Introduction to Modules

by Sophia

WHAT'S COVERED

In this lesson, we'll be looking at modules and how to use them in Python. Specifically, this lesson covers:

1. Modules in Python
2. Importing Modules

1. Modules in Python

The term module is one that we'll hear often. In Unit 1, we termed a module as a self-contained piece of code that can be used in different programs. We have already seen instances where we imported modules in lesson examples.

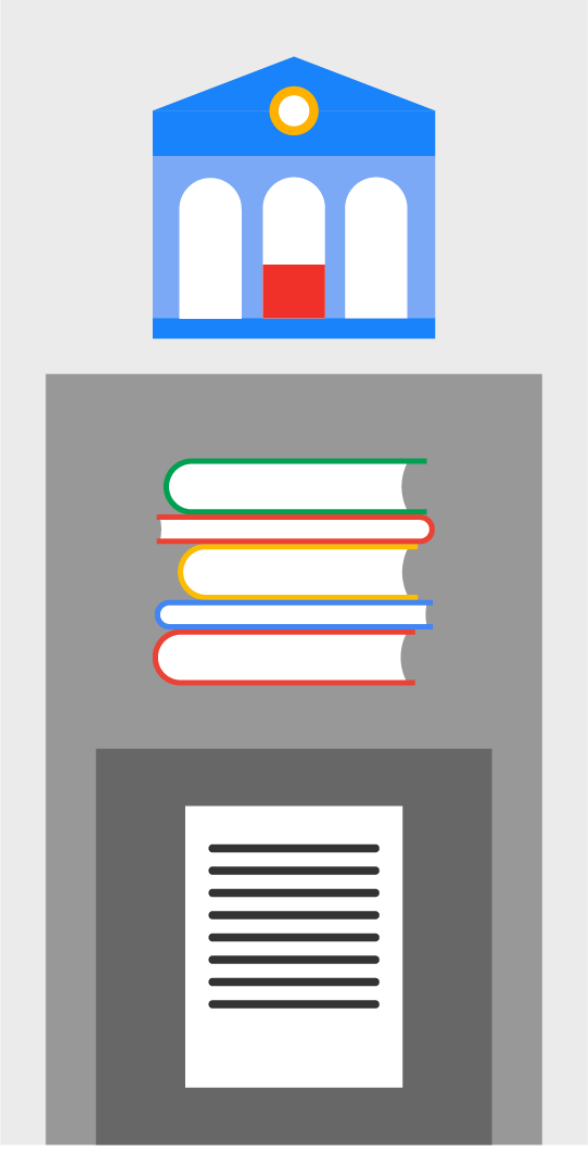
- We imported the `random` module in Unit 1 to use the `random()` function for number generation.
- We imported and used the `math` module in Unit 2 to utilize the `sin()` function.
- And recently in Unit 3, we imported the `datetime` module to get the `date.today()` function in our `Employee` Class program.

For each of these examples, we imported a module so we could make use of a particular function that each of them contained.
Modules can be contained in packages. We will discuss packages more in a minute but think of a package like a book and modules like chapters in that book. In fact, these chapters/modules can be 10 pages or 1000 pages depending on the content. Also, these books/packages can contain one chapter/module or many chapters/modules. Much like the difference between a large novel vs. a simple children's book. And where can you find a bunch of books? At a library or bookstore (physical or digital).

There is a digital library that is distributed with Python. The **Python Standard Library** is composed of collections of built-in modules that provide access to system functionality and standardized solutions that Python programmers can utilize when developing programs. We will discuss more about the Python Standard Library, or just standard library (in reference to Python) later.

THINK ABOUT IT

At a high level, let's envision this "library system" of valuable prebuilt elements. If you think of a standard library as a physical library, a package would be considered a book in that library and a module would be considered a chapter in that book. In essence, a package can contain many modules and a library can contain many different packages.



Library:
Python's Standard Library

Books:
Packages

Chapters:
Modules

The Python Standard Library

With each version release or update to Python, the standard library is distributed with the Python source code and installers. The standard library can be found on Python's site:

docs.python.org/3/library/

TRY IT

Directions: Try visiting the link above to see the Python Standard Library on python.org. See if you can find the modules we have used in the past in datetime and math.

Packages

To help organize modules and provide a naming hierarchy, Python uses the concept of packages. **Packages** are a collection of one or more modules that are typically related in functionality.

Packages can be imported in the same manner as what has been done with modules so far; however, importing packages is less common than importing a module.

Back to Modules

Modules are a big part of what makes Python easy to use. Within them, code is grouped together based on different purposes and functionality.

Import Parts of a Module

We don't have to import everything including the kitchen sink to use just a piece of code like the random number generator that we've used before. We will show how to just import the items we need from a module later.

Import the Whole Module

On the other hand, if we need some related items like functions for dates and times, we don't have to import them one by one. Importing the entire module that contains these functions will get us what we need. We will start on a full module import next.

TERMS TO KNOW

Python Standard Library
The Python Standard Library is composed of collections of built-in modules that provide access to system functionality and standardized solutions that Python programmers can utilize when developing programs.

Packages
Packages are a collection of one or more modules that are typically related in functionality.

---

2. Importing Modules

We can import functionality from modules in a few different ways. The most common is to import the entire module. To do so, we use the **import** reserved keyword which is used to import modules, along with the name of the module that we want to import.

```
import <modulename>
```

For example, the following code imports the entire `math` module.

EXAMPLE

```
import math
```

Directory and Help Functions

After we import a module, we can use the `dir()` and `help()` functions to get more information about a module.

The **dir() function** (directory) returns the content of the object including all properties and methods without the values.

Let's see what the `dir()` function does when we print it out.

EXAMPLE

```
import math

print(dir(math))
```

The output of the `dir()` function looks like this.

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb
```

This may seem complex since it provides us with a full list of the different properties and methods within the math module.

Let's look at the `math` module using the `help()` function.

EXAMPLE

```
import math

help(math)
```

Here is the beginning of that output.

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.
```

```
        The result is between 0 and pi.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

        The result is between -pi/2 and pi/2.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

        The result is between -pi/2 and pi/2.
```

TRY IT

Directions: Try entering the previous code to run the `help()` function on the `math` module.

The `help()` function outputs every single one of the properties and methods that were returned as part of the `dir()` function earlier, but with details including how each of the properties and methods are used and what they are used for. You do not need to use the `dir()` function each time you want to use the `help()` function, but you can use the `dir()` function to get in the ballpark of what you need, then zero in on detailed information with the `help()` function.

Notice the bottom of the output on the math module.

```
   trunc(x, /)
        Truncates the Real x to the nearest Integral toward 0.

        Uses the __trunc__ magic method.

    ulp(x, /)
        Return the value of the least significant bit of the float x.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    (built-in)
```

The `math` module contains properties that contain specific data values. Listed here for example is pi. It is easier to use this value than to create a variable in your program for it.

Importing Parts of a Module

By importing the `math` module, we are getting access to every single one of the properties and methods of that module. However, there are times when we don't need to import everything and the extra data associated with it slows down processing. Ideally, we should import only the items that are necessary. The **from** reserved keyword is used to import only a specified method or property from a module. We can use the following syntax to import only the pi property.

EXAMPLE

```
from math import pi
```

Using the data section that we just identified in the `math` module, we're importing the property pi (seen in the last screenshot). It does not import anything else from the `math` module.

THINK ABOUT IT

Many modules contain values for particular properties much like the pi property of the math module. These properties aid in simplifying programming development. Remember, the scope rules still apply, so always keep in mind where you are using variables and in what level of scope they exist. For example, if we imported pi from the `math` module but redefined pi later in our program, it would defeat the purpose of importing the property of pi in the first place.

After importing pi specifically from the `math` module, we can refer to "pi" simply as pi.

EXAMPLE

```
from math import pi

print(pi)
```

So, just using pi will work with the `print()` function.

```
3.141592653589793
```

When we imported the `math` module, we would not be able to access pi by just simply using the name “pi”. For example, here we are importing the `math` module and trying to print out pi as we just did.

EXAMPLE

```
import math

print(pi)
```

However, we do not get the same results.

```
Traceback (most recent call last):
  File "/home/main.py", line 3, in <module>
    print(pi)
NameError: name 'pi' is not defined
```

We get a *NameError* since we did not reference pi as part of the `math` module.

Rather, we have to prefix pi with the `math` module. We do so with the name of the module and a period (.).

EXAMPLE

```
import math

print(math.pi)
```

Here is that output.

```
3.141592653589793
```

Making Aliases

Sometimes the names of the modules can be lengthy. We can use aliases to make it easier to refer to the module name. For example, we can use the letter “m” instead of math as part of the `import` statement using the **as** keyword. If you remember, “as” is one of Python’s reserved keywords and it is simply used to create an alias. When we use “m”, it replaces it with the `math` module.

EXAMPLE

```
import math as m
print("The value of pi is", m.pi)
```

The output still looks correct using the aliases.

```
The value of pi is 3.141592653589793
```

As you can see, the use of pi is easy once you know it’s available. However, the use of pi is not part of Python’s Standard Library, meaning it’s not just in the library as a “book”. To use pi as the property defined in the `math` module, we always have to import the `math` module or the property within the `math` module first.

Remember, with each release of Python, the entire standard library is available for you to reference. There are many modules contained in the standard library including many built-in functions, methods, and property values for us to use. The standard library can always be found on Python's site.

docs.python.org/3/library/

Importing Multiple Items

We have the option to import multiple items from a module by listing out their names separated by a comma. Suppose we want to use pi and the sqrt (square root) function. The code would look like the following.

EXAMPLE

```
from math import pi, sqrt
```

You may also wonder if we can import everything from a module and use the names directly. To do that, we can use the * (asterisk) character.

EXAMPLE

```
from math import *
```

By doing that, we can access all of the properties and methods without having to prefix it with the word `math`.

EXAMPLE

```
from math import *

print(pi)
```

Just using the word pi now gets the desired output.

```
3.141592653589793
```

The * (asterisk) character is short for “everything”, so the command is the same as “import math”, which also imports the entire `math` module but with a subtle difference. That difference allows us to use those items without prefixing it with the module name.

Just a note, however, that using the * (asterisk) character when importing is not always a good practice. If you happen to be importing a lot of different modules with different pieces from each, it can get confusing since there may be instances where modules will have the same properties and methods between them. Always check the standard library before you start to program, so you know which modules contain which properties and methods you may need to use in your program.

TERMS TO KNOW

import
The `import` reserved keyword is used to import modules.

dir()
The `dir()` (directory) function returns the content of the object including all properties and methods without the values.

from
The `from` reserved keyword is used to import only a specified method or property from a module.

SUMMARY

In this lesson, we learned about **modules in Python**. We discovered that modules contain a variety of properties and methods that we can use in our programs. Modules can be grouped into packages; however, they are all contained in the Python Standard Library, which is always available. We saw that using the `dir()` and `help()` functions can help us identify what properties and methods each module contains. We learned how to **import an entire module** or just a part of it.

Best of luck in your learning!

TERMS TO KNOW

Packages
Packages are a collection of one or more modules that are typically related in functionality.

Python Standard Library
The Python Standard Library is composed of collections of built-in modules that provide access to system functionality and standardized solutions that Python programmers can utilize when developing programs.

dir()
The `dir()` (directory) function returns the content of the object including all properties and methods without the values.

from
The `from` reserved keyword is used to import only a specified method or property from a module.

import
The `import` reserved keyword is used to import modules.