Modules and Packages

Lesson Objectives

After this lesson, you will be able to…

- Add packages and modules to a Python program.
- Create a program utilizing PyTime.
- Navigate library documentation.

Discussion: Let's Dive In

Do you remember `itertools`?

What do you think these have in common?

- `itertools`.
- A Python function that tells us when Mother's Day is.
- A Python function that gets all the contents of a webpage for us.

What is a Module?

Answer: They're all available to us via modules! (In fact, `itertools` IS a module).

Modules are collections of useful Python code and functions that we can use.

- This is much like a class that someone else has written.
- It's free - less work for us!

Use a function by `import <module>` at the top of your code, then `<module>.function_you_want()`.

```
# import < module name > - brings in the module file, so we can use it.
import itertools


food = ['pizza', 'tacos', 'sushi']

colors = ['red', 'green']


# itertools.chain : "Look in the itertools module, and use the chain functic
chained_list = list(itertools.chain(food, colors))
```

Pro Tip: Check the Additional Reading at the end of the lesson to see how to write your own module!

Python Standard Library

We're going to look at several different modules to get you used to them.

The Python Standard Library bundles all common modules, so we can just `import` (use) them.

We've seen the `itertools` module already. Let's look at another module, `random`:

```
import random

# Done! Now we can use any functions in the random module!


# randint is a function in the random module
my_random_number = random.randint(2,8)


# This could be 2, 3, 4, 5, 6, 7, or 8
print(my_random_number)
```

run ▶

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
```

We do: Exploring the Random Module

How do we know:

- What `randint` does?
- What the `random` module has?

Every module has documentation, which has:

- What functions are in the module.
- How to use them.

Here is the documentation for the random module.

- Can you find our `randint` function?

documentation

run ▶

Not sure what to do? Run some examples (start typing to dismiss)

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
```

Quick Review

Modules are collections of useful Python code and functions that we can use.

Use a function by `import <module>` at the top of your code, then `<module>.function_you_want()`.

We've looked at two modules: `itertools` and `random`.

```
# Import statements go at the top of your file
import itertools
import random


# Using the randint function in the random module
my_random_number = random.randint(2,8)
print(my_random_number)


# Using the chain function in the intertools module
food = ['pizza', 'tacos', 'sushi']
colors = ['red', 'green']
```

The Python Standard Library bundles all common modules, so we can just `import` (use) them.

```
people_in_lottery = ["Tina", "Batu", "Gina", "Jim", "Andres"]

lottery_winner = random.choice(people_in_lottery)

print(lottery_winner, "wins a new car!") # Gina
```

run ▶

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

Partner Exercise: Let's Get Random!

Get with a partner and create a local file, `random_test.py`

- Generate a random number with `random.randrange()`. Print it out.
- Create a list, like `deck = ["ten", "jack", "queen", "king", "ace"]`
- Use `random.choice()` to pick a random card in your deck. Print it out.
- Use `random.shuffle()` to mix up your deck; print that out.

Here is the documentation for the random module, so you can look up functions.

run ▶

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
```

Quick Review

Modules are collections of useful Python code and functions that we can use.

Use a function by `import <module>` at the top of your code, then `<module>.function_you_want()`.

The Python Standard Library bundles all common modules, so we can just `import` (use) them. `itertools` and `random` are modules inside the Python Standard Library.

```
# Import statements go at the top of your file - they import straight from t
import itertools
import random


# Using the randint function in the random module
my_random_number = random.randint(2,8)
print(my_random_number)


# Using the chain function in the intertools module
food = ['pizza', 'tacos', 'sushi']
colors = ['red', 'green']
```

What is a Package?

A package, also called a **library**, is a place where one or more related modules are stored.

- In technical terms, *one or more* modules bundled together under a single namespace.
- A package is like a folder, while a module is like a file.

The Python Standard Library bundles all common modules - it's the package with `itertools` and `random` modules inside it.

All packages are modules, but not all modules are packages.

We Do: ModuleNotFoundError

The Python Standard Library has a huge list of modules. But not every Python module in the world is part of it!

`pytime` is a non-standard module. PyTime can:

- Get dates, date ranges, and times.
- Find the date of a particular holiday.

Create a new file called `pytime_test.py`. Put this line in it and run it:

```
import pytime
```

What's happened?

`ModuleNotFoundError`:

- The module isn't part of the standard library.
- If we want to use modules from other packages, we'll have to tell Python that those packages exist.

Including PyTime

When importing from the standard library, the package is implied:

```
# (from standard) import MODULE

import random
```

Otherwise, you need to specify the package!

```
# from PACKAGE import MODULE

from pytime import pytime


# The names don't need to be the same:

from pygame import joystick

# Yes - that's real!
```

Change your file to read `from pytime import pytime`. Does it work?

> ***Protip**: Remember that package means library!*

Installing PyTime

New packages need to be installed.

- Let's install `pytime`.

In your command prompt:

```
pip3 install pytime
```

Once that's successful, try again to run your file.

***Protip**: `pip` stands for `Pip Installs Packages`. `pip3` uses Python3.*

Note: Repl.it is a great website for testing, because it automatically installs libraries for us.

PyTime Holidays

Let's explore PyTime:

- Scan the PyTime docs, to find the `mother` function.

When is Mother's Day?

```
# This gets mother's day of 2016

mothers_day = pytime.mother(2016) # 2016-05-08
```

What about this year?

```
# This gets mother's day of this year

mothers_day = pytime.mother()
```

Try these in your file to be sure you can call PyTime functions!

PyTime Documentation

Here are some examples of PyTime in action:

```
>>>pytime.father()                    # Father's Day without a year (defaults to c
datetime.date(2015, 6, 21)
>>>
>>>pytime.mother(2016)                 # 2016 Mother's Day
datetime.date(2016, 5, 8)
>>>
>>>pytime.easter(1999)                 # 1999 Easter
datetime.date(1999, 4, 4)
```

Quick Review

Not all modules are in the standard library. If you try to import a module Python doesn't recognize, you'll get a `ModuleNotFoundError`.

When importing from the standard library, the package is implied:

```
# (from standard) import MODULE

import random
```

Otherwise, you need to specify the package!

```
# from PACKAGE import MODULE

from pytime import pytime

from pygame import joystick
```

If you're using a non-standard package like `pytime` or `pygame`, you'll have to also install it.

You can use the website repl.it for testing small pieces of code - it has packages installed.

Up next: Continuing exploring documentation.

You Do: PyTime Festivals

Look through the PyTime docs; can you find the `father` and `easter` functions?

In your local file, pick a year. In that year, print the month and day of:

- Mother's Day.
- Father's Day.
- Easter Sunday.

run ▶

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

We Do: The Grinch Who Stole Christmas?

Why does the documentation only have 3 holidays?

Time for some sleuthing! Most times, you only need to look at a module's documentation. However, sometimes the person that did the documenting didn't write everything down.

Because a module is simply a `.py` file, we can view it.

- Open up the project's Github page.
- Look at the files and folders near the top of the page.
- Click on the folder `pytime`.
- Click on the file `pytime.py`.
- Scroll down to the `Festivals` section at the bottom of the file.
- What function do you see which would likely give you Christmas Day?

You Do: Using the PyTime Module

In your `pytime_test.py`:

1. Decide on a list of three holidays you like. • (e.g. Christmas, Halloween…)
1. Write a function that prompts a user for a year and a holiday.
- (e.g. “Enter a year”, then “Choose Christmas, Halloween, or Mother’s Day”)
1. Have your function print out the date of that holiday for that year.
- (e.g. “In 2016, Mother’s Day was on 2016-05-08”)

Summary and Q&A

Modules are `.py` files with functions. They're written by other people for us to use!

- A packages (a.k.a. library) is a bundle of one or more modules.
- Python's standard library has a lot of common modules! `random`, `itertools`, etc.
- Nonstandard libraries need to be installed (`pip3 install pytime`).

To use modules, at the top of your file, put:

```
# From the standard library: `import MODULE`
import random

# From non-standard packages: `from PACKAGE import MODULE`
from pytime import pytime
from pygame import joystick

# And preface your function with the module name.
mothers_day = pytime.mother(2016) # 2016-05-08
```

Additional Resources

- Python Modules
- Python's Standard Library
- Write a module in python 3 - Digital Ocean
- Itertools
- Random
- PyTime
- List of Commonly Used Packages
- Useful Modules by Discipline
- Further Reading
- Formatting Datetime