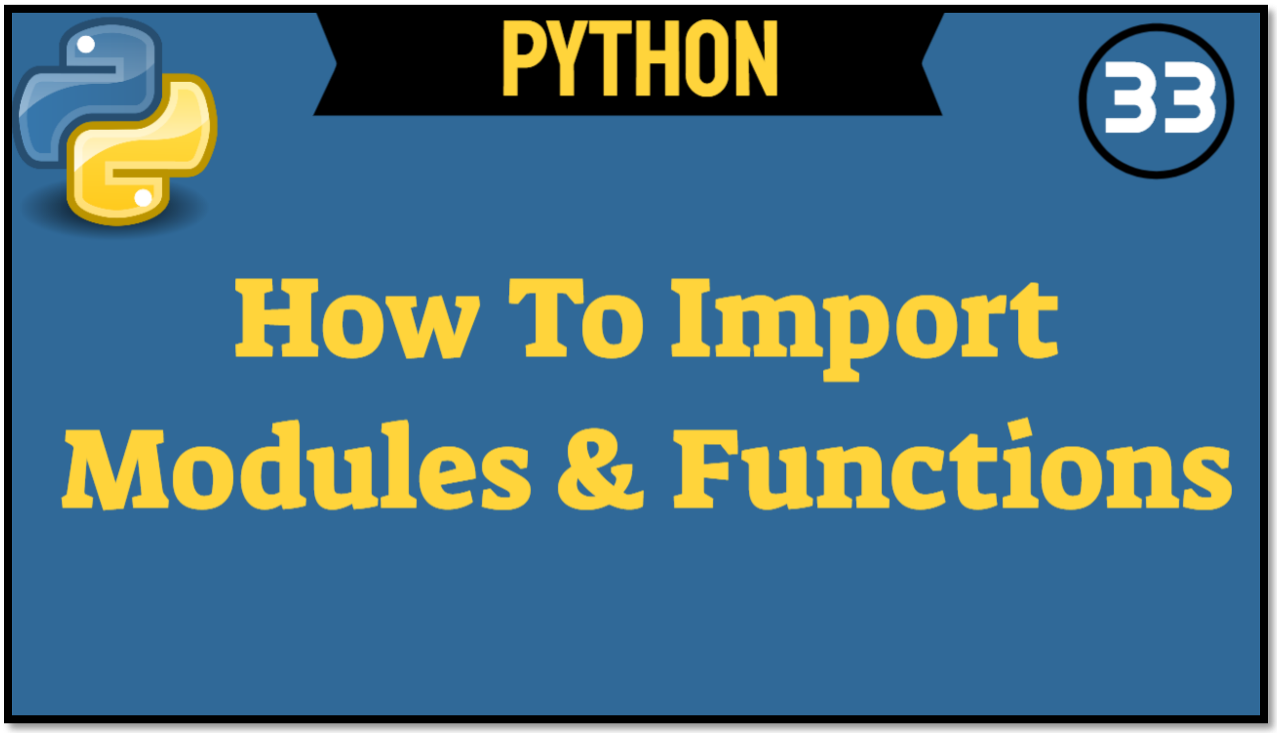
Rex
Jones II

How To Import Python Modules & Functions



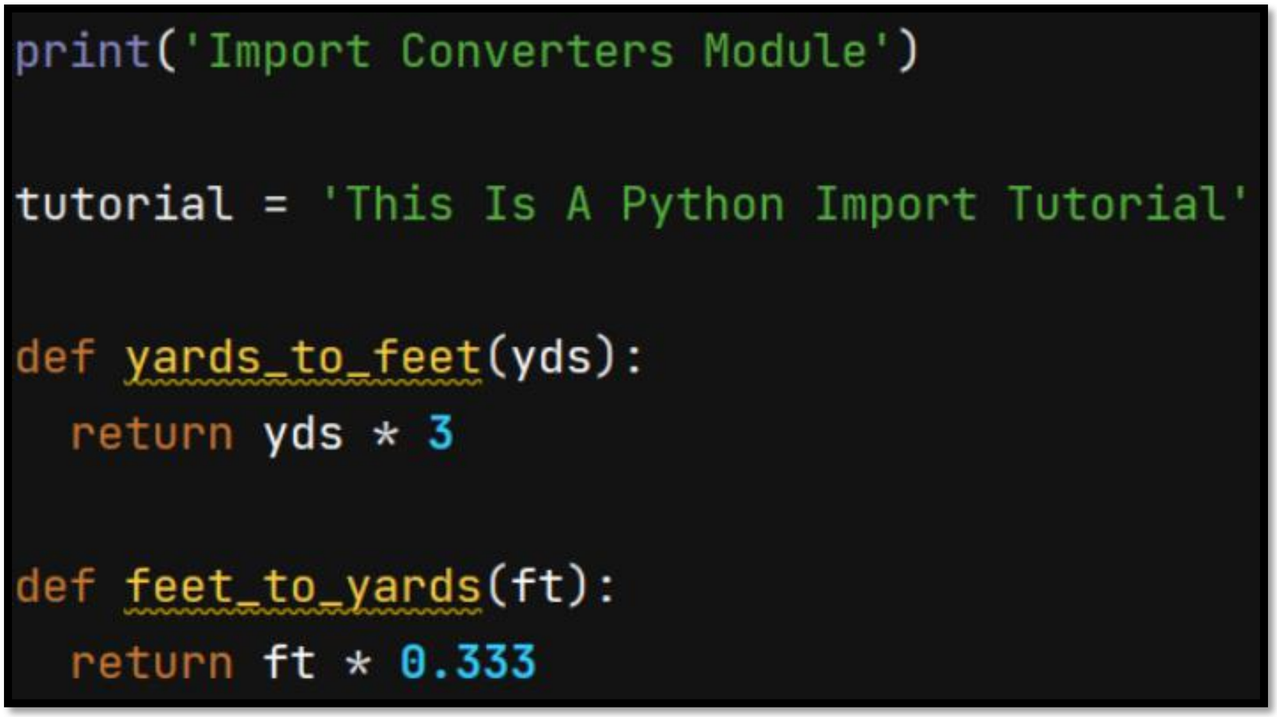
Import Functions, Modules, & Classes

Python Video = https://youtu.be/CEGMzasxvBE

In this session, we are going to focus on importing modules and importing functions. Each module is a Python program file containing related information. It can contain 1 or more functions. The benefit of placing a function inside of a module is to separate blocks of code from our main program. That's a way of dividing our code into logical sections.

We import existing code to our main program or import built-in code provided by Python. There are different ways to import. We can use keyword 'from' and keyword 'import'. If you want the transcript PDF document. I will place it on GitHub. Also, you can follow me on Twitter, connect with me on LinkedIn and Facebook. Also, you can subscribe to my channel. Placing our function in a module separated from our main program allows us to hide the details of the code.

In this module called module_converters, I have a print statement, a variable, and 2 functions. The functions are yards to feet and feet to yards. They are converter functions.



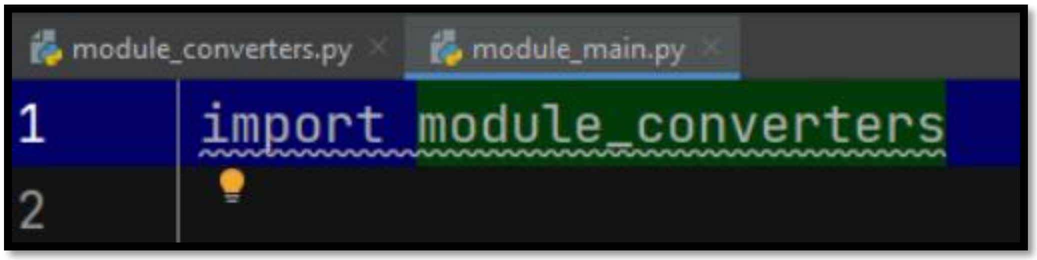
```
print('Import Converters Module')

tutorial = 'This Is A Python Import Tutorial'

def yards_to_feet(yds):
  return yds * 3

def feet_to_yards(ft):
  return ft * 0.333
```

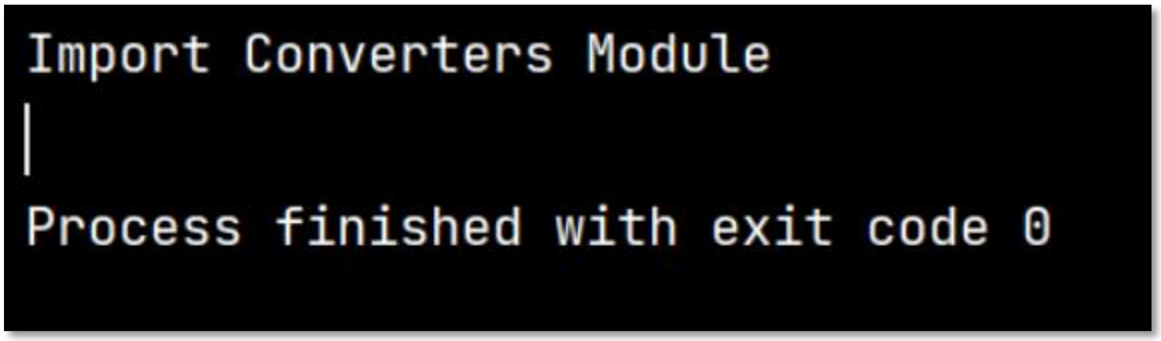
Notice this other module called module_main. It is empty but I'm going to import module_converters. The purpose of import is to use code from another file. When we import, it will execute all of the code including the print statement. Go back to this empty file and write import. We are going to import the module_converters file. Leave off the .py extension.



module_converters.py | module_main.py

```
import module_converters
```

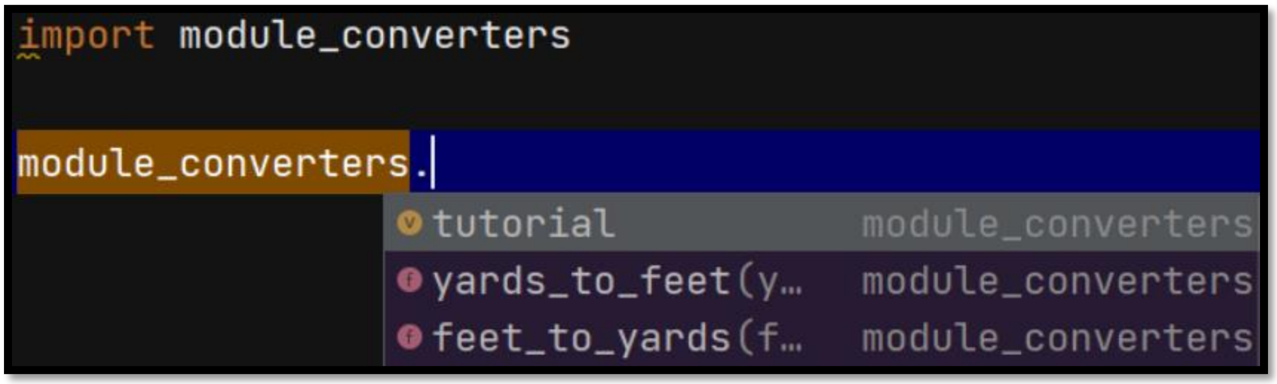
If I run right now, the console shows the print statement Import Converters Module.



```
Import Converters Module

Process finished with exit code 0
```

When Python reads the import statement line, it tells Python to open the module_converters file and copy all the data from module_converters into module_main. To call a function or variable, we can write module_converters then the dot operator. The dot operator provides access to each function and variable. That's why we see tutorial, yards to feet, and feet to yards.

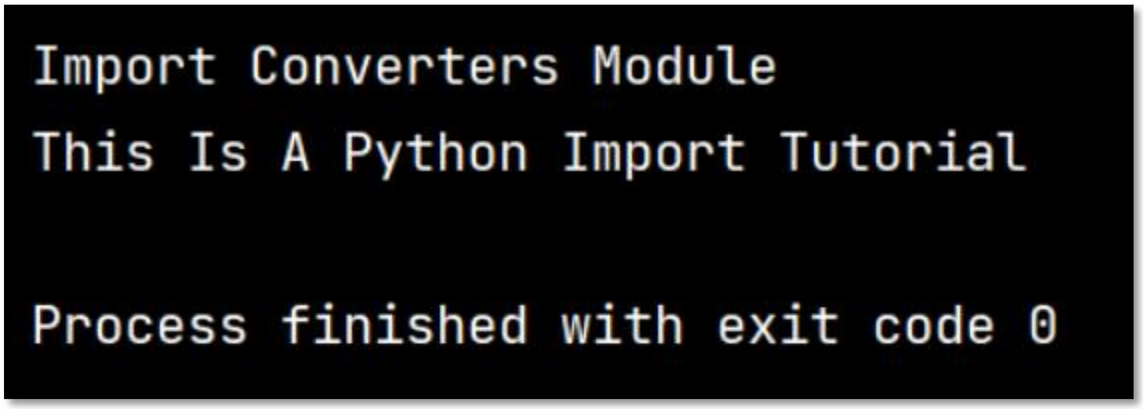


```
import module_converters

module_converters.
```

tutorial module_converters
yards_to_feet(y… module_converters
feet_to_yards(f… module_converters

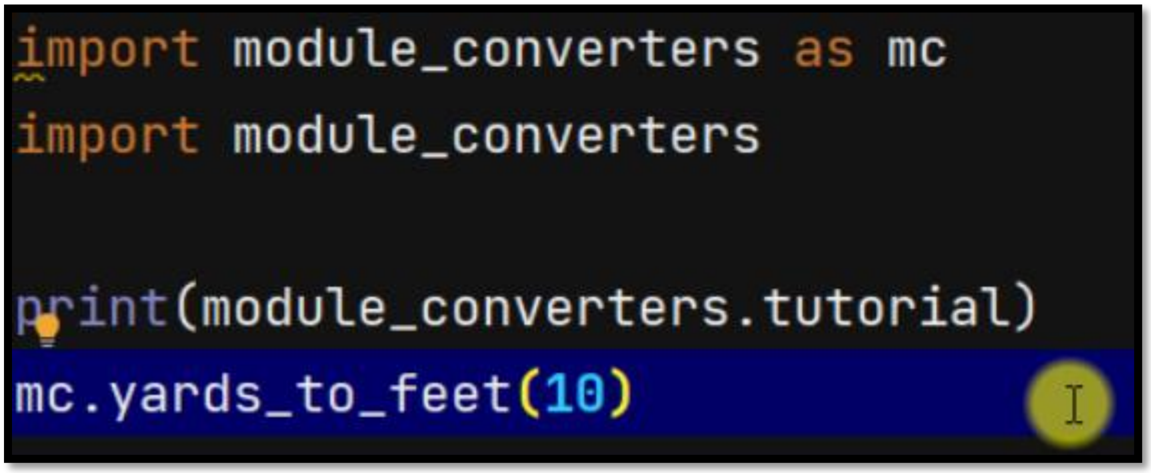
Select tutorial then print the value for tutorial. This time when I run. We see Import Converters Module. This Is A Python Import Tutorial.



```
Import Converters Module
This Is A Python Import Tutorial

Process finished with exit code 0
```

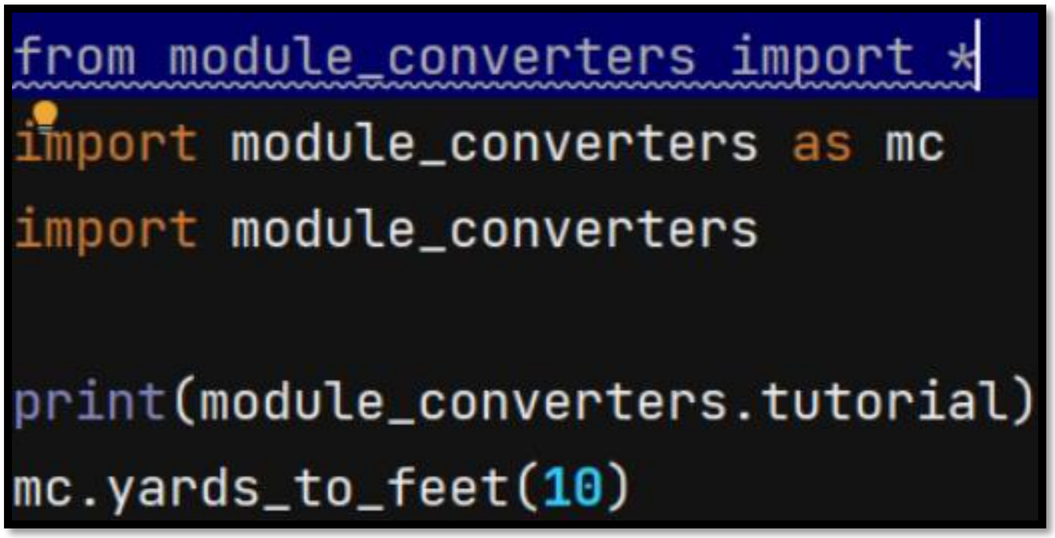
The good part about Python. We can specify a nickname for a module, function, or class. I'm going to start over and write import module_converters as. The keyword as allows us to rename a module, function, or class. Since the module name is long, let's rename module_converters to mc. Now, when I write mc and the dot operator. We still see the variable and both functions. Select yards_to_feet and pass in 10. 10 represents the number of yards that will be converted to the number of feet.



```
import module_converters as mc
import module_converters

print(module_converters.tutorial)
mc.yards_to_feet(10)
```

So far, we see how to import the entire module. These 2 import statements copy all of the data from a module. However, we can also import an individual function or variable by writing from. At this point, we need to write the module name which is module_converters then import. Some people import everything from a module by writing an * (asterisk).

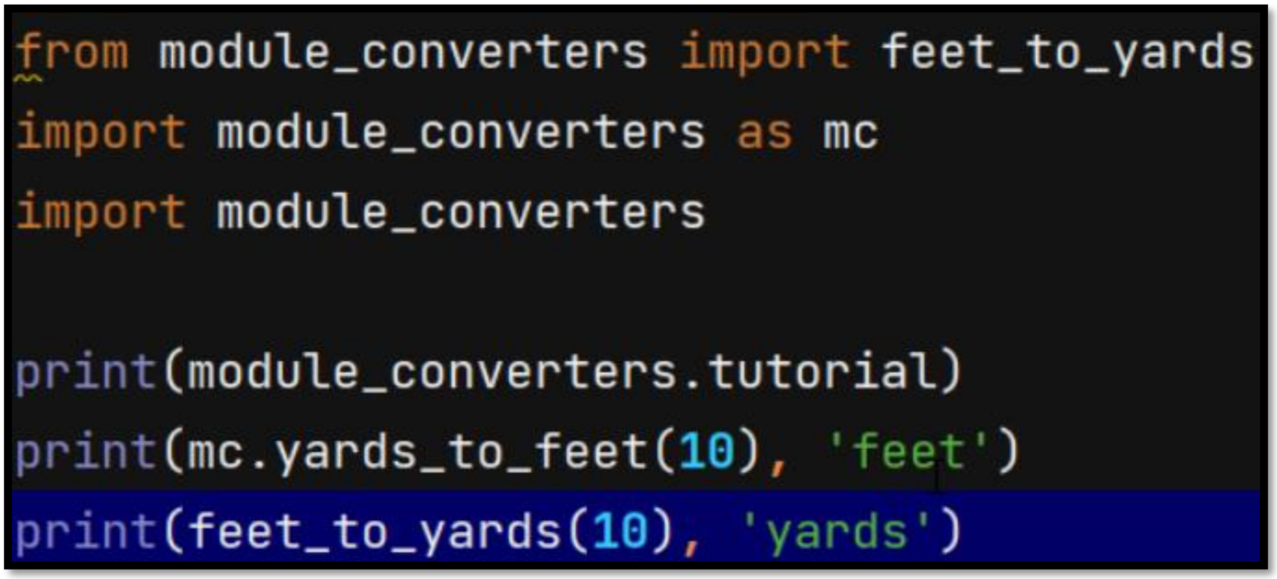


```
from module_converters import *
import module_converters as mc
import module_converters

print(module_converters.tutorial)
mc.yards_to_feet(10)
```

However, this is not recommended because it copies all of the code just like the previous 2 statements. The recommended way is to import 1 function at a time. How about feet_to_yards? Now we do not have to write the module name and dot operator. We can write feet_to_yards then pass in 10.

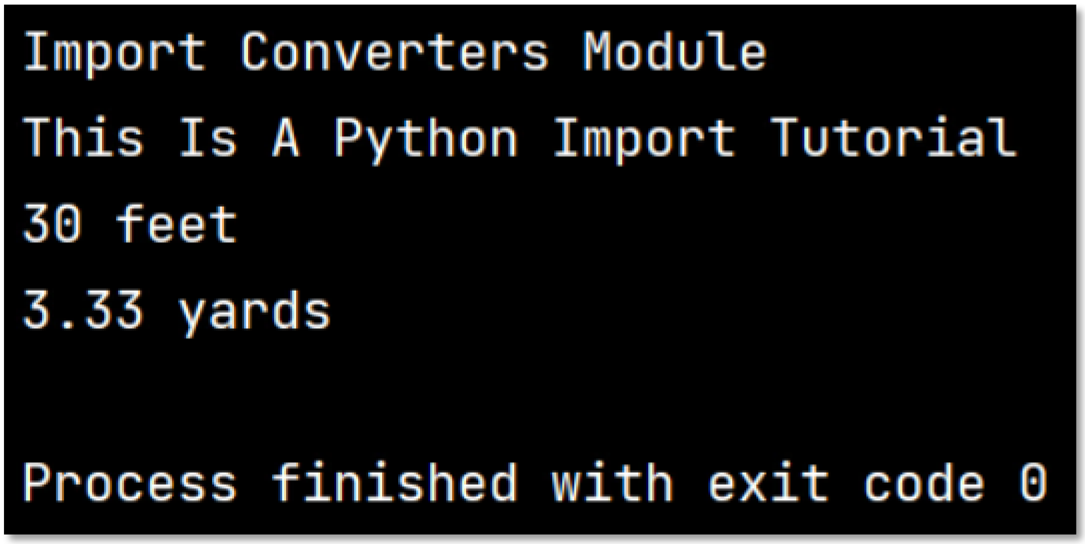
That's all we need because we imported that individual function. Also print the values. The 1st converter function will print() the number of feet by converting by 10 yards to , 'feet' and the 2nd converter function will print() the number of yards by converting 10 feet , 'yards'.



```
from module_converters import feet_to_yards
import module_converters as mc
import module_converters

print(module_converters.tutorial)
print(mc.yards_to_feet(10), 'feet')
print(feet_to_yards(10), 'yards')
```

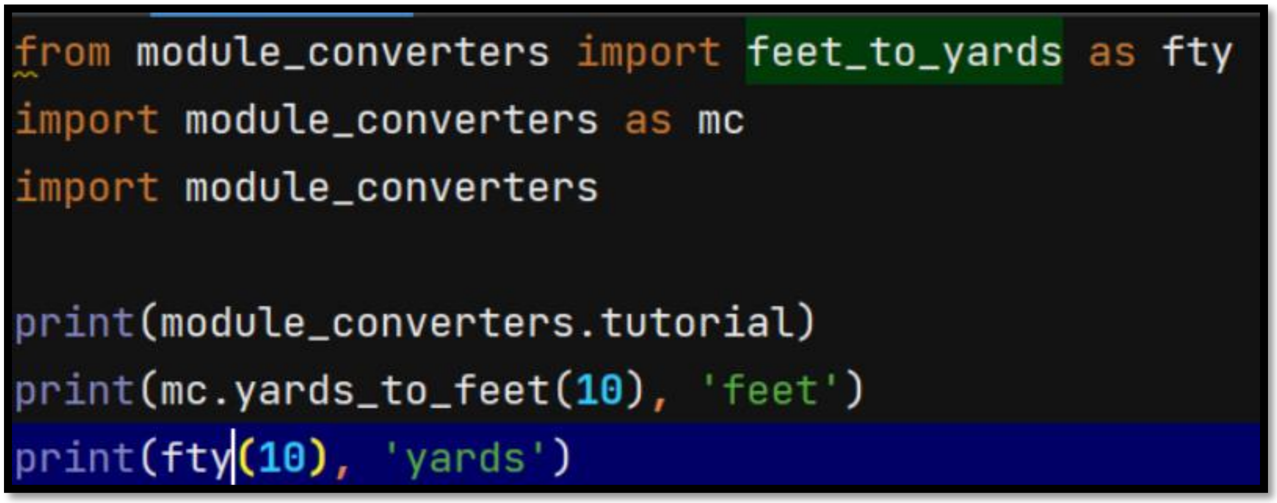
Let’s Run and the console shows 30 feet is the same as 3.33 yards.



```
Import Converters Module
This Is A Python Import Tutorial
30 feet
3.33 yards

Process finished with exit code 0
```

It’s the same. It equals to each other. We can also create an alias for the function by writing as fty then update the function to fty.



```
from module_converters import feet_to_yards as fty
import module_converters as mc
import module_converters

print(module_converters.tutorial)
print(mc.yards_to_feet(10), 'feet')
print(fty(10), 'yards')
```

However, I’m going to remove the nickname because it’s not descriptive and update the function. When it comes to individual functions, we can also add as many as we want by separating each function with a comma, yards to feet. It’s the same if we want to import classes and built-in functions the same way. That’s it for importing in Python.

Contact Info

- ✔ Email Rex.Jones@Test4Success.org
- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/