# Getting started with Projects

# Today's lab

- Project overview and timeline
- Brief comments on scraping and API
- Work phase
    - 20 minutes: work individually to devise a plan
    - 10 minutes: share plan with others
    - 30 minutes: work together to implement plan
- Data collection summary

# Project timeline

| Sun | Mon | Tues | Wed | Thurs | Fri | Sat |
|---|---|---|---|---|---|---|
| | | | | 11/16 | 11/17<br>Milestone 2 | |
| | | | | DATA COLLECTION | | |
| DATA COLLECTION | | | THANKSGIVING: INDEPENDENT WORK | | | |
| | REGROUP | 11/28<br>Milestone 3 | MODELING AND ANALYSIS | | | |
| | | COMMUNICATION | | 12/7<br>Project due | | |

# Final project deliverables

- **Milestone 1** (2 points) – done
- **Milestone 2** (3 points) – due this Friday
- **Milestone 3** (15 points) – due just after Thanksgiving

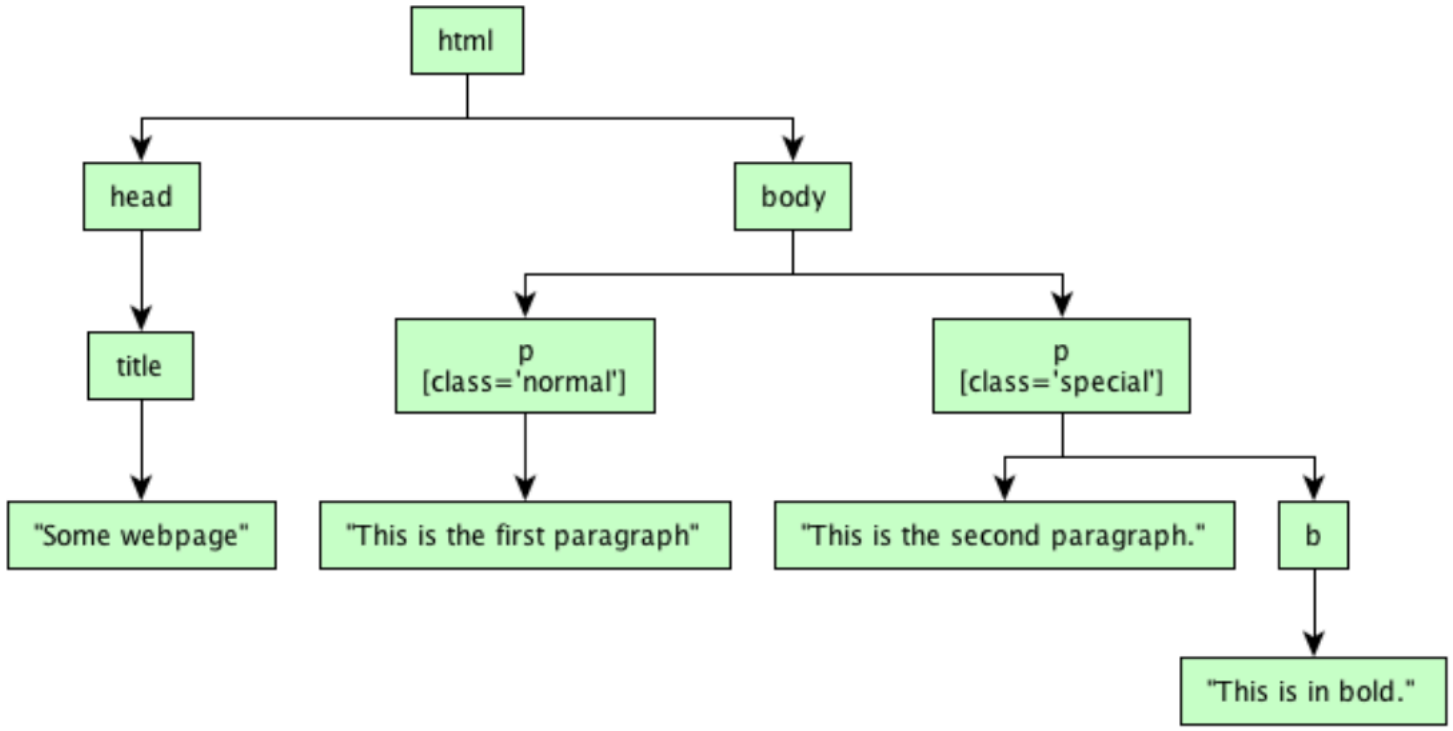
- **Project submission** (80 points)
  - Well-commented Jupyter notebook (30 points)
  - Report on a website (40 points for report + 5 points for website style)
  - Peer evaluation (5 points)

# Scraping with Beautiful Soup

This is the first paragraph

This is the second paragraph **This is in bold.**

```
1
2  <html>
3      <head>
4
5          <title>Some webpage</title>
6
7      </head>
8
9      <body>
10
11         <p class="normal">
12             This is the first paragraph
13         </p>
14
15         <p class="special">
16             This is the second paragraph
17             <b>
18                 This is in bold.
19             </b>
20         </p>
21
22     </body>
23
24 </html>
```



| Code | Result | Example |
| --- | --- | --- |
| `soup.tag` | returns the first instance of tag | `soup.b` |
| `parent.child_tag` | access the tag named 'child_tag' from it's parent tag | `soup.html.head` |
| `tag.contents` | returns all content (text and descendent tags) | `soup.html.head.contents` |
| `tag.string` | returns any strings in the tag, not belonging to child tags | `soup.html.head.string` |
| `tag.children` | returns a "list" of all child tags | `soup.html.head.children` |
| `soup.find_all(tag_name)` | returns a "list" of all tags named "tag_name" | `soup.find_all('b')` |

Review your scraping work from homework 1

# API (application programming interface)

A short introductory video: https://www.youtube.com/watch?v=s7wmiS2mSXY

Informally, say I have data that you want, but I don't want you to mess up the, so I write functions to be used on the data. You can only access the data using my functions. That is, an API uses other people's tools to access other people's proprietary data.

So you need to read the documentation and terms of service carefully.

The Spotify project uses API.

# Let’s get to work!

For all projects, be respectful of the website and read terms and conditions

- Sports and Crime: look at the websites and devise a plan for scraping. See example codes in a-2017 project folder.
- Spotify: familiarize yourself with http://spotipy.readthedocs.io/en/latest/. Read and follow the installation instructions. See also https://github.com/plamere/spotipy
- Alzheimer’s: Look at sample data (http://adni.loni.usc.edu/data-samples/) and see if you have the appropriate Python libraries to read the data.
- Recommendations: download data (https://www.yelp.com/dataset/challenge) in one json object per line format. Load business.json and visualize.

# Sports: data collection

See the starter code in a-2017

1. Choose a sport
2. Scrape a single game's boxscore data for play-by-play
3. Pull off the url's for each game from a schedule for a full season
4. Scrape all games' Boxscore data for play-by-play
5. For one game, parse the events in the play-by-play data into features (lots of indicator variables, presumably, being sure to keep track of player 'IDs' in some way)
6. Expand this parsing and feature creation for a full season
7. Start attempting to fit a win probability model based on the features created. See references in project description

*Suggestion: start with the latest complete season of data for your chosen sport.*

# Crime: data collection

See an example code in a-2017, merging data from different sources

At the FBI site (https://ucr.fbi.gov/ucr-publications), scrape and save text on violent crimes and murders. Then for each year:

1. scrape the murder data for each Metropolitan Statistical Area (MSA) from the FBI site.
2. download the Census data (https://factfinder.census.gov/faces/nav/jsf/pages/index.xhtml) for each MSA with your your chosen set of predictors.
3. combine the murder data and Census data into a table. Be sure to check that the MSAs from the FBI data and the MSAs from the Census data are in the same order (.join, .merge may be useful). Note that the table format of the Census data is somewhat more complex than a simple table. You may need to do some data wrangling.

# Crime: data collection

Below is a snapshot of how the first three rows of the 2016 table could look, if one were only interested in gender and total population.

Table 1: Data from 2016

| M.S.A. | Pop. Tot. (FBI) | Murder Tot. | Pop. Tot. (CB) | Male (CB) | Female (CB) |
|---|---|---|---|---|---|
| Abilene, TX | 169,885 | 11 | 170,860 | 87,459 | 83,401 |
| Akron, OH | 703,561 | 41 | 702,221 | 341,200 | 361,021 |
| Albany, GA | 152,566 | 18 | 152,506 | 72,073 | 80,433 |

# Alzheimer's Disease: data collection

Data is from http://adni.loni.usc.edu/. To access this data, you need to first register https://ida.loni.usc.edu/collaboration/access/commonNoAccess.jsp;jsessionid=B315479616DFD26F331A6DE917E194D7.

1. Review the Data Use Agreement and agree to the terms
2. Complete the four steps to sign up
3. Follow instructions in the confirmation email – *you will be able to download the data within a week*
4. While you're waiting for access the full data, go to the Projects folder in a-2017 and download a partial data set of Alzheimer's data.

# Recommendations: data collection

1. Download data in one json object per line format.  See https://www.yelp.com/dataset/documentation/json
2. review.json is really big, user.json is smaller, and business.json is even smaller
3. Start by loading business.json into pandas (pd.Dataframe.from_json) and do some EDA
4. For review.json, consider taking a random sample from it
5. Constructing the training, testing, and validations sets may be challenging. You'll want to subsample the jsons for reasonable run times – make sure a given user is represented in all three jsons

# Spotify

1. Install spotipy.  If you don't have pip, google and get it.
2. Work through introductory examples at https://github.com/plamere/spotipy
3. The first few rows/columns of your data may look like this:

| | acousticness | added_date | danceability | duration | energy | followers | instrumentalness | key | liveness | loudness |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.053328658 | 2016-12-09T14:58:03Z | 0.4556 | 255623.1667 | 0.83645 | 425688 | 0.040480439 | 5.483333333 | 0.205875 | -6.542016667 |
| 1 | 0.032499882 | 2016-12-12T08:37:01Z | 0.471827586 | 210527.8621 | 0.909431034 | 193874 | 0.040731536 | 4.965517241 | 0.194610345 | -4.557551724 |
| 2 | 0.893902439 | 2017-02-16T06:07:09Z | 0.607926829 | 187029.439 | 0.162643902 | 14474 | 0.821707317 | 5.56097561 | 0.123239024 | -17.06192683 |
| 3 | 0.593163793 | 2017-02-09T13:45:42Z | 0.547465517 | 247844.5517 | 0.323248276 | 204511 | 0.37078869 | 5.74137931 | 0.117093103 | -12.64267241 |