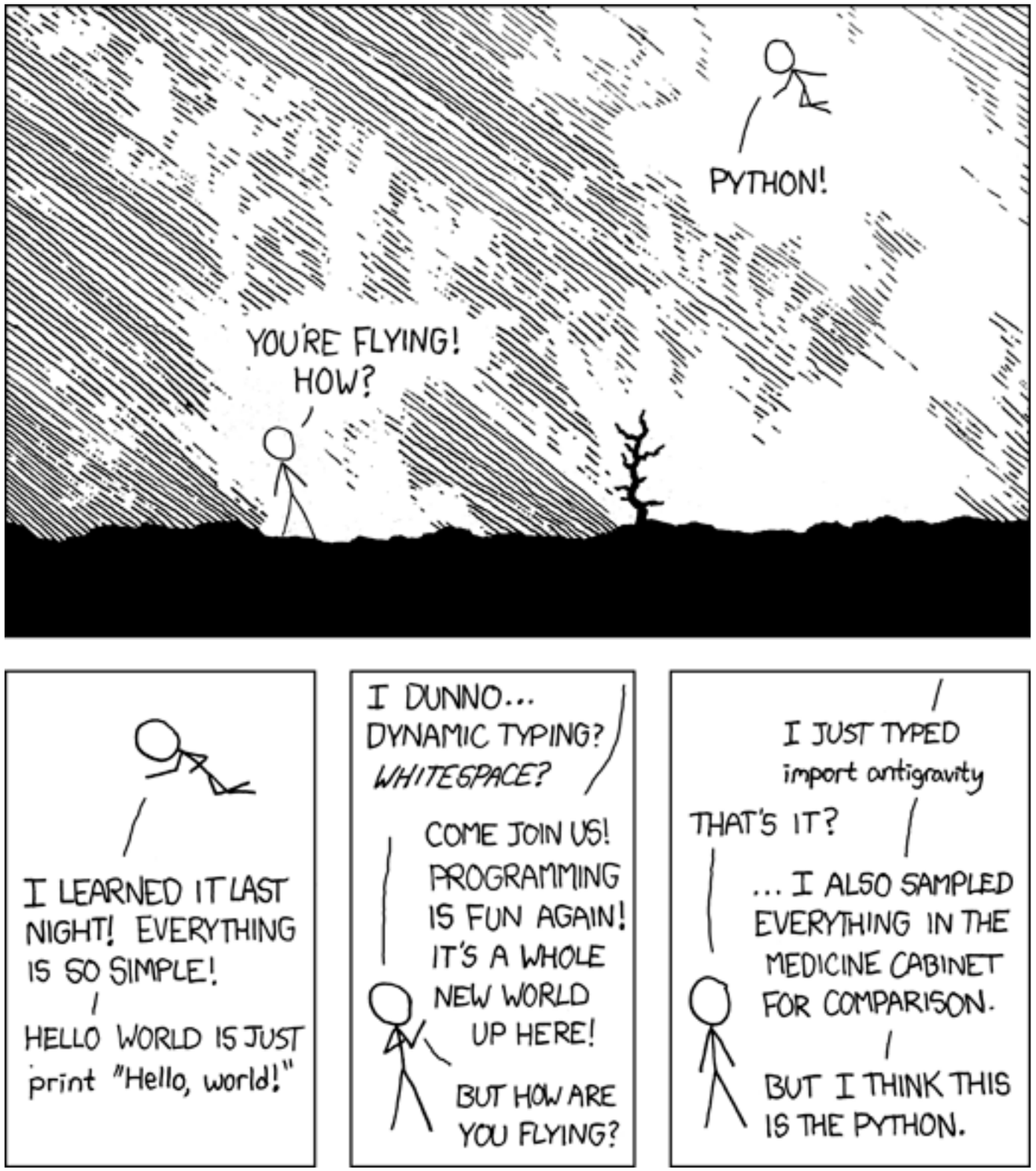
Python Evangelism

And Other Potentially Useful Information

11 April 2018

Outline

- Introduction to Python
- Survey of Useful Python Tools
- Computing in Python
- Scraping Tutorial
- Research in Python



Source: https://www.xkcd.com/353/

Why I Use Python

- Easy to code and test code
- Library available for almost any coding goal
- Well-documented trouble shooting
- The perfect glue language in any other case

Getting Started

- Code Academy

- Most Python documentation

- Quantitative Economics program by John Stachurski and Tom Sargent: https://quantecon.org/notebooks.html

- https://shapiromh.github.io - Walkthroughs of scraping examples

Ways to Use Python

1. Terminal Script (Hello, World!)
2. Interactive Python Terminal (IPython)
3. Notebook format (Jupyter Notebook)

Sample of Tools

Sample of Tools

- Data Analysis (Pandas, Statsmodels)
- Numerical Computing (Numpy/Scipy)
- Graphing (Matplotlib, plotly among others)
- Mapping / Geography (shapely among others)
- Machine Learning (tensorflow, scikit-learn)
- Bayesian Estimation (pymc)

Research in Python

Demo

Scraping

Basic Tools

- Any browser
- Python (and these modules for retrieval and parsing):
  - Requests
  - Selenium
  - Beautiful Soup
  - Regex (“re”)
  - Pandas (data management)

Some Terms

- Page Source
- HTML Tag
- HTTP Requests
- (RESTful) Application Programming Interface (API)
- Web Traffic Tools

Example:
Scraping Box Office Mojo

Steps

1. Figure out how to get to the site with data
2. Determine how to access data on page

Steps

1. Figure out how to get to the site with data
2. Determine how to access data on page
    - Here most data is cleanly laid out in a table
    - Beautiful Soup will be a great tool in these cases
3. Wrap Python (or language of choice) around the problem

Other Hints

- When parsing, use well-documented tags to your advantage
  - Unique attributes of a tag can uniquely identify a part of the webpage
  - E.g: <table border="0" cellspacing="1" cellpadding="5">
  - Use the search function in web inspector to verify it is unique

Demo

Example:
Scraping Charging Station Usage

Steps

1. Figure out how to get to the site with data

    - Site is a pain! Sad! Find the API

2. Determine how best to access data on page

    - Data on site is a pain! Find the API…

3. Hunt for the API

API Hunting

- The "network" tab of web inspectors records all resources loaded by the page
- Usually look for **XHR** type resources used to load data from servers
  - Inspector yields the url, "**Request URL**"
  - In some cases the base url is everything before a "?"
  - **Params** are variables determining what information is returned
  - **Response** shows you what the request will return
  - Experiment to figure out what params should be manipulated

API Hunting (Old Site)

- Example, base url and parameters

https://na.chargepoint.com/dashboard/getChargeSpots?
&lat=34.0522342&lng=-118.2436849&ne_lat=34.094891308785547&
ne_lng=-118.13450826425782&sw_lat=34.009555616869 54&sw_lng
=-118.35286153574219&user_lat=0&user_lng=0&search_lat=34.052
2342&search_lng=-118.2436849&scWidth=1320&scHeight=798&sor
t_by=distance&f_estimationfee=false&driver_filters=false&f_available
=false&f_free=false&f_l1=false&f_l2=false&f_chademo=false&f_saec
ombo=false&f_tesla=false&f_cp=false&f_blink=false&f_semacharge
=false&f_evgo=false&community=true&non_community=true&show_
mode2_only=false&show_mode1_only=false&_=1459222733665

API Hunting (New Site)

- Example, base url and parameters

https://mc.chargepoint.com/map-prod/get?{%22station_list%22:
{%22page_offset%22:%22%22,%22sort_by%22:%22distance%22,%22scree
n_width%22:451,%22ne_lat%22:41.2962962963,%22ne_lon%22:-123.37333
333,%22sw_lat%22:40.9444444444,%22sw_lon%22:-124.4,%22page_size%
22:100,%22screen_height%22:764,%22filter%22:
{%22connector_l1%22:false,%22connector_l2%22:false,
%22is_bmw_dc_program%22:false,%22is_nctc_program%22:false,
%22connector_chademo%22:false,%22connector_combo%22:false,
%22connector_tesla%22:false,%22price_free%22:false,
%22status_available%22:false,%22status_available%22:false,
%22network_chargepoint%22:false,%22network_blink%22:false,
%22network_semacharge%22:false,%22network_evgo%22:false,
%22connector_l2_nema_1450%22:false,%22connector_l2_tesla%22:false},
%22include_map_bound%22:true},%22user_id%22:0}

API Hunting

- The url request may be a file, html, in this case JSON
- How best to extract the data depends on what the request returns…
  - Pray for JSON
  - HTML can be parsed like in the previous example

- **NB:** Make sure the website does not explain its API before searching

Review Approach

1. Check for a website-provided API

2. Check for an API by examining web traffic

3. All else fails, download page sources and figure out how to parse

Levels of Frustration

1. **Static urls directing to desired data**
2. API provided by the website
3. **API hidden by the website**

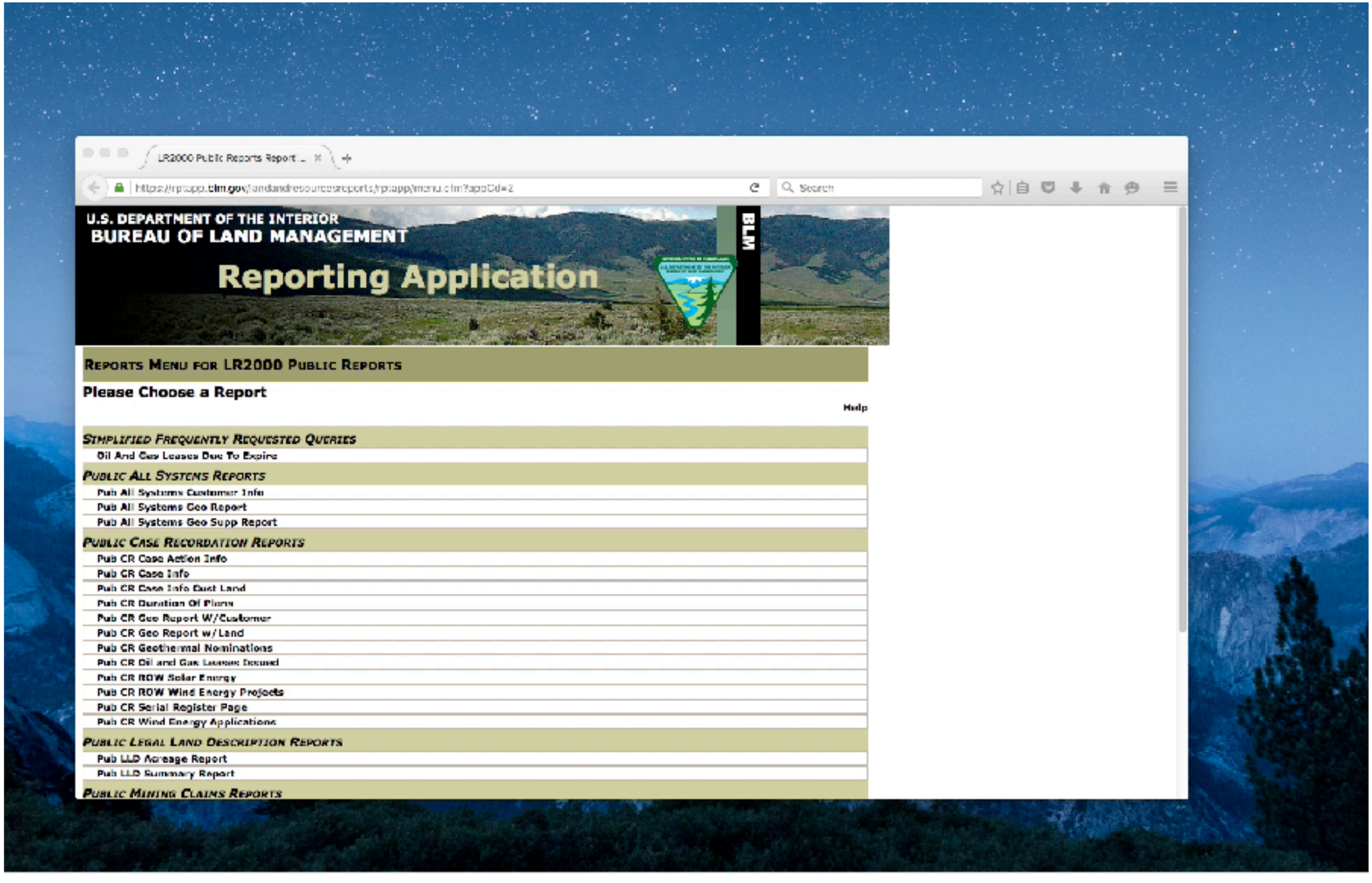
 API completely obscured

- A web driver required to automate scraping
- Try Selenium

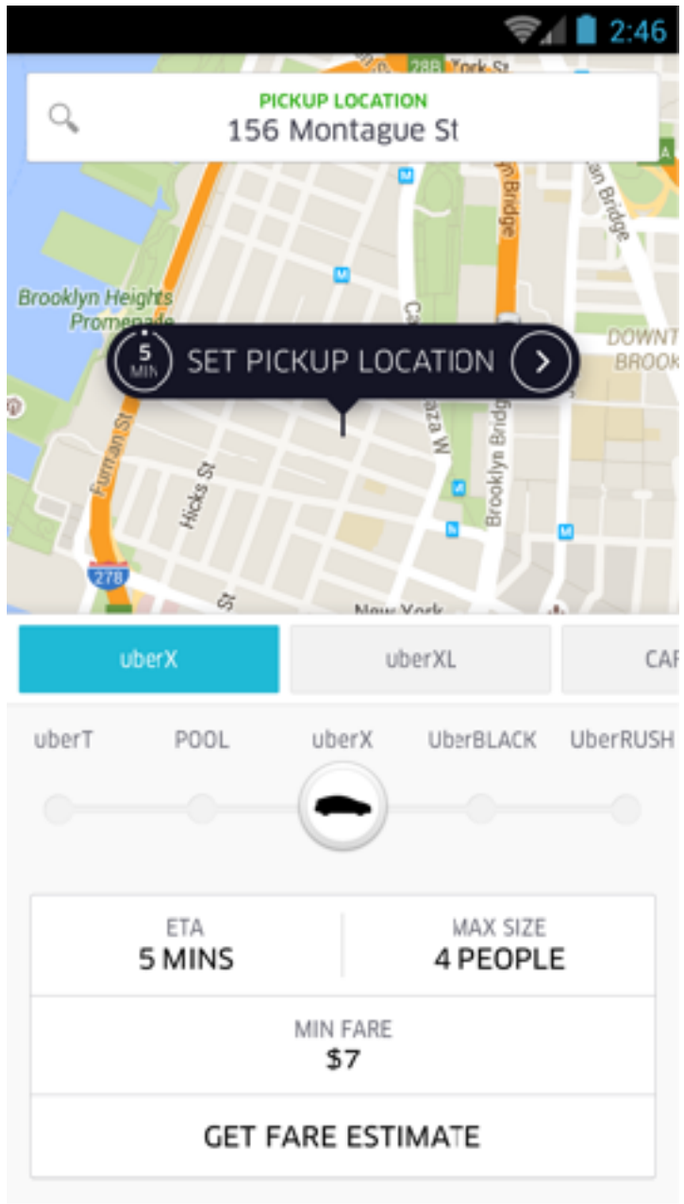
Advanced

- Javascript can screw up everything
- Selenium is an easy way to get around most complications
- Most network traffic can be captured using **proxies**
    - **Charles Proxy** is a cheap program to intercept non web-page traffic
    - Works similarly to examples explored
- Can even automate simulated smart phones with a bit of work

Selenium Demo



Automating a Smart Phone



Note: Automated through Genymotion, Appium (like Selenium for Android)

If you see it, you can scrape it!

More Research in Python

Demo