Dynamic Imports

The importlib module supports the ability to import a module that is passed to it as a string. So let's create a couple of simple modules that we can work with. We will give both modules the same interface, but have them print their names so we can tell the difference between the two. Create two modules with different names such as **foo.py** and **bar.py** and add the following code in each of them:

```
def main():
    print(__name__)
```

Now we just need to use importlib to import them. Let's look at some code to do just that. Make sure that you put this code in the same folder as the two modules you created above.

```
 # importer.py}
import importlib
import foo


def dynamic_import(module):

    return importlib.import_module(module)


if __name__ == '__main__':
    module = dynamic_import('foo')
    module.main()

    module_two = dynamic_import('bar')
    module_two.main()
```

Here we import the handy importlib module and create a really simple function called **dynamic_import**. All this function does is call importlib's

import_module function with the module string that we passed in and returns the result of that call. Then in our conditional statement at the bottom,

we call each module's **main** method, which will dutifully print out the name of the module.

You probably won't be doing this a lot in your own code, but occasionally you'll find yourself wanting to import a module when you only have the module as a string. The importlib module gives us the ability to do just that.