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**.

Why is open-source software important?

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
- Git

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

	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	
`.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!)	

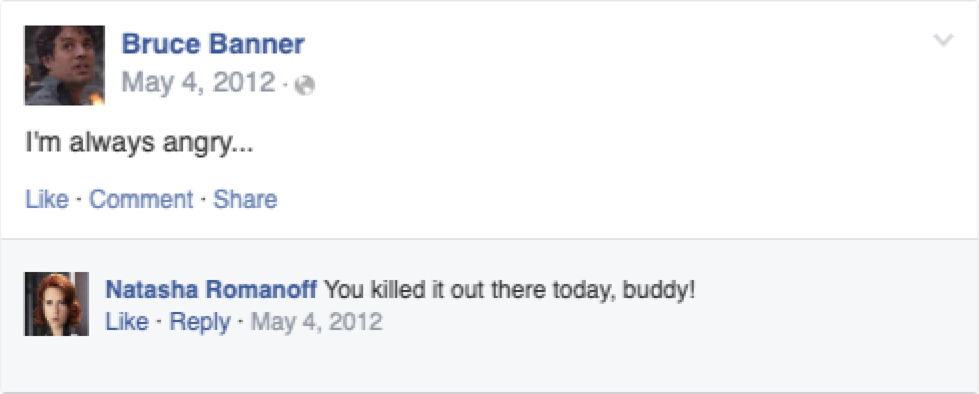
	Input	Output	Use Case
`.randint(a, b)`	Two integers, *a* and *b*	A random integer between the two integers--*a* and *b* are inclusive	Generate integers
`.random()`	None	A random floating-point number in the range [0.0, 1.0)	Generate fractional numbers (e.g. percentages)
`.choice(seq)`	A sequence (e.g. 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!

JavaScript Object Notation (JSON)

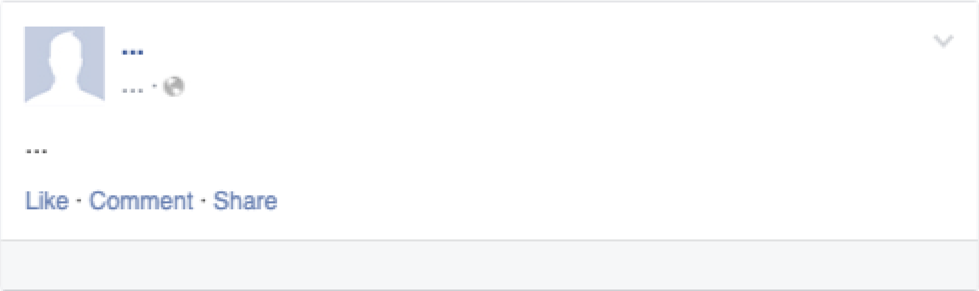
Data power programs.



Bruce Banner
May 4, 2012 ·

I'm always angry...

Natasha Romanoff You killed it out there today, buddy!



...
... ·

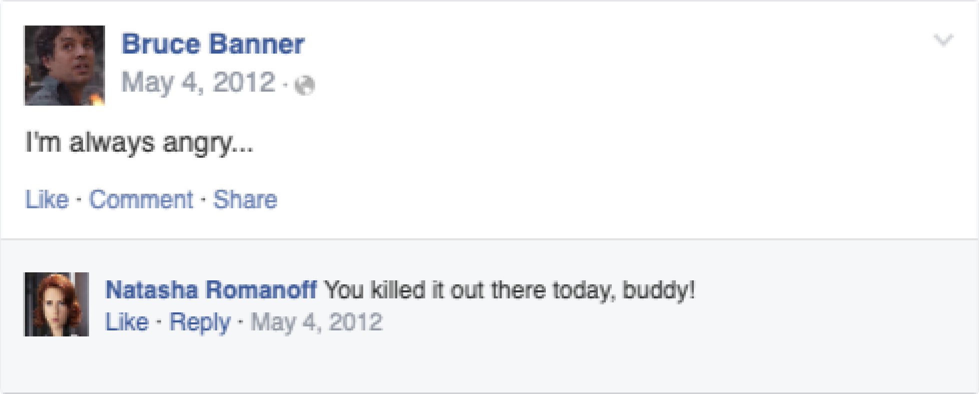
...

What is one program you use on a daily basis that relies on data?

Textual data can be stored in many different ways...

```
red,red,blue,orange,green,blue,blue,green,green,red,blue,blue,green,green,
orange,green,blue,red,blue,blue,red,blue
```

E.g. Comma-separated values (CSV)



Bruce Banner
May 4, 2012 ·

I'm always angry...

Natasha Romanoff You killed it out there today, buddy!

```
"Bruce Banner","05/04/2012","I'm always angry...","Natasha
Romanoff","05/04/2021","You killed it out there today, buddy!"
```

We need to know what each value *is*!

```
post = ["Bruce Banner", "05/04/2012", "I'm always angry...",
        "Natasha Romanoff", "05/04/2012",
        "You killed it out there today, buddy!"]
post_author = post[0]
print(post_author)
```

```
Bruce Banner
```

```
post = ["05/04/2012", "Bruce Banner", "I'm always angry...",
        "Natasha Romanoff", "05/04/2012",
        "You killed it out there today, buddy!"]
post_author = post[0]
print(post_author)
```

```
05/04/2012
```

What are some things that could change the positions of items in a list?

Two problems to using CSV:

1. Don't know what each value *is*
2. Position of value could change

JavaScript Object Notation (JSON) to the rescue!

```
{
    "author": "Bruce Banner",
    "date": "05/04/2012",
    "text": "I'm always angry...",
    "comments": [
        {
            "author": "Natasha Romanoff",
            "date": "05/04/2021",
            "text": "You killed it out there today, buddy!"
        }
    ]
}
```

Benefits of JSON:

1. Encodes what each value *is*
2. *Standardizes* data; every post will have the same keys--we can have expectations about the data that are unlikely to change (top-level `author` key will always map to post's author)

What data structure does JSON look most similar to?

Does any part of the JSON look like another data structure?

Keep in mind that JSON is *not* a Python dictionary! It is simply text formatted *like* a dictionary, with keys and values. You can store it in a text file.

But, data is only useful if you actually *use* it.

Because it looks most like a dictionary, it makes the most sense to interact with it as a dictionary.

Let's load the JSON into a Python dictionary we can use.

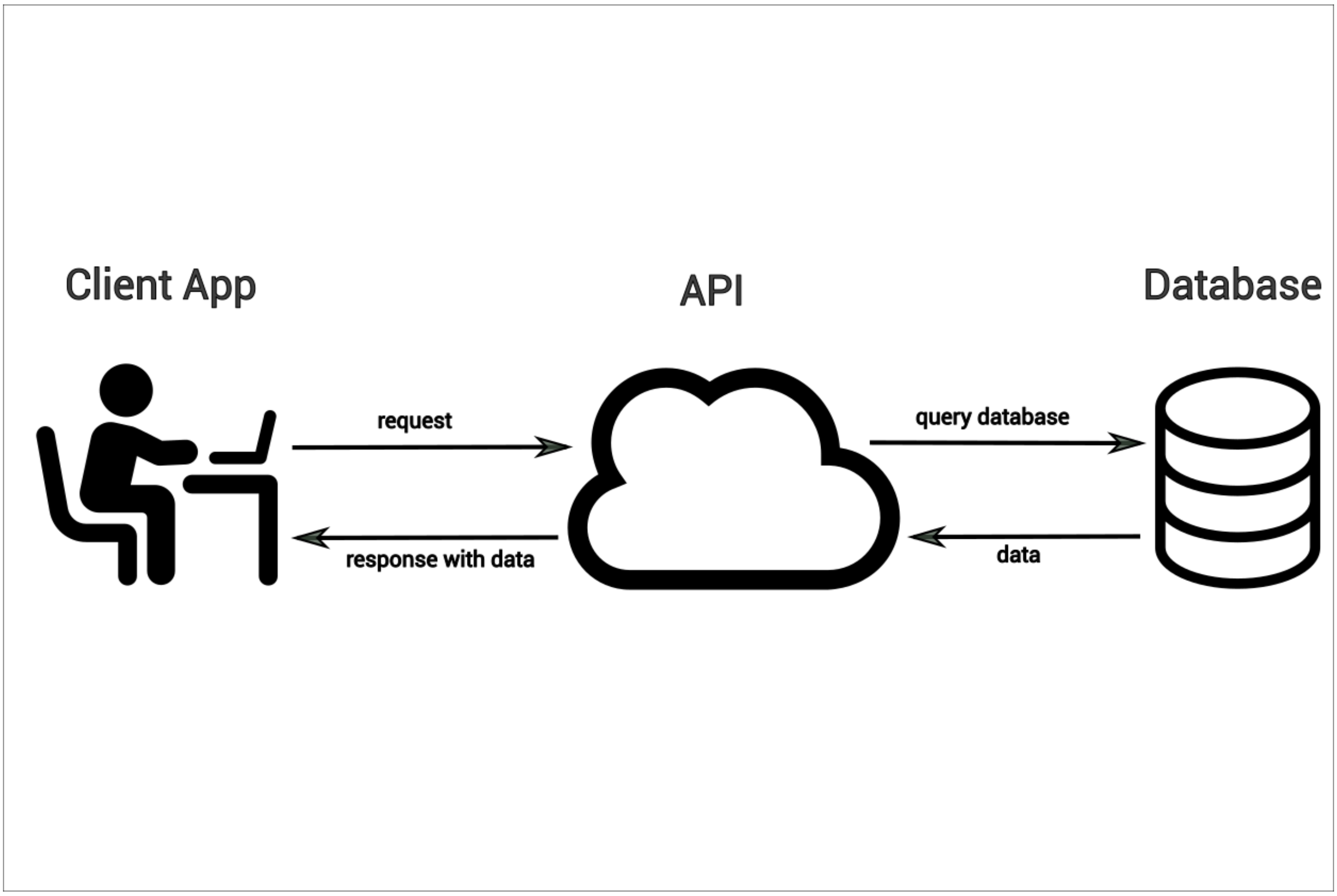
There's a module for that: `json`

Two ways you might get JSON:

1. As a string
2. In a file

Either way, need to `import json`

What is meant by "get" JSON?



Navigate to api.nasa.gov/index.html

1. Click on an API that interests you
2. Copy an example query (if it has `GET`, don't include it!) and put into the URL bar
3. Describe what you see
    - What does the structure of the data look like?
    - Does it contain any other links you can navigate to (e.g. images)?

`json` module functions:

1. As a string - `.loads(string)`
2. In a file - `.load(file_object)`

```
import json

post_string = '{"author":"Bruce Banner","date":"05/04/2012","text":"I\'m
always angry...","comments":[{"author":"Natasha
Romanoff","date":"05/04/2021","text":"You killed it out there today,
buddy!"}]}'

post = json.loads(post_string)

print(post)
```

```
{'author': 'Bruce Banner', 'date': '05/04/2012', 'text': "I'm always
angry...", 'comments': [{'author': 'Natasha Romanoff', 'date':
'05/04/2021', 'text': 'You killed it out there today, buddy!'}]}
```

With the same JSON in `post.json` file...

```
import json

post_file = open('post.json')

post = json.load(post_file)

print(post)
```

```
{'author': 'Bruce Banner', 'date': '05/04/2012', 'text': "I'm always
angry...", 'comments': [{'author': 'Natasha Romanoff', 'date':
'05/04/2021', 'text': 'You killed it out there today, buddy!'}]}
```

`astronauts.py`

movies.py