

Introduction to APIs

Session 2, Oct. 25

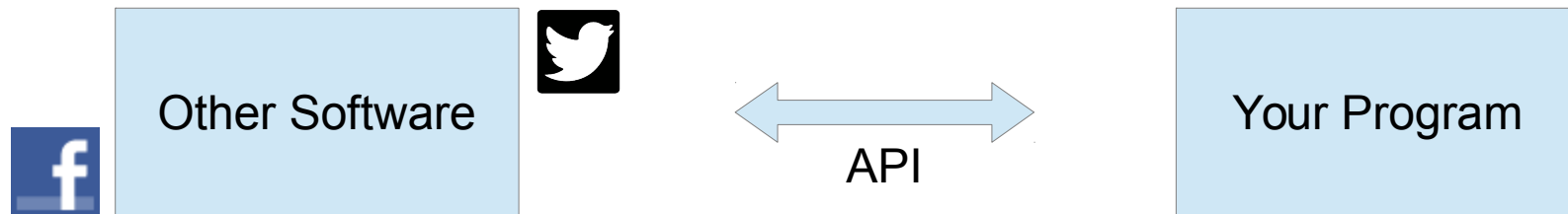


API: Application Programming Interface

- What the heck does that mean?!
- Interface: allows a user to interact with a system
 - **Graphical User Interface (GUI)**: interact with a program using a point/click/type interface
 - **Command-Line Interface (CLI)**: interact with a program via the command line: for example, `cd /home/downloads`
- API: interact with an existing program programmatically
- **e.g.** The Twitter API allows you to interact with Twitter (write programs that post tweets, mine tweets for data, or look at social structures)

APIs, continued

- How do they work?



- You are provided with resources
 - **Software modules:** Allows you to call functions that have been written for you!
 - **Documentation:** Tells you how to use the software modules and different function calls
 - **Authentication:** Allows you to prove you are authorized to interact with the software

Web APIs

Web APIs

- Gives you a way to ask for and receive data over the internet
- How? Using **hyper-text transfer protocol (HTTP)**
- Web APIs are “universal”
 - All programming languages know how to use HTTP!
- HTTP “Methods” (actions)
 - **GET**: asks for data from a server
 - **POST**: sends data to a server

Asking for data

- When you type a URL into the navigation bar of your browser, you are requesting data for that webpage



Asking for data with URLs

- We will be asking for data with URLs (**Universal Resource Locator**)
- If you want to see a specific YouTube video, you ask YouTube for the video by encoding its ID in the URL:

<https://www.youtube.com/watch?v=1wnE4vF9CQ4>

HTTP GET URL --> server returns 200 OK and data

Receiving data

- When requesting to view a webpage in your browser, the information is sent back to you as HTML
- Your browser parses and displays the page based on the HTML it receives!
- APIs often return data in JSON format, as it is easy to parse and display information (like a Python map)

Sample JSON: Wikipedia Web API

HTTP **GET** `http://en.wikipedia.org/w/api.php?format=json&action=query&titles=Main%20Page`

```
{
  "query": {
    "pages": {
      "15580374": {
        "pageid": 15580374,
        "ns": 0,
        "title": "Main Page"
      }
    }
  }
}
```

API Documentation

- Every API is different: no “one true way”
- Luckily, every API is documented!
 - Use your favorite web search engine for “\$software API documentation”
- For Wikipedia web API example:
 - http://www.mediawiki.org/wiki/API:Main_page

Using an API to build a data set

- What do we need?
 - All the Python tools we learned last time (e.g. variables, lists, loops)
 - Ability to open URLs on the Web
 - Ability to create custom URLs
 - Ability to save files
 - Ability to understand the data the API gives us
 - A few new tools...

urllib2 library

- Libraries are python modules written by others that perform common tasks
- The functions in these modules can be used by anyone
- `urllib2` is a python library for opening URLs
 - There is also an `urllib`, but `urllib2` is better.

```
>>> import urllib2
```

urlopen

- Function that allows you to open urls

```
>>> page = urllib2.urlopen('https://uwaterloo.ca')
```

- All information for the page we opened is now saved in our 'page' variable as a special object

```
>>> data = page.read()
```

- The HTML for the page is now stored in data

Formatting Strings

```
>>> name = "Spongebob Squarepants"  
>>> print "Who lives in a  
pineapple under the sea?\n%s!"  
% name
```

Who lives in a pineapple under the
sea?

Spongebob Squarepants!

Formatting Strings, continued

```
>>> howmany = 101
```

```
>>> print "I have %s dalmations!"  
      % int
```

```
I have 101 dalmations!
```

```
>>> print "I have %s dalmations!  
          %s!!!" % (int,int)
```

```
I have 101 dalmations! 101!!!
```


File Operations

- You may have seen this already if you attended the Shakespeare session!
- Idea: files allow us to store and process a **lot** more data (GBs+)
- We will cover opening files for **reading** and **writing**

Create new files

```
>>> newfile = open('myfile', 'w')
```

- No need to include any modules, this is a standard Python function like `print`
- 'w' stands for “write” mode

Writing to files

```
>>> newfile.write("Hello, world!")
```

```
>>> str = "We like files"
```

```
>>> newfile.write(str)
```

- Writes the data to our newly created file

```
>>> newfile.close()
```

- Closing files when you're done is “polite,” like closing the door behind you
- (Real reasoning for this is beyond the scope of these workshops)

Read from existing files

```
>>> file = open('myfile', 'r')
```

- 'r' stands for “read” mode

```
>>> line = file.readline()
```

```
>>> line
```

```
Some text from the file
```

Read file in other ways

```
>>> for line in file:  
...     ((do something))
```

- Iterate over lines in a file

```
>>> file_as_string = file.read()
```

```
>>> print(file_as_string)
```

```
'Lots of text from the file in  
string form...'
```

- Read file into Python all at once as a single string

Live Web API Demo!

placekitten

A quick and simple service for getting pictures of kittens for use as placeholders in your designs or code. Just put your image size (width & height) after our URL and you'll get a placeholder.

Like this: <http://placekitten.com/200/300>
or: <http://placekitten.com/g/200/300>



Requesting kittehs!

- The documentation for placekitten is very simple! We've just read all of it.
- We specify size by putting it in the URL request (height/width):

<http://placekitten.com/250/350>

- We specify grayscale by adding a 'g' to the URL:

<http://placekitten.com/g/200/300>

Exercise: Try placekitten for yourself!

- First import urllib2

```
>>> from urllib2 import urlopen
```

- Then, request data by opening the URL

```
>>> site = urlopen('http://placekitten.com/250/350')
```

- Now read the data into a variable

```
>>> data = site.read()
```

Saving our kitten to a file

- We've successfully requested our data, so let's save it

```
>>> kitten_file = open('kittteh.jpg', 'w')
```

```
>>> kitten_file.write(data)
```

```
>>> kitten_file.close()
```

- Find your file, and see what it is in it!

placekitten exercise

- Write a program that asks for a image dimensions and retrieves a kitten of that size and save your solution in the file 'getkitten.py'.
- Toolkit
 - `raw_input()`
 - String formatting:
`'ninjapants123%spineapples456%s' % (var1, var2)`
 - Open file for writing (don't forget to close it!)
`file = open('myfilename', 'w')`
`file.write(content)`

Other Loose Ends

Other APIs

- Each API is different: be sure to read documentation!
- Examples:
 - Facebook
 - Twitter
 - Dropbox
 - Wikipedia
 - Basically any of your favorite websites

Rate Limiting

- If we request too much data too often, the servers can't handle all the requests
- Requesting too much information is known as a Denial Of Service (DoS) attack
 - This affects everyone who is using the site
- Popular APIs limit the amount of requests you can make in a time window
- **e.g.** Twitter may allow 15 requests every 15 minutes from a single program

Authentication

- You may need to establish your identity to an API
 - e.g. Twitter doesn't want just anyone to be able to programmatically access your direct (private) messages!
- You will often be provided with a “secret” to prove the identity of your program
 - Also called “development token”, “access token”, etc.
- For the afternoon, Twitter session attendees need to provide authentication data to talk to the Twitter API

Text Encoding

- Not required knowledge, but may help you understand bugs
- What is text encoding?
 - Text is stored as 0's and 1's in your computer, so we have special
 - English alphabet: “encoded” in a small alphabet called ASCII that uses 7 “bits” per character
- Types of encodings:
 - Many special characters: extended ASCII
 - Very large alphabet, including Chinese, Arabic, Hindi, etc.: UTF-8
- We had encoding issues with the Twitter exercises and the Windows console!