

MODULES

An Introduction to Computer Science



Let's learn about Modules.

Modules

Modules are Python Programs



Whenever you create a new Python program, you are actually creating a Module.

In other words, a module is just a Python program.

A module is useful, however, because it can be imported into another module.

Any functions, variables, and other definitions created in the module will then be available in the other module.

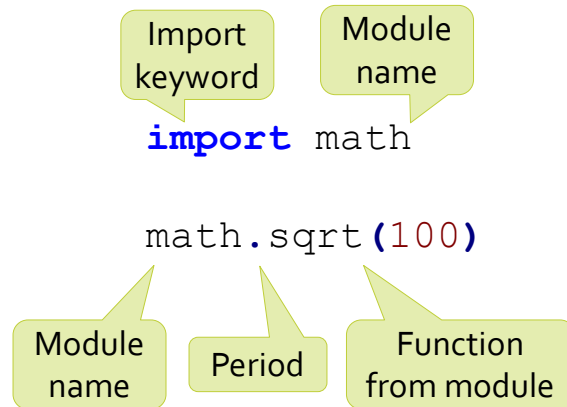
Terminology

- Module
- Library
- Package



Modules are sometimes also known as packages or libraries.
Typically, a package is a collection of one or more modules.
However, the words are all more-or-less interchangeable.

Import



To import data from another module, you use the Import statement.

There are three different ways to import a module.

The simplest is shown here, where we write the keyword `import` and the name of the module.

Once a module is imported, you can reference its definitions using the name of the module, a period, and then the name of the function, variable, or other definition you want to use.

Specific Import

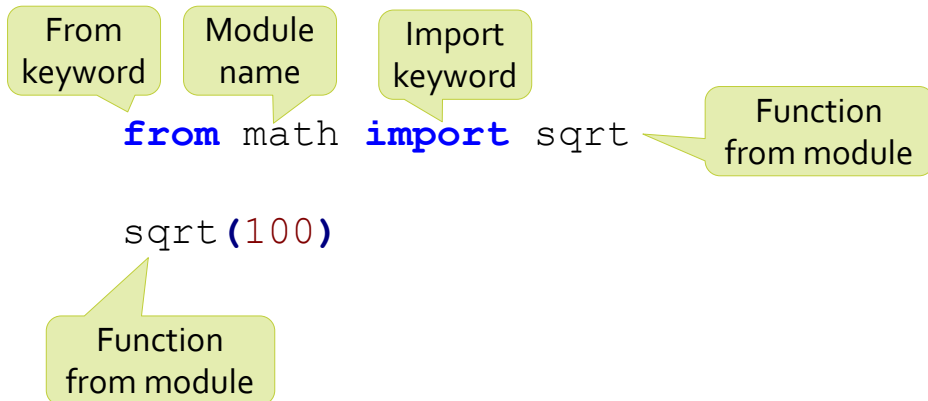
From keyword Module name Import keyword Function from module

```
from math import sqrt
```

Function from module

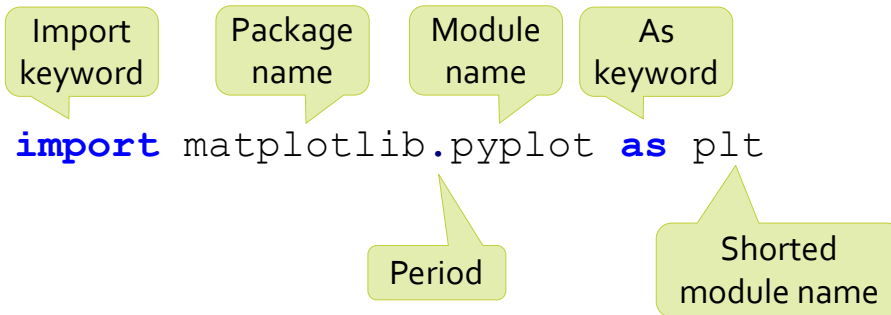
```
sqrt(100)
```

Function from module



It can be bothersome to repeat the module name each time you want to use a function. Therefore, Python has a convenient syntax for importing the functions directly. This version uses the "from" keyword, followed by the module name, the "import" keyword, and then the name of the function or other definition you want to use. Afterwards, you can use the definition as if it were defined in that file.

Longer Imports



Complex packages are often divided into multiple folders containing the modules.

When importing specific parts of these packages, you will need to use periods to indicate folder names.

The result of this import is then aliased (or renamed) using the "as" keyword.

Usually, the documentation will tell you exactly what to import; otherwise it can be difficult to infer the path.

For example, many tutorials for the Matplotlib library (which we will learn later) suggest you import the PyPlot module as shown here.

After this import, the "plt" module is now available.

Standard Library

• array	• codecs	• gc	• marshal	• parser	• stat	• c
• ast	• collections	• heapq	• math	• pickle	• string	• warnings
• atexit	• csv	• imp	• md5	• random	• struct	• weakref
• audioop	• datetime	• io	• mmap	• sha1	• symtable	• winapi
• binascii	• errno	• itertools	• msvcrt	• sha256	• sys	• winreg
• bisect	• faulthandler	• json	• nt	• sha512	• thread	• xxsubtype
• builtins	• locale	• opcode	• signal	• time	• zipimport	
• cmath	• functools	• lsprof	• operator	• sre	• tracemalloc	• zlib



Python has a number of useful built-in modules called the Standard Library.

Further, Python programmers have created many other "third party libraries" that are easy to install.

Many of these modules come prepackaged in distributions like Anaconda.

We will not cover all these modules in this course, but you will often come across them when solving real-world problems.

Installing New Libraries

- `pip install pygame`
- `pip install requests`
- `pip install matplotlib`



Although you may not have to do it very often, it is very easy to install new modules from a command line.

Simply use the "pip" command to install a module.

You will need to know the exact module name.

Make sure that you have the right module name; when you install a new module, you are trusting the developer to not install anything malicious!