>> type(now)
<class 'datetime.datetime'>
>>> print(now)
2019-04-24 16:54:55.155199
```

First, we import the module. Next, we call the **now()** method which belongs to the **datetime** class contained within the **datetime** module. This method generates an instance of the datetime class for the current date and time. This instance has some methods which we can call:

```
>>> print(now)
2019-04-24 16:54:55.155199
>>> now.year
2019
>>> print(now + datetime.timedelta(days=28))
2019-05-22 16:54:55.155199
```

When we call the print function with an instance of the datetime class, we get the date and time printed in a specific format. This is because the datetime class has a **__str__** method defined which generates the formatted string we see here. We can also directly call attributes and methods of the class, as with **now.year** which returns the year attribute of the instance.

Lastly, we can access other classes contained in the datetime module, like the **timedelta** class. In this example, we’re creating an instance of the timedelta class with the parameter of 28 days. We’re then adding this object to our instance of the datetime class from earlier and printing the result. This has the effect of adding 28 days to our original datetime object.