Aliasing Modules

It is possible to modify the names of modules and their functions within Python by using the `as` keyword.

You may want to change a name because you have already used the same name for something else in your program, another module you have imported also uses that name, or you may want to abbreviate a longer name that you are using a lot.

The construction of this statement looks like this:

```
import [module] as [another_name]
```

Let's modify the name of the `math` module in our `my_math.py` program file. We'll change the module name of `math` to `m` in order to abbreviate it. Our modified program will look like this:

my_math.py

```
import math as m


print(m.pi)
print(m.e)
```

Within the program, we now refer to the `pi` constant as `m.pi` rather than `math.pi`.

For some modules, it is commonplace to use aliases. The `matplotlib.pyplot` module's official documentation calls for use of `plt` as an alias:

```
import matplotlib.pyplot as plt
```

This allows programmers to append the shorter word `plt` to any of the functions available within the module, as in `plt.show()`. You can see this alias import statement in use within our "How to Plot Data in Python 3 Using `matplotlib` tutorial."

Conclusion

When we import modules we're able to call functions that are not built into Python. Some modules are installed as part of Python, and some we will install through `pip`.

Making use of modules allows us to make our programs more robust and powerful as we're leveraging existing code. We can also create our own modules for ourselves and for other programmers to use in future programs.