Rags to Riches

(Using modules written by others)

In the past, you have written several of your own modules...

- `caesar_cipher.py` ➡️ `import caesar_cipher`
- `validator.py` ➡️ `import validator`
- `text_mod.py` ➡️ `import text_mod`

There are two key benefits of writing your own modules:

1. Modules help you to organize code.
2. Modules allow you to reuse code.

These become more evident the larger your system gets!

Navigate to **github.com/GatorEducator/gatorgrader**

1. What would `gatorgrader.py` look like without the use of modules?
2. In the `gator` folder, what is one module that is reused across different files?

In the past, you have also imported modules that you did not write yourself...

- `import random`
- `import sys`

These modules added additional functionalities to your programs...

- `import random`: Used to generate random numbers (e.g. `random.random()`)
- `import sys`: Used to access the command-line arguments (e.g. `sys.argv[1]`)

But, where do these come from?

Navigate to **github.com/python/cpython**

When you type the command `python` (e.g. `python todo_list.py`), you are *actually* running a program that executes Python code.

The repository you are looking at contains the source code of this program. This is an example of open-source software.

What is **open-source software**?

What can you do with open-source software?

Well, it depends... Look at the **Copyright and License Information** section.

Most open-source licenses allow you to...

- Import the program into your program
- Copy parts of the code into your program

- Take the entire program and improve it as your own program

Open-source software powers the world!

- Websites
- Operating systems (JupyterLab!)
- Programming languages

The `python` program ships with several modules.

Look in the `Modules` folder. Which file contains the `random` module?

`random`

Used to generate randomness.

Some `random` functions you have seen (and one new function)...

	Input	Output
`.randint(a, b)`	Two integers, *a* and *b*	A random integer between the two integers--*a* and *b* are inclusive
`.random()`	None	A random floating-point number in the range [0.0, 1.0)
`.choice(seq)`	A sequence (e.g. list)	A randomly selected item from the given sequence
`.shuffle(seq)`	A sequence (e.g. list)	None--it randomly shuffles the items in the given list (Note that this will change the given list!)

	Input	Output	Use Case
`.randint(a, b)`	Two integers, *a* and *b*	A random integer between the two integers--*a* and *b* are inclusive	Generate integers
`.random()`	None	A random floating-point number in the range [0.0, 1.0)	
`.choice(seq)`	A sequence (e.g. list)	A randomly selected item from the given sequence	
`.shuffle(seq)`	A sequence (e.g. list)	None--it randomly shuffles the items in the given list (Note that this will change the given list!)	

	Input	Output	Use Case
`.randint(a, b)`	Two integers, *a* and *b*	A random integer between the two integers--*a* and *b* are inclusive	Generate integers

	Input	Output	Use Case
`.random()`	None	A random floating-point number in the range [0.0, 1.0)	Generate fractional numbers (e.g. percentages)
`.choice(seq)`	A sequence (e.g. list)	A randomly selected item from the given sequence	
`.shuffle(seq)`	A sequence (e.g. list)	None--it randomly shuffles the items in the given list (Note that this will change the given list!)	

	Input	Output	Use Case
`.randint(a, b)`	Two integers, *a* and *b*	A random integer between the two integers--*a* and *b* are inclusive	Generate integers
`.random()`	None	A random floating-point number in the range [0.0, 1.0)	Generate fractional numbers (e.g. percentages)
`.choice(seq)`	A sequence (e.g. list)	A randomly selected item from the given sequence	Choose one
`.shuffle(seq)`	A sequence (e.g. list)	None--it randomly shuffles the items in the given list (Note that this will change the given list!)	

	Input	Output	Use Case
`.randint(a, b)`	Two integers, *a* and *b*	A random integer between the two integers--*a* and *b* are inclusive	Generate integers
`.random()`	None	A random floating-point number in the range [0.0, 1.0)	Generate fractional numbers (e.g. percentages)
`.choice(seq)`	A sequence (e.g. list)	A randomly selected item from the given sequence	Choose one
`.shuffle(seq)`	A sequence (e.g. list)	None--it randomly shuffles the items in the given list (Note that this will change the given list!)	Randomize data (e.g. to choose distinct multiple)

Make sure to write your partner's name in the `README.md` and commit and push your work!