JSON, APIs, and Python

What is an API?

- API stands for “Application Program Interface”
  - Not to be confused with an IPA. Don’t order an API at the bar.

- So what is that, in English?
  - An API is:
    - Part of a server that receives requests and sends responses
    - The specification describing what functionality is available, how it must be used, and what formats are accepted as input and returned as output
  - So when you go to Facebook, to your browser (the client) Facebook’s server is an API. It sends requests for data and the server responds.

- Whenever you want to access data from an application, you have to call the API. Think of it as a middleman between you and the data you want.

What formats for data exchange do APIs use?

- Technically, they can communicate using any format that the programmers decide to use.

- Pragmatically, two formats have emerged as dominant:
  - XML (Extensible Markup Language)
  - JSON (JavaScript Object Notation)

- Both are language-independent but use broader conventions familiar to programmers (and follow broadly universal data structures)

Why JSON?

- XML has historically been the dominant data exchange format. But JSON seems to be the emerging favorite (Twitter’s API, for example, has moved away from XML support). Why?
- JSON is:
  - Easy for humans to read (you can open a .JSON file in a text editor and figure out what is going on fairly easily)
  - Easy for machines to read and parse
  - Leaner than XML
  - Especially easy to use if you work in JavaScript

An example of data in JSON vs. XML

JSON (140 characters)

```
{
  "id": 123,
  "title": "Object Thinking",
  "author": "David West",
  "published": {
    "by": "Microsoft Press",
    "year": 2004
  }
}
```

XML (167 characters)

```
<?xml version="1.0"?>
<book id="123">
  <title>Object Thinking</title>
  <author>David West</author>
  <published>
    <by>Microsoft Press</by>
    <year>2004</year>
  </published>
</book>
```

Some JSON syntax rules

- Squiggly brackets are ‘containers’
- Square brackets hold arrays
- Names and values are separated by colons
- Array elements are separated by commas

But really, I don’t care about any of that

- For my purposes, JSON (and XML) files that I get from an API are just boxes that hold data that I want.

- I don’t really care about the nuances of how they are created, structured, or organized outside of the “what I need to know” to open the box and get the data I want out (and usually throwing the box and the rest of its content away).

- My brain wants data to exist in flat files like spreadsheets (or in the R and Python world, data frames).

So let’s take the Swiss Army Knife approach

- Is a Swiss Army Knife the best way to:
  - Tighten a screw?
  - Open a can?
  - Remove the cork from a wine bottle?
  - Carve a turkey?

- Yes, if that’s the only tool that you have (or can afford). So let’s forge ahead.

How Do I get the data?

- Many APIs are open to you. You may have to create an account and apply for an API key. In so doing, you have to agree to the terms and conditions regarding the use of the data.
- ProgrammableWeb (http://www.programmableweb.com) is one of a couple of online API directories (and a good source of info on APIs in general)
- Let’s assume you want to use Google’s API to do some geocoding.

We are already in a grey area

- Google's Terms and Conditions for the Google Maps API says that you can use the service for free to create maps for your own website, but is a little less clearly supportive of other uses

- Regardless, you can create an account and (subject to API usage limits) query the Google Maps API (even without an API key)

- The overriding consideration: *don't be a jerk*. If you make too many queries or make queries too quickly, your IP will get frozen.

Formatting the request

- Google maps uses a standard format for request from data from its API:

https://maps.googleapis.com/maps/api/geocode/json?address=*(well-formatted street address here)*

- Google is nice to you:
  - You can access this without an API key
  - It helpfully fixes your poorly-formatted street addresses for you
  - It returns a JSON-formatted file with your geocoding data

So what to do next?

- Combine your knowledge of loops, parsing data, etc. in Python to write code to create datasets for you.

- The Git repository contains a couple of examples:
    - *football_json.py* uses fantasyfootballnerds.com's API to create a data frame with NFL schedules, and some other goofy stuff
    - *geocode_reshalls.py* is mostly working and uses the Google Maps API to geocode (find lat/lon coordinates) of some of KU's residence halls

- Play with those scripts to get a sense of the process—or write your own.