Modules and Scripting

Overview

In this lesson, students will discover how to import and use libraries to enhance their programs. They will also learn how to read documentation of new libraries.

Duration

120 minutes

Learning Objectives

In this lesson, students will:

- Understand how libraries, packages, and modules relate to one another.
- Use import statements to access Python libraries.
- Use Python scripts to automate tasks.
- Create Python scripts to read and write to files.

Pre-Class Materials and Preparation

For remote classrooms: Virtual breakout rooms and Slack may be needed to facilitate the partner exercise and discussions. As you plan for your lesson:

- Consider how you'll create pairs for the partner exercise (randomly, or with pre-assigned partners).
- Determine how (if at all) exercise timing may need to be adjusted.
- For helpful tips, keep an eye out for the **For remote classrooms** tag in the speaker notes.
- Prepare screenshots and answers to exercises in advance so that they can be easily shared in Slack during your lecture.

FOR PRESENTER USE ONLY

Suggested Agenda

Time	Activity
0:00–0:50	**Welcome + Modules**
0:50–1:00	**Break**
1:00–1:50	**Scripting**
1:50–2:00	**Wrapping Up, Q&A, and Exit Ticket Completion**

Jupyter Notebook

The exercises referenced in this lesson can be found in the Python Workbooks + Data folder.

Modules and Scripting

GENERAL ASSEMBLY

Today's Learning Objectives

In this lesson, you will:

- Understand how libraries, packages, and modules relate to one another.
- Use import statements to access Python libraries.
- Use Python scripts to automate tasks.
- Create Python scripts to read and write to files.

Modules and Scripting

Modules

A Little Help From Our Friends

Modules are collections of helpful Python code and functions we can use. Instead of reinventing the wheel, modules provide us with immediate benefits:

- Reliable, heavily tested code.
- Well-known patterns for easy collaboration.
- The ability to focus on your application's higher-level needs.

Importing from the Standard Library

Some modules are so commonly used that they are bundled with Python itself.

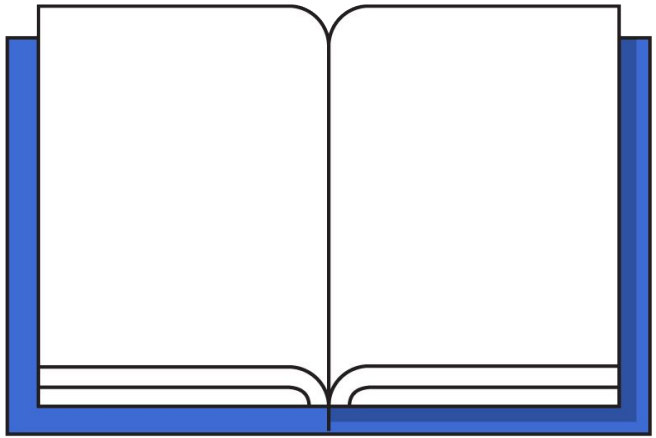
To get a feel for modules, let's explore the **`random`** module. In order to use a module, we first have to import it by including an import statement at the top of our code:

```
import random

# Now, we can use the random module and its methods, such as randint.

random_die_roll = random.randint(1,6)
```


Read the Docs

The downside of borrowing someone else's code is having no idea how to use it — at first, that is!

Python's Standard Documentation provides a good place to begin reading up on the `random` module. Other modules may have different documentation sources.

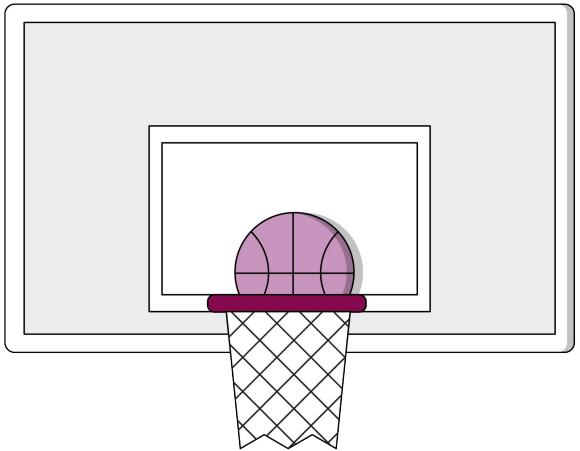


Solo Exercise:

6.1 Games of Chance

20 minutes

Let's explore the **`random`** module using some challenges involving random chance. Read the documentation to discover helpful methods for each task.

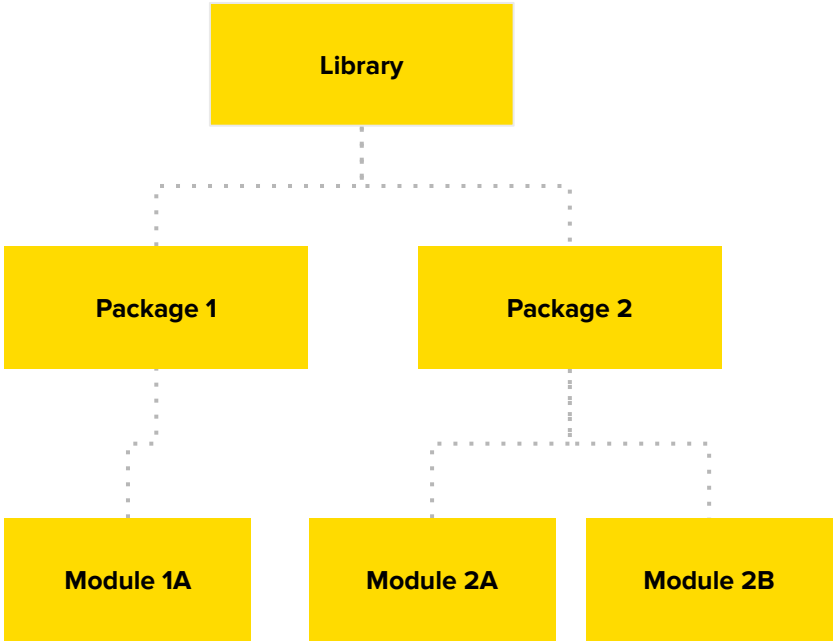


Library Hierarchy

Libraries are large collections of tools that will prove very helpful to you.

These libraries are often organized into one or more **packages**.

Packages can be organized further into one or more **modules**.



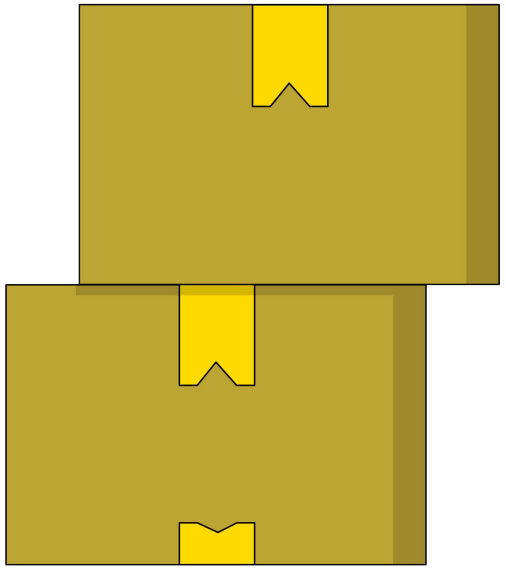
Package Delivery

Libraries, packages, and modules need to be installed in order to use these tools on your computer:

```
%pip install library_name
```

If you want to import only one module from a package:

```
from package_name import module
```

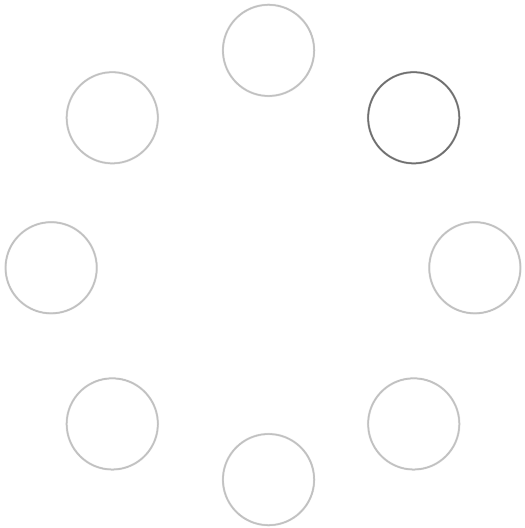


Group Exercise:

6.2 Geocoder Road Trip

20 minutes

Install and use the “geocoder” library for the challenges in Section 6.2.



Modules and Scripting

Scripting

Putting It All Together

Armed with the foundational building blocks of Python, we can start creating genuinely useful scripts to automate specific tasks, like:

- Filing analysis.
- Sending reporting emails.
- Formatting files.
- Processing data.

What's a Script?

When people refer to **scripts**, they usually mean code that:

- Takes input.
- Gives output.
- Reads or writes to a file.
- Performs a task.

We have "Performs a task" down! Let's look at how we can build useful tools in Python by combining these other layers in our programs.

Input From Users and Files

User Input

Prompt users to type in a response to given text, then process that response as a variable.

```
name = input("Type your name")
# the script pauses to wait
# continues when user submits
print(name)
```

File Input

Access the contents of a specific file from within a Python script.

```
log = open("./log.txt", "w")
# opens log.txt in 'write' mode
log.write("Another great day!")
# writes to the end of the file
log.close()
```


Open for Business

When accessing files, you need to specify the “mode” in which the file should be opened.

You must also be sure to close the file after working with it. Use the following pattern to let Python do that for you automatically:

```
with open("log.txt", "a") as log:
    log.write("All clear.")
```

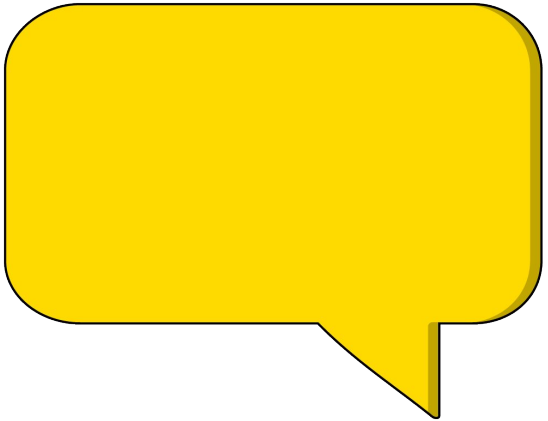
	Mode
'r'	Read-only mode.
'w'	Write.
'a'	Append new contents to the end of the file.
'+'	Add to any mode to include read and write.

Partner Exercise:

6.3 So Much to Do...

10 minutes

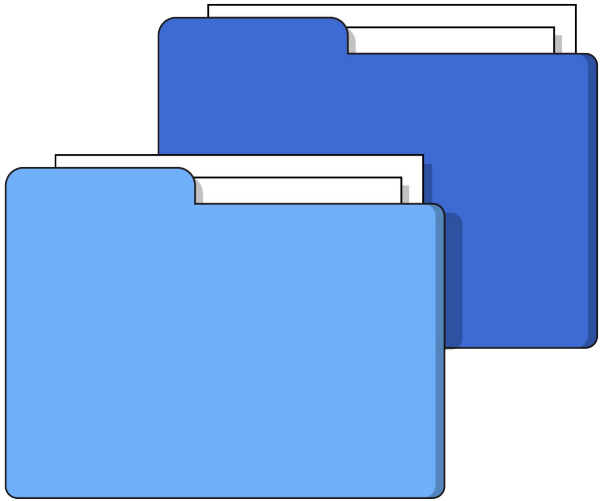
Add a comment to each line of the “to-do list” script in Section 6.3 of the notebook.



With These Powers Combined...

Managing files manually can get a bit awkward. Fortunately, there are plenty of **modules** out there to make things easier — especially for common file and data formats like .csv!

Let's explore a basic module called "**csv**" for just this purpose.



Discussion:

What Are Spreadsheets Made Of?

Before we start manipulating .csv files in Python, let's reflect:

What Python data structure might be best for storing a .csv row ID and the data in the corresponding row in the .csv file?

A dictionary is best, because it is a set of key (row ID) and value (the data corresponding to that row) pairs.

Discussion:

What Are Spreadsheets Made Of? (Cont.)

Before we start manipulating .csv files in Python, let's reflect:

What Python data structures might be best for representing a whole table of rows?

A list of dictionaries, because a list preserves the order of the rows.

Discussion:

Reading and Writing

Let's explore how we would read and write to a .csv file by consulting the documentation for the **csv** library. Consider:

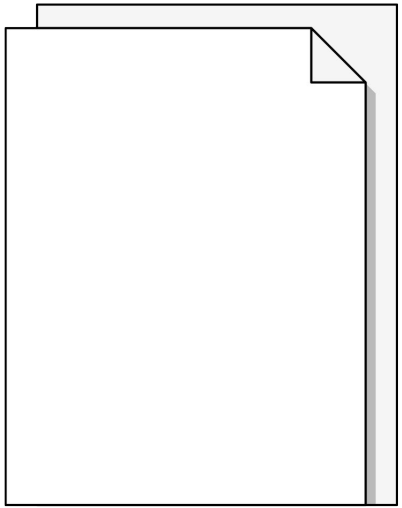
1. How can we read a row from a .csv file?
2. How can we write a new row to an existing .csv file?
3. How can we modify the existing rows of a .csv file?

Partner Exercise:

6.4 TPS Reports

20 minutes

Let's translate a given list of dictionaries into a .csv file using the **csv** module.



Discussion:

Automating With Python

One of the main uses of Python as a productivity tool is creating reusable scripts to analyze and generate .csv files.

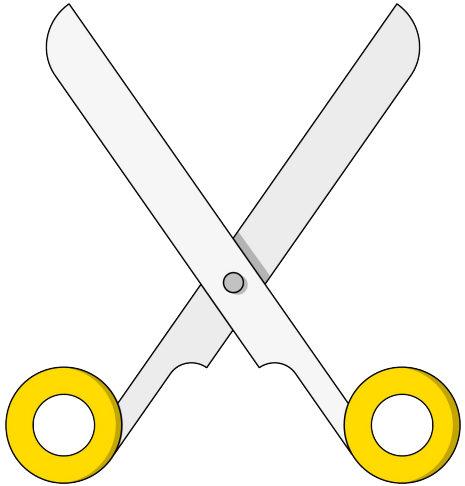
What regularly scheduled processes can you anticipate creating a Python analysis script for, either for your own role or to enhance the productivity of another team member?

Solo Exercise:

6.5 Ultimate RPS Challenge

20 minutes

As a bonus challenge, let's use our .csv-writing abilities to keep track of all of the games played from a command line version of Rock, Paper, Scissors.



Modules and Scripting

Wrapping Up

Recap

In today's class, we...

- Discussed how libraries, packages, and modules relate to one another.
- Used import statements to access Python libraries.
- Used Python scripts to automate tasks.
- Created Python scripts to read and write to files.

Looking Ahead

On your own:

- Work through the Python progress assessment on myGA (due at the end of the unit).
- Start thinking about capstone project ideas!

Next Class:

APIs

Don’t Forget: Exit Tickets!


GA