Road map

- Review
- HTTP
- Web API's
- JSON in Python
- Examples
import bottle

@bottle.route("/")
def any_name():
    response = "<html><body><p>"
    response = response + "Hello from the server!"
    response = response + "</p></body></html>"
    return response

bottle.run(host="0.0.0.0", port=8080, debug=True)
Python Web Server

import bottle

Import the library

We had to install it first

pip install --user bottle
Python Web Server

bottle.run(host="0.0.0.0", port=8080, debug=True)

Run the server on port 8080
@bottle.route("/\")
def any_name():
    response = "<html><body><p>"
    response = response + "Hello from the server!"
    response = response + "</p></body></html>"
    return response

Host content on the root path "/"
Web Server

- **Client** Sends requests to server
- **Client** Sends requests to server
- **Client** Sends requests to server

**Web Server**
- Software runs continuously and waits for requests from clients
- Responds to requests
How do clients send requests to a server?
Road map

Review

▶ HTTP ◀

Web API's

JSON in Python

Examples
HTTP

We communicate with web servers by making HTTP requests.

The server will send an HTTP response in return.

The request is sent to a specific URL in the format:

<protocol>://<server>/<path>?<query_string>

- **Protocol**: HTTP or HTTPS
- **Server**: The domain name for the server (e.g., [www.buffalo.edu](https://www.buffalo.edu))
- **Path**: Name for the resource being requested
  - The path corresponds to the string in the annotation in bottle
- **Query String**: Provide additional info in key-value pairs
  - Unused by most sites

[https://engineering.buffalo.edu/computer-science-engineering.html](https://engineering.buffalo.edu/computer-science-engineering.html)
import urllib.request

url = "https://engineering.buffalo.edu/"
url = url + "computer-science-engineering.html"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)

Prints the HTML for CSE@UB's homepage
HTTP Request

```python
import urllib.request
```

Import the urllib.request module

Built-in to python

No need to install
url = "https://engineering.buffalo.edu/"
url = url + "computer-science-engineering.html"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)

urllib.request contains the "urlopen" function that will setup an HTTP request to a provided url and return a response object.

The response can be read by calling read which returns a binary string.

Calling decode on the binary string converts it to a string.

Now do anything you can do with strings. For this example we only print the response to the screen.
import urllib.request

url = "https://engineering.buffalo.edu/"
url = url + "computer-science-engineering.html"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
Query Strings

A set of key-value pairs

key and value separated by "="
key-value pairs separated by "&"

https://www.youtube.com/watch?v=5jmN_tBS0t4
Query String: "v=5jmN_tBS0t4"
• key "v" with value "5jmN_tBS0t4"

https://www.google.com/search?q=cats&as_filetype=gif&lr=lang_ja
Query String: "q=cats&as_filetype=gif&lr=lang_ja"
• key "q" with value "cats"
• key "as_filetype" with value "gif"
• key "lr" with value "lang_ja"
import urllib.request

url = "https://www.amazon.com"
url = url + "/s?keywords=pens&sort=price-desc-rank"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)

Sends an HTTPS request to the server "www.amazon.com"

Requests the path "/s"

With a query string containing 2 key key-value pairs
• key "keywords" with value "pens"
• key "sort" with value "price-desc-rank"

Searches for the most expensive pens on amazon
Amazon Server Sends a Response

Uni-Ball Uni-Ball Elite Stick Micro Point Roller Ball Pens, 3 Black Ink Pens
by Uni-ball
$129,999.00 (1 new offer)

I'll pass
The Internet, as most people know it, is designed for human consumption.

What if we want to write software that reads data from the Internet?

How do we parse through the raw HTML in Python?
Web Scraping

A web scraper is software that reads data from HTML.

Many libraries exist to make this easier.

We won't explore this in CSE115, though it can be a fun area to explore on your own.
Road map

Review

HTTP

▷ Web API's ◁

JSON in Python

Examples
Web API

The Internet, as most people know it, is designed for human consumption.

What if we want to write software that reads data from the Internet?

Web APIs are hosted by web servers at urls, but instead of sending HTML/CSS/JavaScript they send raw data.

Designed for programmatic consumption.

Typically send data as JSON.
import urllib.request

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)

Connect to the open notify api

Documentation: http://open-notify.org/Open-Notify-API/ISS-Location-Now/

Returns a JSON String
import urllib.request

text = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)

{
    "message": "success",
    "timestamp": 1540176365,
    "iss_position": {
        "latitude": "20.6716",
        "longitude": "-163.2050"
    }
}

We can get raw data, but what do we do with this JSON string?

What is a JSON string?
Road map

Review

HTTP

Web API's

▶ JSON in Python ◀

Examples
JSON (JavaScript Object Notation) is a data format that can be represented as strings.

Send these strings to communicate across the Internet, and elsewhere.

All programming languages can read strings:
• Doesn't matter what language was used to write the client or server program
• They can all "speak" JSON since it's just strings

More flexible than CSV
JSON

Only 6 different data types

- **String**: Any value in "double quotes"
- **Number**: Any value not in quotes "true", "false", and "null" will be interpreted as a number. Should be formatted as an integer or floating point number
- **Boolean**: Either "true" or "false" without the quotes
- **Null**: The word "null" without the quotes
- **Array**: A comma-separated list of values surrounded by [brackets]
- **Object**: A comma-separated list of key-value pairs surrounded by {braces}

```
[{"title":"God Am (Live 1996)","artist":"Alice in Chains","ratings":[5,4],"youtubeID":"74P4W_okEqA"},{"title":"Fade to Black","artist":"Metallica","ratings":[5,2],"youtubeID":"WEQnzs8wl6E"}]
```

Closely resembles Javascript and Python syntax that we've seen, except it is a string. See e.g. [https://www.json.org](https://www.json.org) or [http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf](http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf) for more info.
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()
content = json.loads(content_string)
print(content)

Use the built-in json module to handle JSON strings

Call json.loads to convert a JSON string to python types
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()
content = json.loads(content_string)
print(content)

{
    'message': 'success',
    'iss_position': {
        'longitude': '-110.1453',
        'latitude': '-39.6226'},
    'timestamp': 1540177620
}
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()
content = json.loads(content_string)

print(content['iss_position']['longitude'])
print(content['iss_position']['latitude'])

The result looks similar to the JSON string, but now it is a **Python dictionary** instead of a string

We can use dictionary functions to process the data.
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()
content = json.loads(content_string)

print(content['iss_position']['longitude'])
print(content['iss_position']['latitude'])

-70.6226
-51.3781

*Note: The numbers are different across slides since each time the code is executed we are getting the current location of the ISS. With web APIs we can work with live data!
Road map

Review

HTTP

Web API's

JSON in Python

▶️ Examples ◀️
Want More?

Big list of APIs
- https://github.com/toddmotto/public-apis

Hackathon next week
- You have a lot of power at your fingertips with what we've covered so far!
- Build something great!