

# Topic 3 – Modules

Author: Jake Palczewski, CCIE #63320

Last Updated: December 6, 2020

## Table of Contents

<b><i>Introduction .....</i></b>	<b><i>2</i></b>
<b><i>    Import .....</i></b>	<b><i>2</i></b>
<b><i>    External Modules .....</i></b>	<b><i>4</i></b>
<b><i>Conclusion .....</i></b>	<b><i>4</i></b>

## Introduction

The vast array of modules available for Python help make it one of the most flexible coding languages. A module in Python is a collection of software designed to perform a specific purpose. Each module is essentially a .py file that is either built-in or external to Python's Standard Library. External modules can be downloaded from the Internet via pip. This lesson will explore built-in and external Python modules.

## Import

Let's say we want to use Python to find the square root of some integers. Python has a built-in module within the Python Standard Library that makes finding square roots easy! However, if we try to use the `sqrt` attribute to find the square root of 16, for example, we run into an error:

```
3 print(sqrt(16))
4
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```
NameError: name 'sqrt' is not defined
```

The `sqrt` attribute is part of the `math` module. The `math` module is already part of Python's Standard Library when you download and install Python, but to keep Python lean and simple, it is not automatically loaded into Python's main process. To call the `math` module, we use an **import** statement:

```
1 import math
2
3 print(math.sqrt(16))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
4.0
```

In the example above:

- We **import** the `math` module from Python's Standard Library

- It is common practice to define `import` statements at the top of your Python script. This way, modules are loaded as the script begins to run.
- To execute the square root attribute, we use `math.sqrt` in a `print` function.

We can also `import` a module `as` a name of our choosing:

```
1  import math as m
2
3  print(m.sqrt(16))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

4.0

- As you can see, we `import` the `math` module `as` “`m`”
- We, therefore, call the `sqrt` attribute with `m.sqrt` instead of `math.sqrt`.

To avoid needing to call the module and the attribute every time we need to run the attribute, we can use a `from` statement with our `import` statement to import select/all attributes from a module.

```
1  from math import sqrt, factorial
2
3  print(sqrt(16))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TER

4.0

In the example above:

- We use the `from` and `import` statements to `import` the `sqrt` and `factorial` attributes `from` the `math` module
- As a result, in our `print` function, we simply call `sqrt` instead of `math.sqrt`

We can also use an asterisk to `import` all attributes `from` a module, which still allows us to use attributes without defining the module name:

```
1  from math import *
2
3  print(sqrt(16))
4
```

PROBLEMS 54 OUTPUT DEBUG CONSOLE TERMINAL

4.0

In the example above:

- We still only need to call `sqrt` instead of `math.sqrt` to run the `sqrt` attribute.

## External Modules

Python has a vast community of developers, some of which have developed special modules outside of the Python Standard Library. To install external modules, use `pip`. Installed with Python, `pip` is a package manager for Python that efficiently reaches out to PyPI (Python Package Index) to download packages. A package is simply all the files you need to run a module.

```
PS H:\>
PS H:\> pip install netmiko
```

In the example above:

- We use PowerShell with elevated privileges to install the `netmiko` module, which is hosted on PyPI
- Note: if you are installing external modules on your company-owned machine, please be aware of any audit or governance restrictions for Python modules

After executing the `pip` command in PowerShell, we are now able to **import** `netmiko`! Specifically, we will **import** the attribute `ConnectHandler` **from** `netmiko` to prepare for our next lesson.

```
1  from math import sqrt
2  from netmiko import ConnectHandler
```

## Conclusion

Modules add additional functionality to any Python script. Importing them into a script is a matter of ascertaining if the module is built-in or external, and if it's external, using `pip` to install it. The **from** and **import** statements—which should be defined at the top of your script—add the module's functionality to your script. Check out the accompanying **Lesson4-modules.py** file for some practice!

Below is a list of continued reading for modules:

- [Python Standard Library for 3.8](#)
- [PyPI](#)
- [Pip](#)
- [Modules](#)