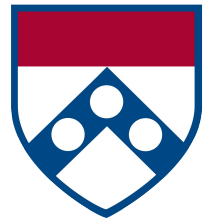
Wharton PhD Tech Camp

Session 6

Alex Miller

Ph.D. Student, Information Systems

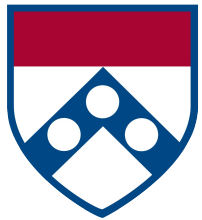
Wharton, OID



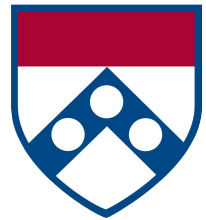
Goals for today

- Things to be aware of when scraping
- More involved scraping walkthrough
- Regex Intro
- Lab Exercise

Walkthroughs



Dark Side of Scraping



Things to be aware of!

- Many (most!) sites don't like scrapers!
  - Do your best to be respectful
  - There *are* ways around most website blockages
    - CORS, UserAgent, Even Logins
  - Legality is kind of up in air right now
    - https://arstechnica.com/tech-policy/2017/07/linkedin-its-illegal-to-scrape-our-website-without-permission/

- You can still scrape, but do it intelligently!

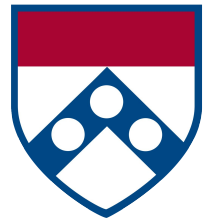
Best Practices

- Use available API whenever possible
- Do not be a bad citizen of the internet
  - Try to respect
- Do not scrape extensively from a Penn/Wharton IP address
  - You can get the whole network blocked!
  - Just depends on the website
    - Google vs. AppAnnie

Best Practices

- For large scraping projects:
  - If you’re comfortable with cloud computing, just fire up some small instances and scrape from there
  - Otherwise just scrape from home
- Minimize the number of requests you have to make
- Download (i.e., save) HTML offline first, parse later
  - This not only requires fewer requests to the web, but also allows you to parse things you didn’t realize were there

Circumventing Common Anti-Scraping Methods



Rate Limiting

- Do NOT make hundreds of requests to the same website in a short period of time
  - This is an easy way for website to detect abnormal behavior
- Iterate of your set of URLs, use Python’s “time.sleep” function to pause between requests
  - This is also just the polite thing to do

Spoofing HTTP Headers

- User agents are a header field passed in most HTTP requests
  - http://www.browser-info.net/useragents#most-popular-user-agents
- When you make a request from Python (or other languages), your code automatically identifies itself:
  - `User-Agent: python-requests/2.9.1`
- Easiest way for websites to block you → Easiest to get around!

```
import requests
spoofed_ua = "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.1; FSL 7.0.6.01001)"
r = requests.get(url, headers={"User-Agent": spoofed_au})
```

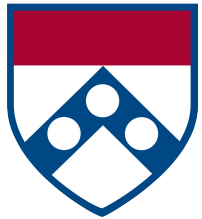
Logins

- Use Python requests “session” functionality
  - Lots of examples
    - https://stackoverflow.com/questions/24102064/staying-logged-in-using-requests-and-python
  - Be careful, as you will be identifiable

Last word of caution

- Almost none of these techniques will work for scraping the largest of websites (Google, Facebook, etc.)
- Don’t poke the sleeping bear!

RegEx



Regular Expressions

- (Almost) universal syntax for searching raw text
  - There are some minor variations between “flavors” of regex
  - Python’s regex format is fairly standard
- Most implementations are **very** fast
  - You can search very large documents very quickly
- Useful for:
  - Counting words/substrings in longer strings
  - Formatting strings
  - Finding all examples that match a pattern
- Essentially your typical “find/search” function on steroids

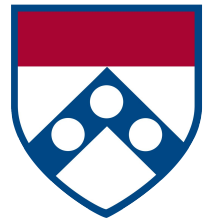
Regular Expressions

- Uses special characters:
  - http://www.regular-expressions.info/refcharacters.html
- Quick example
  - https://regex101.com/
    - http://www.presidency.ucsb.edu/ws/index.php?pid=111711
    - \[.*\]
    - [A-Z]{4,15}:
    - [A-Z]{4,15}:(.|\n)+?(?=[A-Z]{4,15}:)
- Online tutorial
  - https://regexone.com/
    - Very good for beginners

Regex in Python

- https://developers.google.com/edu/python/regular-expressions
- Notebook examples
  - Findall
  - Substr
  - Clean text
- Most importantly:
  - Just know it exists and is typically the “correct” way to search/match/extract data from raw text
  - You’ll always have Google

Scraping/Regex Exercise



Exercise

- Follow instructions at
  - https://github.com/alexmill/techcamp_week1/blob/master/session6/exercises.md