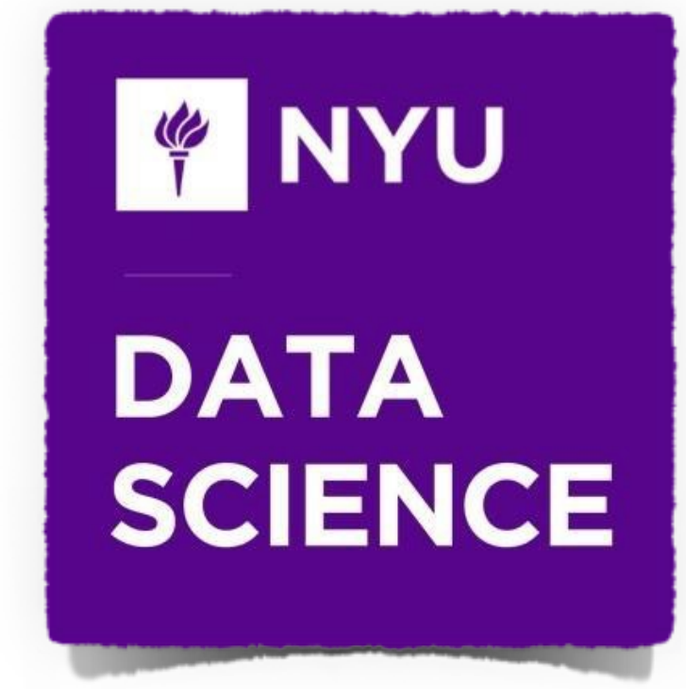
https://github.com/bmtgoncalves/LDSSS17

Lviv Data Science Summer School 2017
Mining Geolocated Datasets

Bruno Gonçalves
www.bgoncalves.com



Requirements

https://github.com/bmtgoncalves/LDSSS17

ANACONDA®

python

3.5

urllib

https://docs.python.org/3/library/urllib.html

- Extensible library for opening and manipulating **URLs**
- https://foursquare.com/tyayayayaa/checkin/5304b652498e734439d8711f?s=ScMqmpSLg1buhGXQicDJS4A_FVY&ref=tw
  - https <- protocol
  - foursquare.com <- server
  - /tyayayayaa/checkin/5304b652498e734439d8711f <- resource within server
  - s=ScMqmpSLg1buhGXQicDJS4A_FVY&ref=tw <- Query string

```python
from urllib import parse

url = "https://foursquare.com/tyayayayaa/checkin/5304b652498e734439d8711f?s=ScMqmpSLg1buhGXQicDJS4A_FVY&ref=tw"

parsed = parse.urlparse(url)
query = parsed.query
query_dict = parse.parse_qs(query)

print(parsed)
print(query_dict)
```

urllib_parse.py

urllib

https://docs.python.org/3/library/urllib.html

- urllib2.urlopen(url) opens a url for reading and returns a "file handle"-like object
- Information about the webpage can be obtained with the .info() method in the form of an HTTPMessage
- The HTTPMessage object obeys the usual Python dictionary interface
- The .geturl() method returns the final location of the webpage.
  .urlopen() follows redirects until it connects with the final content.
- .getcode() returns the status code of the call
  - 200 OK
  - 404 File Not Found
  - 500 Internal Server Error

Challenge - urllib

https://docs.python.org/3/library/urllib.html

- Find the final location of the shortened url:

http://bit.ly/GoogleScholar

- and access the headers and info of the request

```python
from urllib import request

url = "http://bit.ly/GoogleScholar"

webpage = request.urlopen(url)
code = webpage.getcode()
info = webpage.info()

headers  = info

new_url = webpage.geturl()

print(url, "redirected to", new_url)
```

urllib_request.py

posixpath

http://docs.python.org/3/library/os.path.html

- Manipulate paths in a POSIX operating system
- Also useful to extract information from remote resource paths
- Aliased to os.path if your operating systems is POSIX

- https://foursquare.com/tyayayayaa/checkin/5304b652498e734439d8711f
  -> Path in remote filesystem
- .basename(path) -> returns the file name (if there is one) 5304b652498e734439d8711f
- .dirname(path) -> return the directory portion /tyayayayaa/checkin

posixpath

http://docs.python.org/3/library/os.path.html

```
from urllib import parse
import posixpath

url = "https://foursquare.com/tyayayayaa/checkin/5304b652498e734439d8711f?s=ScMqmpSLg1buhGXQicDJS4A_FVY&ref=tw"

parsed = parse.urlparse(url)
filename = posixpath.basename(parsed.path)
directory = posixpath.dirname(parsed.path)

print(filename, directory)
```


posixpath_demo.py

requests

http://requests.readthedocs.org/en/latest/

- "HTTP for Humans" - Simplified HTTP requests:
    - authentication (basic authentication, OAuth1, OAuth2, etc) [See next lecture]
    - header manipulation
    - error handling
    - etc...

requests

http://requests.readthedocs.org/en/latest/

- .get(url) open the given url for reading and returns a Response
- Response.status_code is a field that contains the calls status code
- Response.headers is a dict containing all the returned headers
- Response.text is a field that contains the content of the returned page
- Response.url contains the final url after all redirections
- Response.json() parses a JSON response (throws a JSONDecodeError exception if response is not valid JSON). Check “content-type” header field.

requests

http://requests.readthedocs.org/en/latest/

```
import requests
from bs4 import BeautifulSoup
from pprint import pprint

url = "http://bit.ly/GoogleScholar"

req = requests.get(url)
print("Status code:", req.status_code)
print("Server Header Information:")
pprint(req.headers)

new_url = req.url

print(url, "redirected to", new_url)
```


requests_demo.py

json

https://docs.python.org/3/library/json.html

- JavaScript Object Notation - Serialization format originally developed for Javascript
- Currently widely accepted format for data dissemination
- Most languages have excellent libraries to handle it
- json.loads(obj_str) - load JSON data from a string - returns native Python object
- json.load(fp) - load JSON data from a file handle - returns native Python object
- json.dumps(obj) - convert JSON data to a string
- json.dump(obj, fp) - write the string version of obj to the file handle fp

Challenge - json

https://docs.python.org/3/library/json.html

- Access the JSON file:

http://www.bgoncalves.com/test.json

- and extract all the friend pairs

```
import requests
import json

url = "http://www.bgoncalves.com/test.json"

request = requests.get(url)

data = json.loads(request.text)

for user in data:
    name = user["name"]

    for friend in user["friends"]:
        print(name, "->", friend["name"])
```

json_demo.py

Basic Structure of a web page

http://www.w3schools.com/tags/

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>CSS Basics: A Cool Button</title>

<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
</head>

<body>

    <div id="container">
        <a href="#" class="btn">Push the button</a>
    </div>

</body>
</html>
```

Doctype

Page title

Link to CSS stylesheet

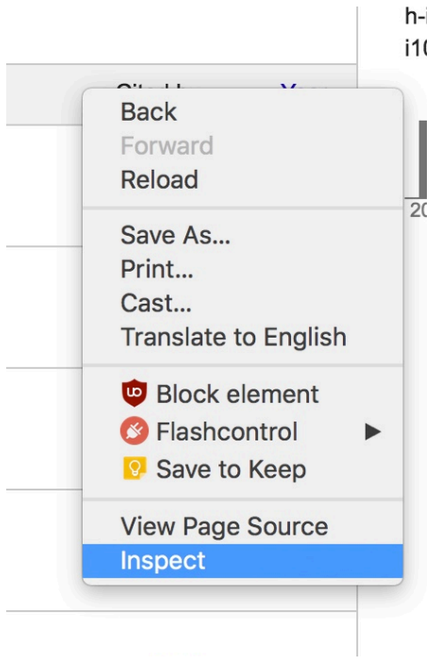
Container div to centre things up

Anchor with class of "btn"

- Tree-like structure (DOM)
- Nested <tags> with attributes and content
- Two main sections under <html>:
    - <head> - meta data and resource location
    - <body> - page contents

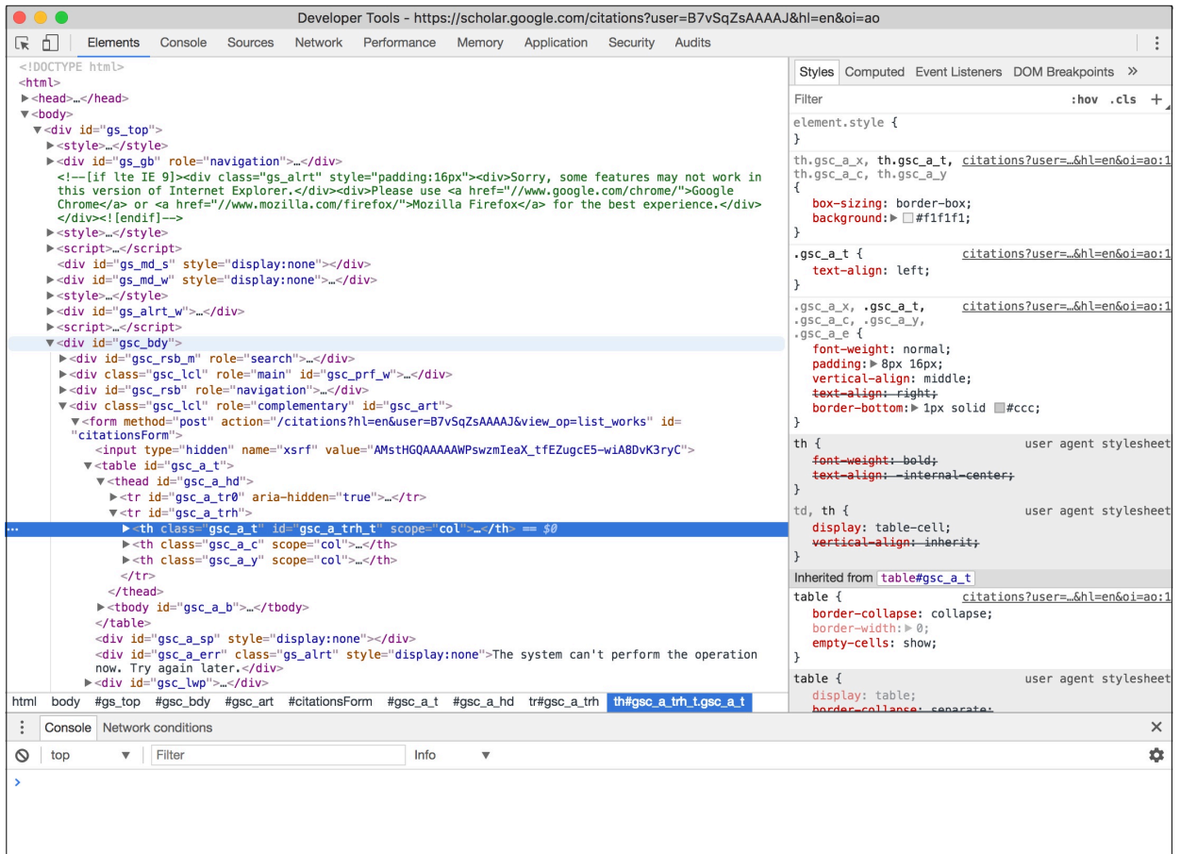
Chrome Developer Tools

- Extremely powerful and intuitive set of tools



- Comes standard with Google Chrome. Just right click anywhere on the page and select "Inspect"

- Allows you to interactively change the DOM of any "live" webpage and find which element corresponds to which part of the page.



Demo

BeautifulSoup

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

- Parses html and xml files into a tree.
- **BeautifulSoup(page)** where page is a string or a "file handle"-like object
- BeautifulSoup parses the contents of the page and returns a BeautifulSoup object, corresponding to the root of the document tree.
- Each leaf of the tree is a Tag object:
  - can be used as dicts to access tag attributes,
  - contains pointers to children (**.findChildren()**), siblings (**.findSiblings()**) and parent (**.findParent()**)
  - can be accessed recursively by name (**head.title.content**)
  - modifying the contents of a tag modifies the contents of the document

BeautifulSoup

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
import requests
from bs4 import BeautifulSoup

url = "http://www.bgoncalves.com/page.html"

request = requests.get(url)

soup = BeautifulSoup(request.text, "lxml")

print("The title tag is", soup.title)
print("The id of the div is", soup.div["id"])

soup.div["id"] = "new_id"

print("And now it's", soup.body.div["id"])
```

- .findAll() returns a list of all tags matching a certain criteria
  - .findAll(name="a") find all "a" tags (links)
  - .findAll(name=["a", "div"]) find all "a" and "div" tags
  - .findAll(attrs = {"class": "btn"}) find all tags with class "btn", regardless of tag name
  - .findAll(name="a", attrs = {"class": "btn"}, limit=2) find the first two "a" tags with class "btn"

soup.py

BeautifulSoup

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
import requests
from bs4 import BeautifulSoup

url = "http://www.whoishostingthis.com/tools/user-agent/"

headers = {"User-agent" : "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101
Firefox/25.0"}

request_default = requests.get(url)
request_spoofed = requests.get(url, headers=headers)

soup_default = BeautifulSoup(request_default.text, "lxml")
soup_spoofed = BeautifulSoup(request_spoofed.text, "lxml")

print("Default:", soup_default.find(name="div", attrs={"class":"info-box user-agent"}).text)
print("Spoofed:", soup_spoofed.find(name="div", attrs={"class":"info-box user-agent"}).text)
```

- Some servers use the User-agent string to decide how to format the output
  - Correctly handle specific versions of web browsers
  - Provide lighter/simplified versions to users on their mobiles
  - Refusing access to automated tools, etc

requests_spoof.py

Challenge - BeautifulSoup

- Extract the title of Feynman's 100 most cited papers from Google Scholar

Richard Feynman

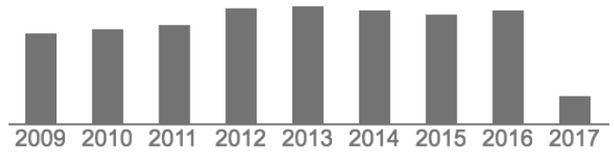
California Institute of Technology
quantum mechanics, quantum electrodynamics
No verified email

Follow

Google Scholar

Title 1–100	Cited by	Year
The Feynman lectures on physics RP Feynman, RB Leighton, M Sands, SB Treiman Physics Today 17, 45	14650 *	1964

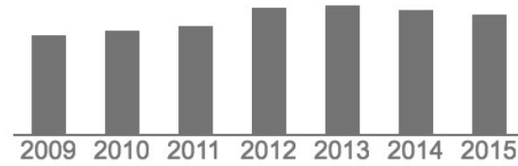
Citation indices	All	Since 2012
Citations	82210	21241
h-index	59	45
i10-index	93	72



Richard Feynman

California Institute of Technology
quantum mechanics, quantum electrodynamics
No verified email

Citation indices	All
Citations	82210
h-index	59
i10-index	93



Title 1–100	Cited by	Year
The Feynman lectures on physics RP Feynman, RB Leighton, M Sands, SB Treiman Physics Today 17, 45	14650 *	1964
Quantum mechanics and path integration RP Feynman, AR Hibbs McGraw–Hill	11256 *	1965
Simulating physics with computers RP Feynman International journal of theoretical physics 21 (6), 467-488	5715	1982
Space-time approach to non-relativistic quantum mech RP Feynman Reviews of Modern Physics 20 (2), 367	4146	1948
Forces in molecules RP Feynman Physical Review 56 (4), 340	3411	1939
There's plenty of room at the bottom RP Feynman Engineering and Science 23 (5), 22-36	3342	1960
Very high-energy collisions of hadrons RP Feynman Physical Review Letters 23 (24), 1415-1417	2801	1969
Theory of the Fermi interaction		

Richard Feynman - Google Sch ×

Secure https://scholar.google.com/citations?hl=en&user=B7vSqZsAAAAJ&view_op=list_wo

Images More...

Richard Feynman

California Institute of Technology

quantum mechanics, quantum electrodynamics

No verified email

td.gsc_a_t | 585 × 68

The Feynman lectures on physics
RP Feynman, RB Leighton, M Sands, SB Treiman
Physics Today 17, 45

Quantum mechanics and path integration
RP Feynman, AR Hibbs
McGraw–Hill

Simulating physics with computers
RP Feynman
International journal of theoretical physics 21 (6), 467-488

Space-time approach to non-relativistic quantum mechanics
RP Feynman
Reviews of Modern Physics 20 (2), 367

Forces in molecules
RP Feynman
Physical Review 56 (4), 340

There's plenty of room at the bottom
RP Feynman
Engineering and Science 23 (5), 22-36

Very high-energy collisions of hadrons
RP Feynman
Physical Review Letters 23 (24), 1415-1417

Theory of the Fermi interaction

Developer Tools - https://scholar.google.com/

Elements Console Sources Network Timeline Profiles

```
<!--[if lte IE 9]><div class="gs_alrt" style="padding:16px"><di
this version of Internet Explorer.</div><div>Please use <a href
Chrome</a> or <a href="//www.mozilla.com/firefox/">Mozilla Fire
</div><![endif]-->
▶<style>…</style>
▶<script>…</script>
 <div id="gs_md_s" style="display:none"></div>
▶<div id="gs_md_w" style="display:none">…</div>
▶<style>…</style>
▶<div id="gs_alrt_w">…</div>
▶<script>…</script>
▼<div id="gsc_bdy">
 ▶<div id="gsc_rsb_m" role="search">…</div>
 ▶<div class="gsc_lcl" role="main" id="gsc_prf_w">…</div>
 ▶<div id="gsc_rsb" role="navigation">…</div>
 ▼<div class="gsc_lcl" role="complementary" id="gsc_art">
  ▼<form method="post" action="/citations?hl=en&user=B7vSqZsAA
  "citationsForm">
    <input type="hidden" name="xsrf" value="AMstHGQAAAAAWPiWa
   ▼<table id="gsc_a_t">
    ▼<thead id="gsc_a_hd">
     ▶<tr id="gsc_a_tr0" aria-hidden="true">…</tr>
     ▶<tr id="gsc_a_trh">…</tr>
     </thead>
    ▼<tbody id="gsc_a_b">
     ▼<tr class="gsc_a_tr">
... ▼<td class="gsc_a_t"> == $0
       <a href="/citations?
       view_op=view_citation&hl=en&user=B7vSqZsAAAAJ&page
       x6o8ySG0sC" class="gsc_a_at">The Feynman lectures
       <div class="gs_gray">RP Feynman, RB Leighton, M Sa
      ▶<div class="gs_gray">…</div>
       </td>
      ▶<td class="gsc_a_c">…</td>
      ▶<td class="gsc_a_y">…</td>
       </tr>
     ▶<tr class="gsc_a_tr">…</tr>
     ▶<tr class="gsc_a_tr">…</tr>
     ▶<tr class="gsc_a_tr">…</tr>
     ▶<tr class="gsc_a_tr">…</tr>
     ▶<tr class="gsc_a_tr">…</tr>
```

html body #gs_top div#gsc_bdy div#gsc_art.gsc_lcl form#citationsForm table#g

Console Network conditions

top Preserve log

>

Challenge - BeautifulSoup

- Extract the title of Feynman's 100 most cited papers from Google Scholar

```
import requests
from bs4 import BeautifulSoup

url = "http://scholar.google.com/citations?hl=en&user=B7vSqZsAAAAJ&view_op=list_works&pagesize=100"

request = requests.get(url)
soup = BeautifulSoup(request.text, 'lxml')

table = soup.find("table", attrs={"id" : "gsc_a_t"})

for paper in table.findAll("td", attrs={"class": "gsc_a_t"}):
    print(paper.a.string)
```


feynman.py

pyquery

https://pyquery.readthedocs.io/en/latest/
See also: *https://www.w3schools.com/jquery/*

- Python version of the popular **jQuery** javascript package.
- More powerful than BeautifulSoup but also more complex.
- It defines three type of selectors:
  - element selector - retrieve all instances of a given HTML element (div, p, li, etc...)
  - #id selector - retrieve the element with id given by id
  - .class selector - retrieve all elements of a given class
- It also defines the usual jQuery pseudo-classes:
  - :first - first element
  - :last - last element
  - :even - even elements (0, 2, 4, ...)
  - :odd - odd element (1, 3, 5, ...)
  - :eq - a specific element (equals)
  - :lt - less than
  - :gt - greater than

pyQuery

https://pyquery.readthedocs.io/en/latest/

- pyQuery(url=url) or pyQuery(string) to parse a given external url of the html code in a specific string

- .attr("attr") returns a specific attribute of a given object.

- .addClass("bla") - add a css class

- .toggleClass("bla ble") - toggle class

- .removeClass("ble") - remove class

- .css("style": "value") - define css style value ("font-size", "15px")

- .items() - iterate over results

Challenge - pyQuery

https://pyquery.readthedocs.io/en/latest/

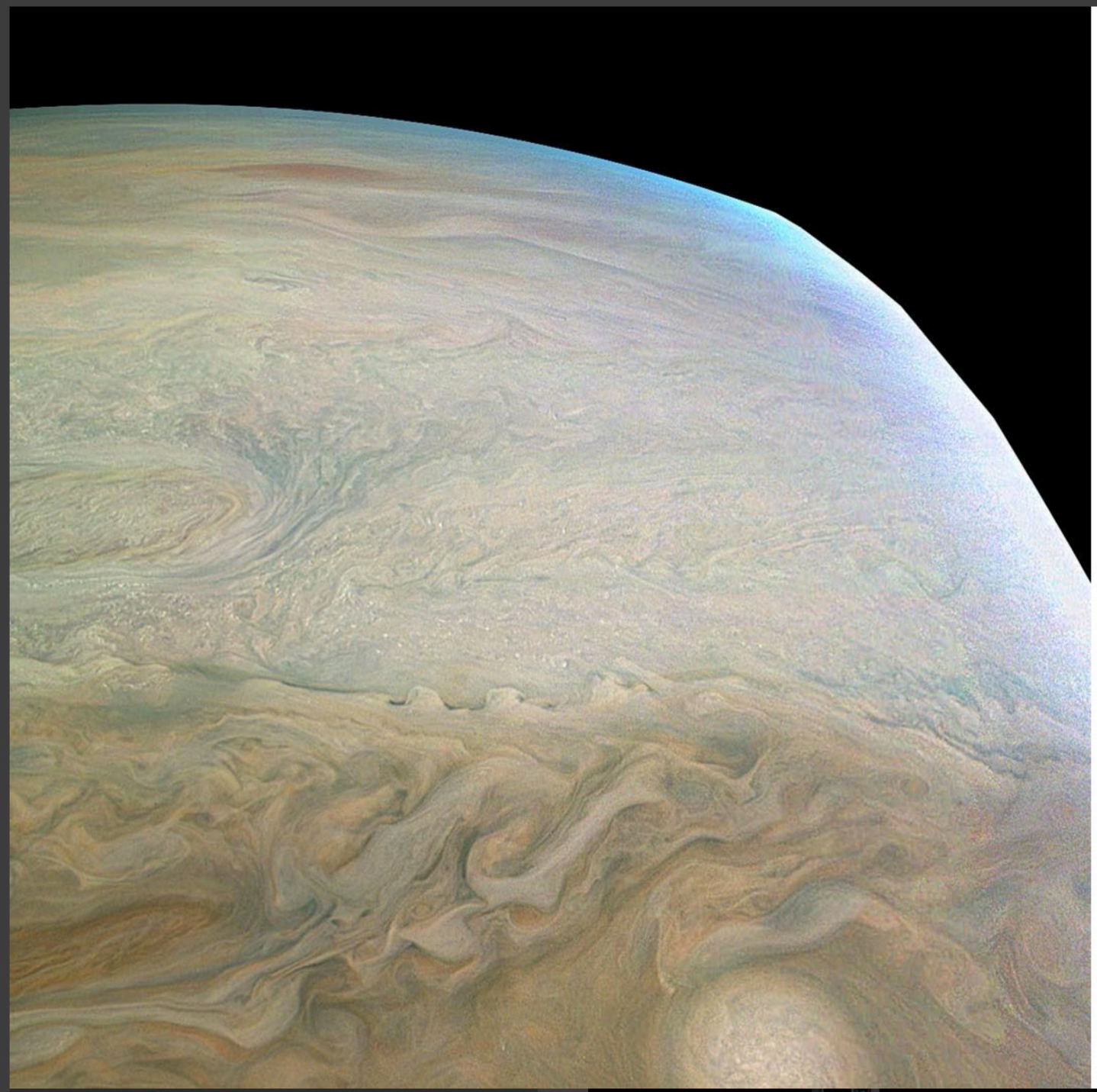
- Extract the title of Feynman's 100 most cited papers from Google Scholar, using pyQuery

```
from pyquery import PyQuery as pq

url = "http://scholar.google.com/citations?hl=en&user=B7vSqZsAAAAJ&view_op=list_works&pagesize=100"

doc = pq(url=url)
table = doc("table#gsc_a_t")

for row in table("td.gsc_a_t").items():
    print(row("a").text())
```


feynman_pyquery.py

Instagram

Search



nasa

Follow

nasa The edge of Jupiter: This enhanced color Jupiter image, taken by the JunoCam imager on our Juno spacecraft, showcases several interesting features on the apparent edge (limb) of the planet. Prior to Juno's fifth flyby over Jupiter's mysterious cloud tops, members of the public voted on which targets JunoCam should image. This picture captures not only a fascinating variety of textures in Jupiter's atmosphere, it also features three specific points of interest: "String of Pearls," "Between the Pearls," and "An Interesting Band Point." Also visible is what's known as the STB Spectre, a feature in Jupiter's South Temperate Belt where multiple atmospheric conditions appear to collide. Credits: NASA/JPL-Caltech/SwRI/MSSS/Bjorn Jonsson

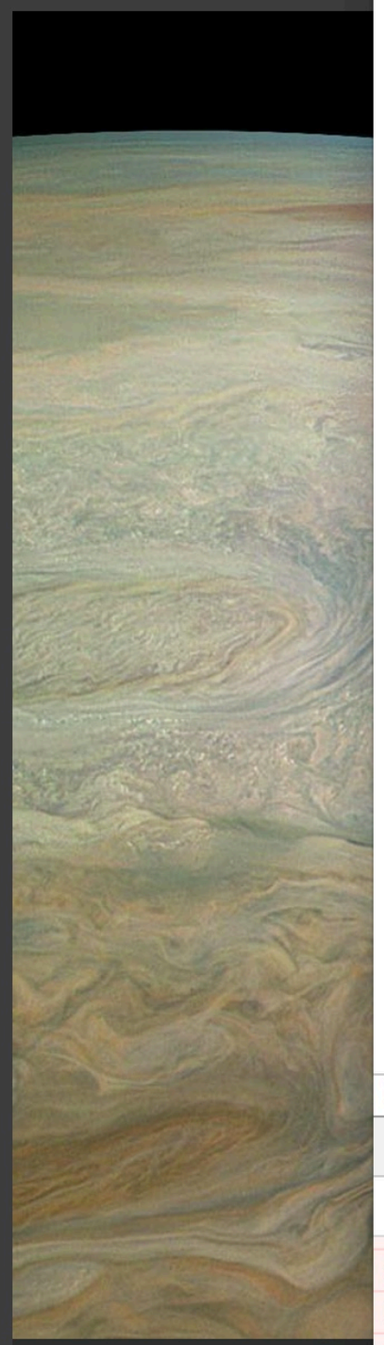
474,493 likes

22 HOURS AGO

Add a comment...

NASA (@nasa) • Instagram pho ×

Secure https://www.instagram.com/p/BTG1P



Developer Tools - https://www.instagram.com/p/BTG1PoGBbtf/?taken-by=nasa

Elements Console Sources Network Performance Memory Application Security Audits

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html lang="en" class="no-js lt-ie9 lt-ie8 lt-ie7 logged-in "> <![endif]-->
<!--[if IE 7]>         <html lang="en" class="no-js lt-ie9 lt-ie8 logged-in "> <![endif]-->
<!--[if IE 8]>         <html lang="en" class="no-js lt-ie9 logged-in "> <![endif]-->
<!--[if gt IE 8]><!-->
<html lang="en" class="js logged-in ">
  <!--<![endif]-->
▶<head>…</head>
▼<body class style="position: fixed; top: 0px; width: 100%;">
  ▶<span id="react-root" aria-hidden="true">…</span>
  ▼<script type="text/javascript">
      window._sharedData = {"activity_counts": {"comment_likes": 0, "comments": 0, "likes": 0,
      "relationships": 1, "usertags": 0}, "config": {"csrf_token": "e40abfde29df9262b16088fcfff6474d",
      "viewer": {"biography": "Data Science, Social Networks, Human Behavior", "external_url":
      "http://www.bgoncalves.com/", "full_name": "Bruno Gon\u00e7alves", "has_profile_pic": true, "id":
      "1557175974", "profile_pic_url": "https://scontent-lga3-1.cdninstagram.com/t51.2885-
      19/s150x150/12424858_1236025479746639_1884090780_a.jpg", "profile_pic_url_hd": "https://scontent-lga3-
      1.cdninstagram.com/t51.2885-19/s150x150/12424858_1236025479746639_1884090780_a.jpg", "username":
      "bmtgoncalves"}}, "country_code": "US", "language_code": "en", "entry_data": {"FeedPage": [{"graphql":
      {"user": {"id": "1557175974", "profile_pic_url": "https://scontent-lga3-1.cdninstagram.com/t51.2885-
      19/s150x150/12424858_1236025479746639_1884090780_a.jpg", "username": "bmtgoncalves",
      "edge_web_feed_timeline": {"page_info": {"has_next_page": true, "end_cursor":
      "KKoBARAACAIoABgAEAAIAAgACAAIAP_____f-df_9__9_3_v_-__e___f_____-__v9_____f_____-
      _____7__v7_____7-___9Tx53_V4zD-Xa4pTb___v_f9__3f_____9_7____3f3__9__Pf_____7vptd___-
      _v__7_9f__9_v__dn_b___v__dT_4___7__7x-02_iQigqEhIAWtNW_gvJWAA=="}, "edges": [{"node": {"__typename":
      "GraphImage", "id": "1496706242402831436", "dimensions": {"height": 1098, "width": 1080}, "display_url":
      "https://scontent-lga3-1.cdninstagram.com/t51.2885-
      15/e35/17883107_533293043523624_8520366081833959424_n.jpg", "is_video": false,
      "edge_media_to_tagged_user": {"edges": []}, "attribution": null, "shortcode":
      "BTFXmXHh8xM6dOiruq8YysbliMHol6GU94MtM00", "edge_media_to_caption": {"edges": [{"node": {"text": "The
      best bread is the homemade bread!"}}]}, "edge_media_to_comment": {"count": 1, "page_info":
      {"has_next_page": false, "end_cursor": null}, "edges": [{"node": {"id": "17877772381001154", "text":
      "Yes. Very much so", "created_at": 1492646725, "owner": {"id": "191871059", "profile_pic_url":
      "https://scontent-lga3-1.cdninstagram.com/t51.2885-
      19/s150x150/16789006_1232382500203049_7038774756111286272_a.jpg", "username": "datachick"}}}]},
      "comments_disabled": false, "taken_at_timestamp": 1492641309, "edge_media_preview_like": {"count": 6,
      "edges": [{"node": {"id": "4484922832", "profile_pic_url": "https://scontent-lga3-
      1.cdninstagram.com/t51.2885-19/s150x150/17076579_283081138789146_8579303371121360896_a.jpg", "username":
      "tgrigorina"}}, {"node": {"id": "2520241055", "profile_pic_url": "https://scontent-lga3-
      1.cdninstagram.com/t51.2885-19/s150x150/18011423_205831123252128_2373193220310958080_a.jpg", "username":
      "voivozeanucrina"}}, {"node": {"id": "1798923895", "profile_pic_url": "https://scontent-lga3-
      1.cdninstagram.com/t51.2885-19/s150x150/14736246_1077335775720807_965769911001415680_a.jpg", "username":
      "popaannamaria"}}, {"node": {"id": "1383131102", "profile_pic_url": "https://scontent-lga3-
      1.cdninstagram.com/t51.2885-19/s150x150/17817977_1207606412690515_8217228698831552512_a.jpg",
      "username": "cristina.rebegel"}}, {"node": {"id": "1412820387", "profile_pic_url": "https://scontent-
```

Styles Co

Filter

```
element.st
}
._ovg3g {
    bottom:
    right: 0
}
._icyx7, .
    left: 0;
    position
    top: 0;
}
#react-roo
header, ma
    -webkit-
    -ms-fle
    align-i
    border:
    box-siz
    display
    display
    display
    -webkit-
    -webkit-
    -ms-fle
    flex-di
    -ms-fle
    flex-sh
    margin:
    padding
    position
}
a, abbr, a
article, a
blockquote
dd, del, d
fieldset,
h3, h4, h5
ins, kbd,
ol, output
```

html body div div div div article._55zw1._p8mqr._lv5ky div._r3k3c div div._22yr2._rx3v8 div._ovg3g

Console Network conditions

top Filter Info

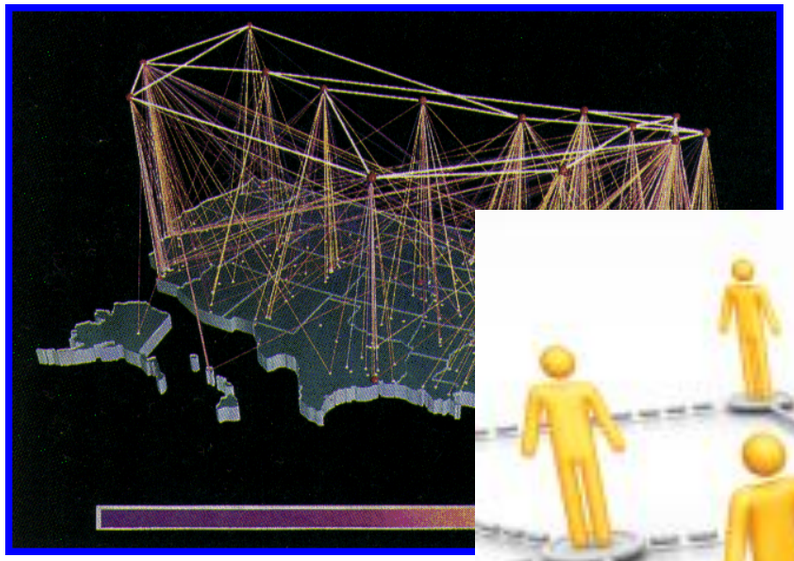
```
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load www.facebook.com/tr/?id=1425767024389221&ev=ViewContent&dl=https%3A%2F%2Fww…7ba79558d6baa209b7854015785a3b3f6
load resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load www.facebook.com/tr/?ev=6021483112529&dl=https%3A%2F%2Fwww.instagram.com%2F&rl=&if=false&ts=149277222427
resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
```

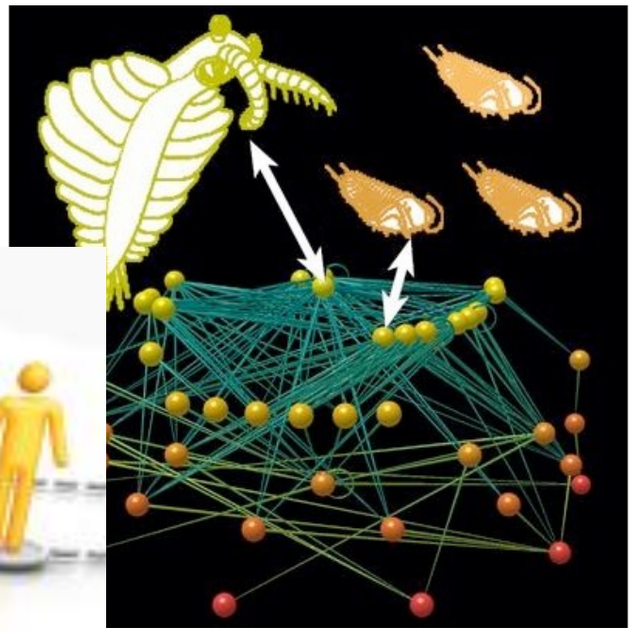
>

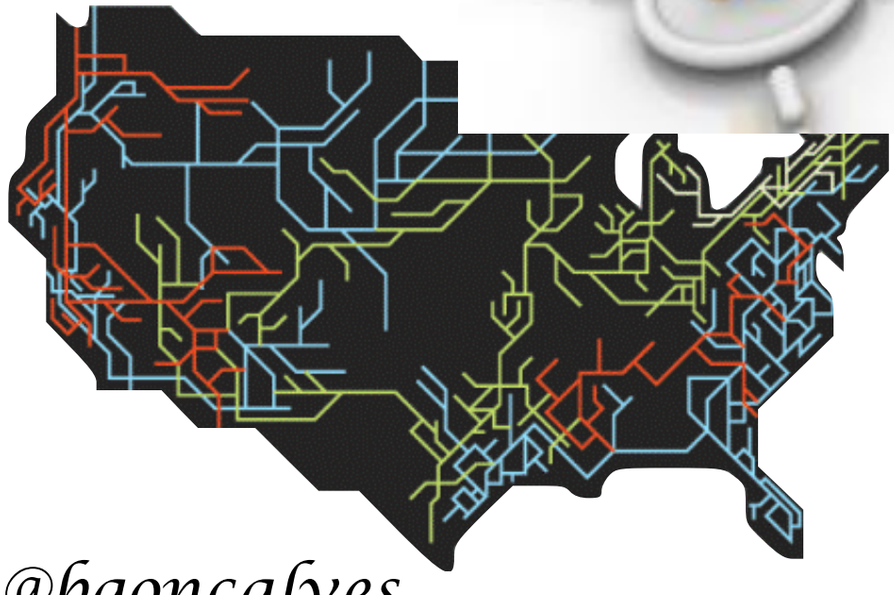
Instagram

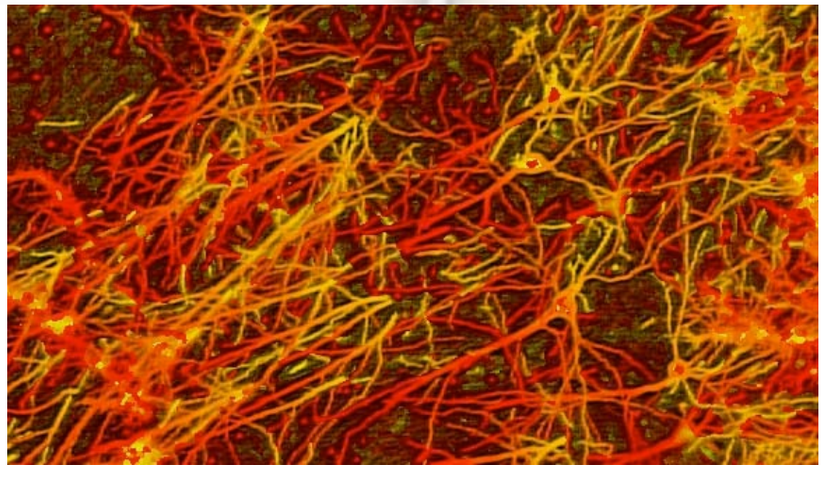
```
import requests
from bs4 import BeautifulSoup

url = "https://www.instagram.com/p/BTG1PoGBbtf/"

jsfuncs = {
    "true": True,
    "false": False,
    "null": None
}

magic_string = "window._sharedData = "
offset = len(magic_string)

page = requests.get(url).content
soup = BeautifulSoup(page, "lxml")

scripts = soup.findAll("script")

for script in scripts:
    if script.text.startswith(magic_string):
        text = script.text[offset:-1]
        data = eval(text, jsfuncs)
        break

comments = data["entry_data"]["PostPage"][0]["graphql"]["shortcode_media"]["edge_media_to_comment"]["edges"]

for comment in comments:
    print(comment["node"]["owner"]["username"], "==>", comment["node"]["text"].encode('utf8','replace').decode())
```

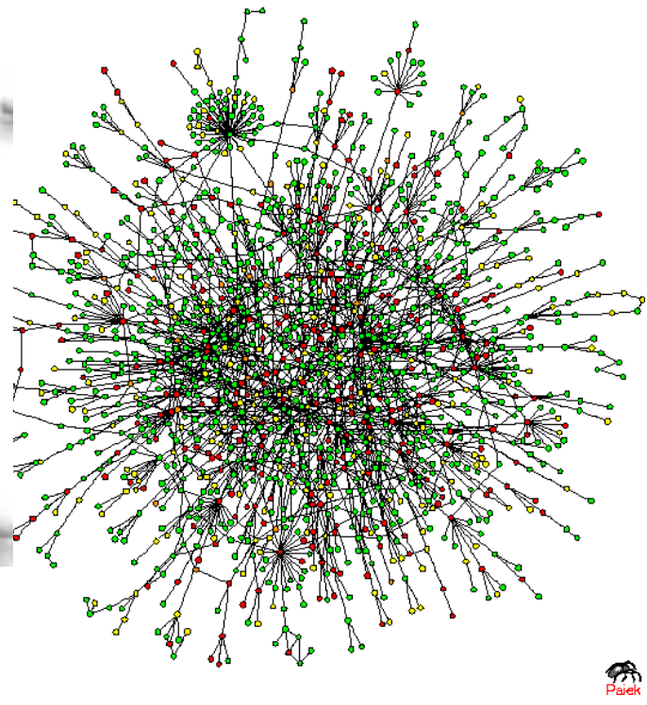
instagram.py



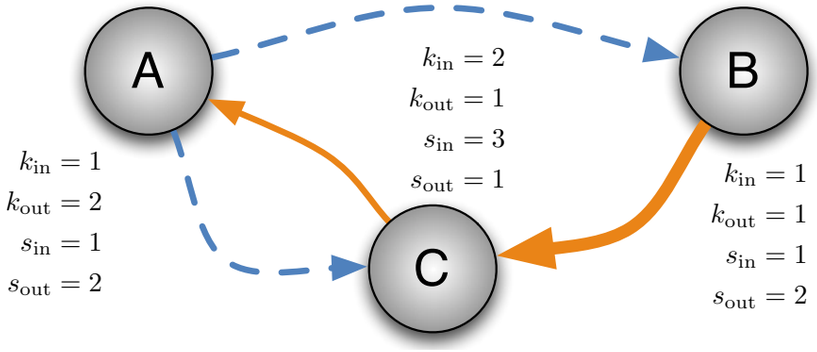


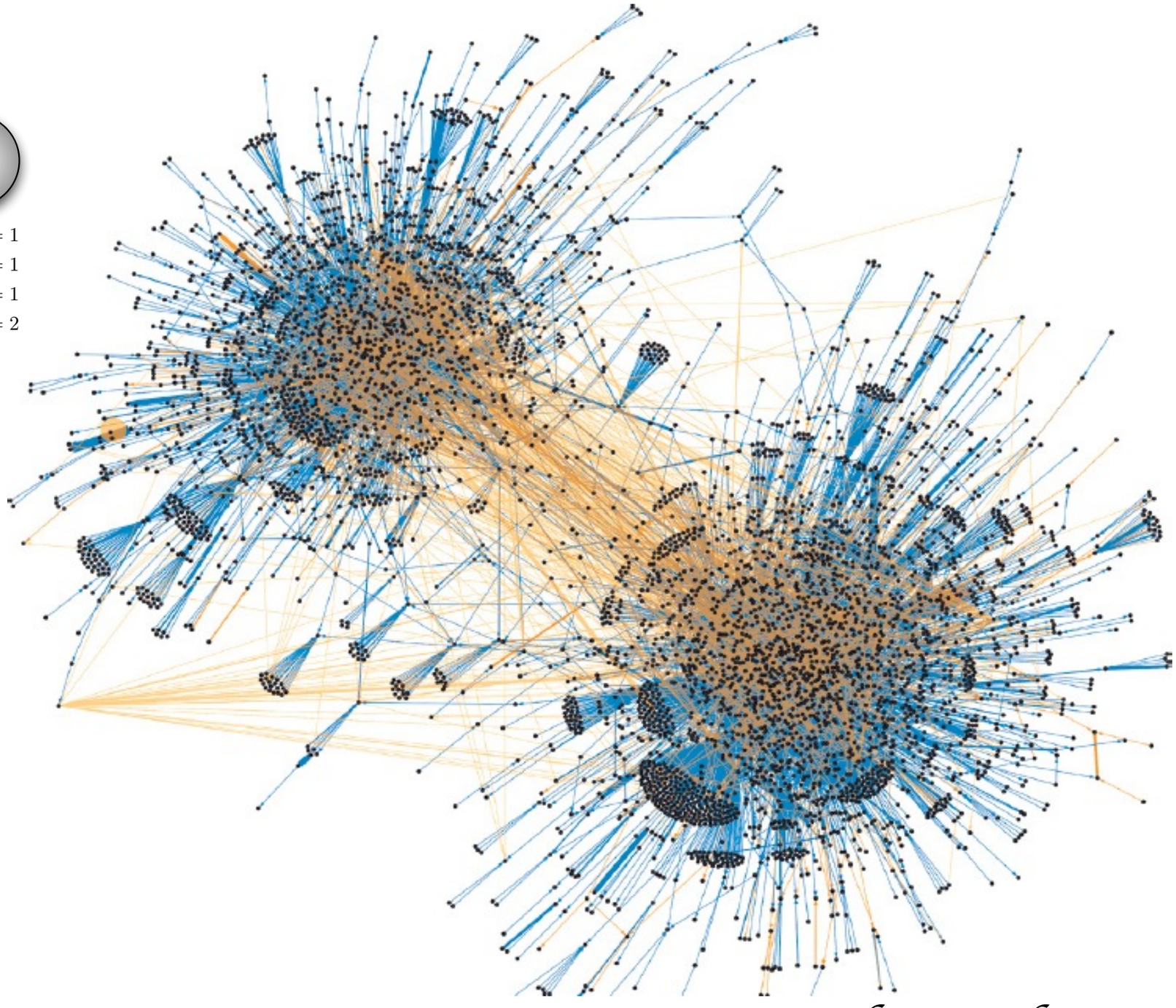





Palek

Information Flow





NetworkX

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Undirected
    - MultiGraph - Multi-edged Graph
    - MultiDiGraph - Directed Multigraph
- Similar interface for all types of graphs
- Nodes can be any type of Python object - Practical way to manage relationships

Growing Graphs

- .add_node(node_id) Add a single node with ID node_id
- .add_nodes_from() Add a list of node ids
- .add_edge(node_i, node_j) Adds an edge between node_i and node_j
- .add_edges_from() Adds a list of edges. Individual edges are represented by tuples
- .remove_node(node_id)/.remove_nodes_from() Removing a node removes all associated edges
- .remove_edge(node_i, node_j)/.remove_edges_from()

Graph Properties

- .nodes() Returns the list of nodes

- .edges() Returns the list of edges

- .degree() Returns a dict with each nodes degree .in_degree()/.out_degree() returns dicts with in/out degree for DiGraphs

- .is_connected() Returns true if the node is connected

- .is_weakly_connected()/.is_strongly_connected() for DiGraph

- .connected_components() A list of nodes for each connected component

NetworkX - Example

```
import networkx as NX
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt

def BarabasiAlbert(N=1000000):
    G = NX.Graph()

    nodes = range(N)
    G.add_nodes_from(nodes)

    edges = [0,1,1,2,2,0]

    for node_i in range(3, N):
        pos = np.random.randint(len(edges))
        node_j = edges[pos]

        edges.append(node_i)
        edges.append(node_j)

    edges = zip(nodes, edges[1::2])

    G.add_edges_from(edges)

    return G
```


networkx_demo.py

NetworkX - Example

```
import networkx as NX
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt

(…)

net = BarabasiAlbert()

degrees = net.degree()
Pk = np.array(list(Counter(degrees.values()).items()))

plt.loglog(Pk.T[0], Pk.T[1], 'b*')
plt.xlabel('k')
plt.ylabel('P[k]')
plt.savefig('Pk.png')
plt.close()

print("Number of nodes:", net.number_of_nodes())
print("Number of edges:", net.number_of_edges())
```


networkx_demo.py

Snowball Sampling

- Commonly used in Social Science and Computer Science

    1. Start with a single node (or small number of nodes)

    2. Get "friends" list

    3. For each friend get the "friend" list

    4. Repeat for a fixed number of layers or until enough users have been connected

- Generates a connected component from each seed

- Quickly generates a *lot* of data/API calls

Challenge - Snowball Sampling

- Write a function t
- from a given:
- using a Network

```python
import networkx as NX

def snowball(net, seed, max_depth = 3, maxnodes=1000):
    seen = set()
    queue = set()

    queue.add(seed)
    queue2 = set()

    for _ in range(max_depth+1):
        while queue:
            user_id = queue.pop()
            seen.add(user_id)

            NN = net.neighbors(user_id)

            for node in NN:
                if node not in seen:
                    queue2.add(node)

        queue.update(queue2)
        queue2 = set()

    return seen

net = NX.connected_watts_strogatz_graph(10000, 4, 0.01)
neve = snowball(net, 0)

print(neve)
```

snowball.py

Authentication

Authentication Methodologies

- Much of the content available online is only accessible to specific individuals for privacy, copyright protection, etc…

- Three main ways of authenticating users:

    - BasicAuth - The first and most basic one. Plain text user name and password sent to the server

    - OAuth 1 - Developed by a consortium of Industry leaders to provide transparent and secure authentication.

    - OAuth 2 - An improvement on OAuth 1 designed to allow users to more easily share they're content on social media, etc…

    - OpenID - A predecessor to OAuth that has gone out of favor.

BasicAuth

http://requests.readthedocs.org/en/latest/

- "The mother of all authentication protocols"
- Insecure but easy to use with standard implementations in all networking tools
- In particular, in requests:
  - requests.get(url, auth=("user", "pass")) open the given url and authenticate with username="user" and password="pass"

```
import requests
import sys

url = "http://httpbin.org/basic-auth/user/passwd"

request = requests.get(url, auth=("user", "passwd"))

if request.status_code != 200:
    print("Error found", request.get_code(),
file=sys.stderr)

content_type = request.headers["content-type"]

response = request.json()

if response["authenticated"]:
    print("Authentication Successful")
```

basic_auth.py

OAuth 1

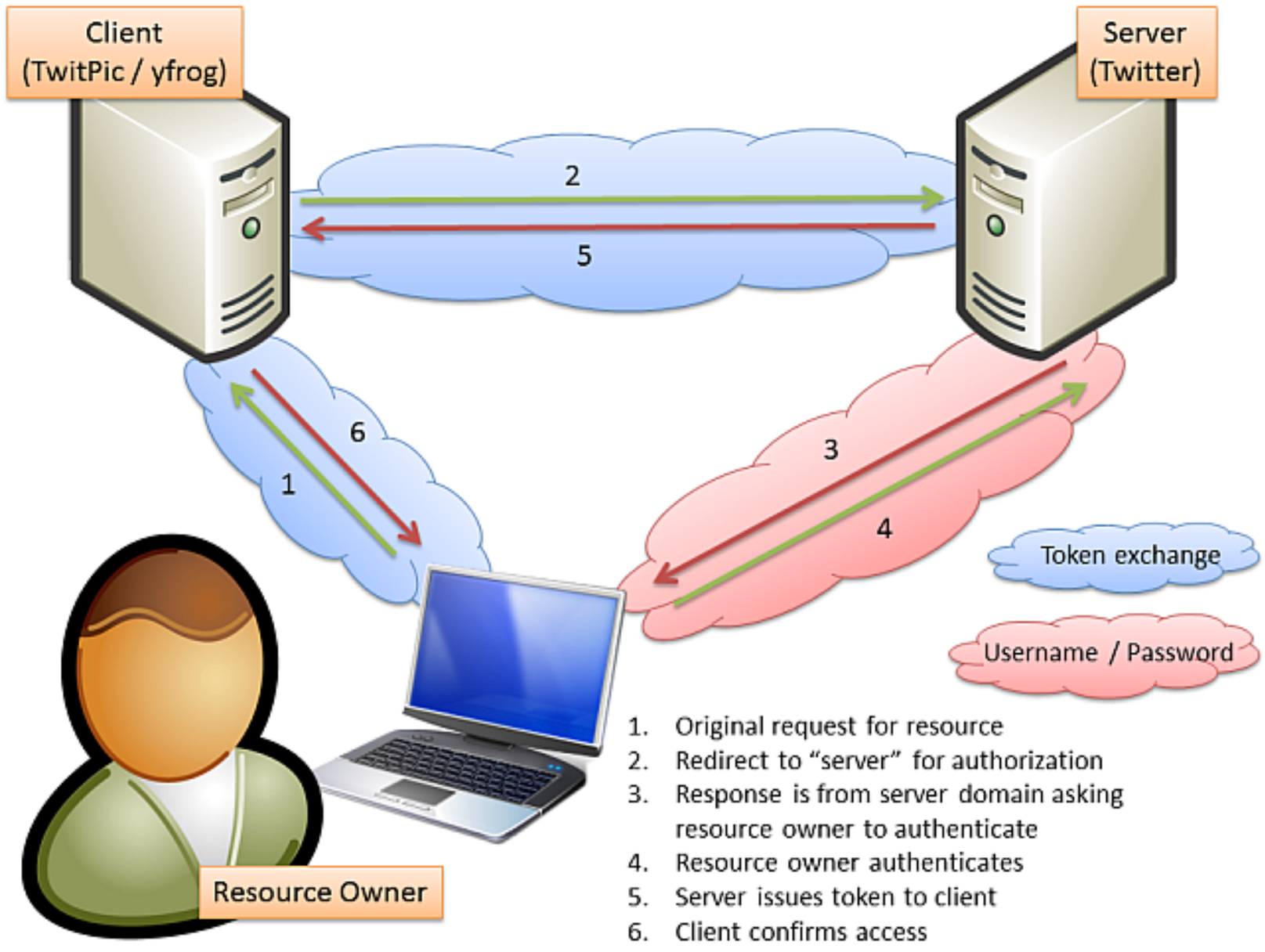
http://hueniverse.com/oauth/
https://tools.ietf.org/html/rfc5849

- "An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications."

- The idea is to allow for a safe way to share privileges without divulging private credentials
  - Give XPTO Application permission to post to your Twitter account without having to trust the developers of XPTO with your username/password and while being able to unilaterally revoke privileges.

OAuth 1

http://hueniverse.com/oauth/
https://tools.ietf.org/html/rfc5849



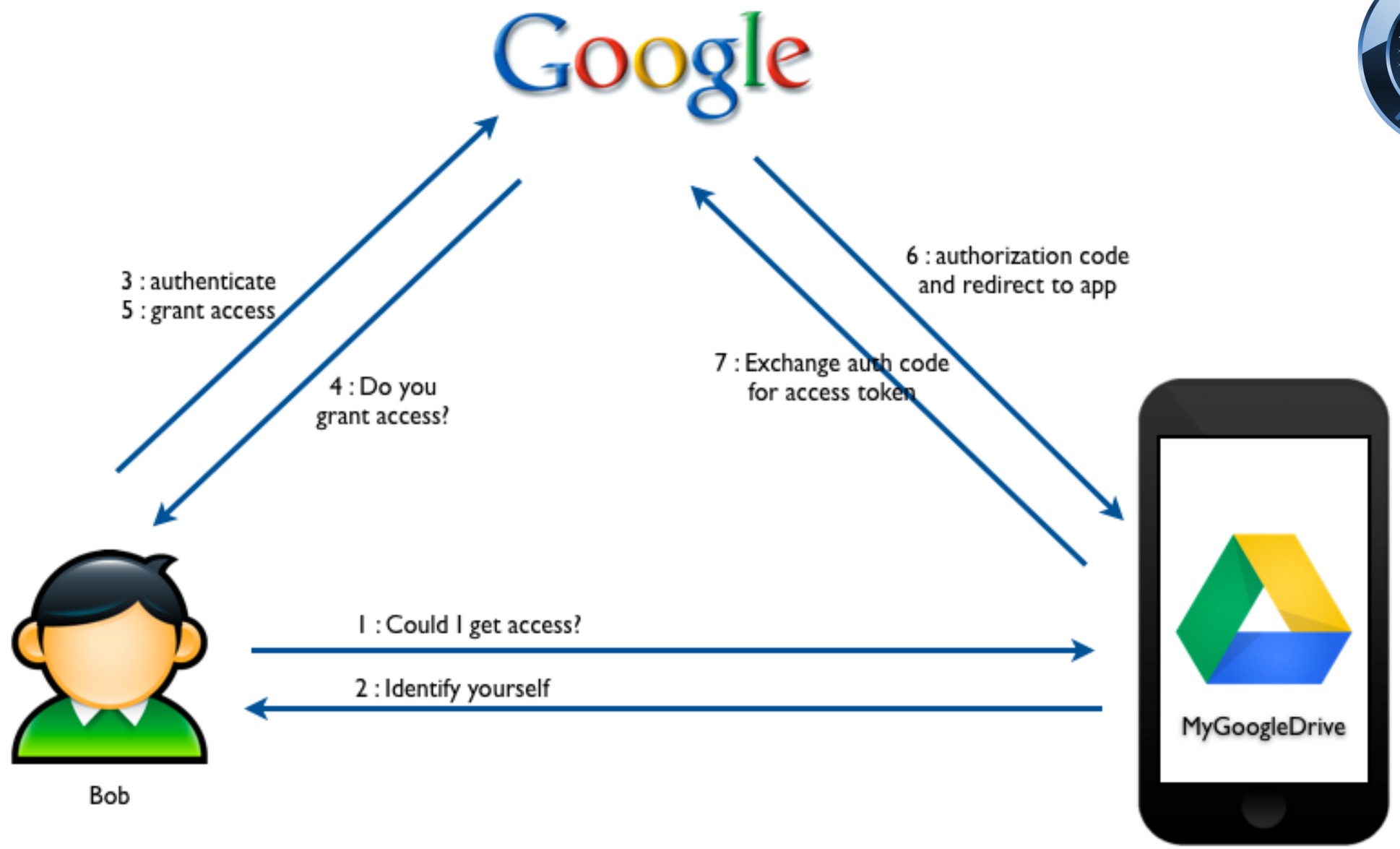
OAuth 1

http://hueniverse.com/oauth/
https://tools.ietf.org/html/rfc5849

- After the "OAuth dance" is concluded, client application has two sets of keys:
  - one that uses to identify itself as a valid application (api_key, api_secret)
  - one that uses to identify the user it wants to access (token, token_secret)
- You can revoke access at any time by letting the token provider that a given app is no long authorized (invalidating token and token_secret).

OAuth 2

https://tools.ietf.org/html/rfc6749
https://tools.ietf.org/html/rfc6750



OAuth 2

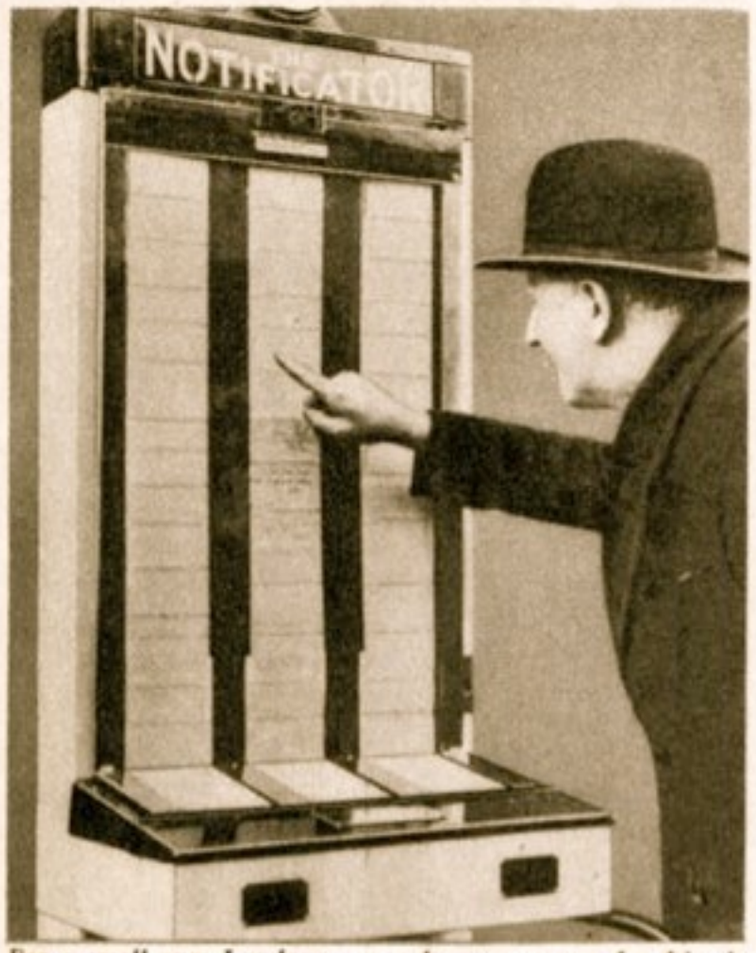
https://tools.ietf.org/html/rfc6749
https://tools.ietf.org/html/rfc6750

- Latest version of OAuth protocol
    - Similar “dance” required
    - Allows for “bearer tokens” - access is given to anyone able to provide a valid token without any further restrictions or authentication
        - access tokens are provided along with the request for the resource through a secure connection
        - tokens can expire automatically
- We will use both OAuth and OAuth2 over the next few days

Twitter

Twitter

Robot Messenger Displays Person-to-Person Notes In Public



For a small sum Londoners may leave messages for friends in public places. When written on "notificator," message moves up behind window, remaining in view for two hours.

TO AID persons who wish to make or cancel appointments or inform friends of their whereabouts, a robot message carrier has been introduced in London, England.

Known as the "notificator," the new machine is installed in streets, stores, railroad stations or other public places where individuals may leave messages for friends.

The user walks up on a small platform in front of the machine, writes a brief message on a continuous strip of paper and drops a coin in the slot. The inscription moves up behind a glass panel where it remains in public view for at least two hours so that the person for whom it is intended may have sufficient time to observe the note at the appointed place. The machine is similar in appearance to a candy-vending device.

Source: Modern Mechanix (Aug, 1935)

Anatomy of a Tweet

Anatomy of a Tweet


```
[u'contributors',
 u'truncated',
 u'text',
 u'in_reply_to_status_id',
 u'id',
 u'favorite_count',
 u'source',
 u'retweeted',
 u'coordinates',
 u'entities',
 u'in_reply_to_screen_name',
 u'in_reply_to_user_id',
 u'retweet_count',
 u'id_str',
 u'favorited',
 u'user',
 u'geo',
 u'in_reply_to_user_id_str',
 u'possibly_sensitive',
 u'lang',
 u'created_at',
 u'in_reply_to_status_id_str',
 u'place',
 u'metadata']
```

Anatomy of a Tweet

```
[u'contributors',
 u'truncated',
 u'text',
 u'in_reply_to_status_id',
 u'id',
 u'favorite_count',
 u'source',
 u'retweeted',
 u'coordinates',
 u'entities',
 u'in_reply_to_screen_name',
 u'in_reply_to_user_id',
 u'retweet_count',
 u'id_str',
 u'favorited',
 u'user',
 u'geo',
 u'in_reply_to_user_id_str',
 u'possibly_sensitive',
 u'lang',
 u'created_at',
 u'in_reply_to_status_id_str',
 u'place',
 u'metadata']
```

```
[u'follow_request_sent',
 u'profile_use_background_image',
 u'default_profile_image',
 u'id',
 u'profile_background_image_url_https',
 u'verified',
 u'profile_text_color',
 u'profile_image_url_https',
 u'profile_sidebar_fill_color',
 u'entities',
 u'followers_count',
 u'profile_sidebar_border_color',
 u'id_str',
 u'profile_background_color',
 u'listed_count',
 u'is_translation_enabled',
 u'utc_offset',
 u'statuses_count',
 u'description',
 u'friends_count',
 u'location',
 u'profile_link_color',
 u'profile_image_url',
 u'following',
 u'geo_enabled',
 u'profile_banner_url',
 u'profile_background_image_url',
 u'screen_name',
 u'lang',
```

```
 u'profile_background_tile',
 u'favourites_count',
 u'name',
 u'notifications',
 u'url',
 u'created_at',
 u'contributors_enabled',
 u'time_zone',
 u'protected',
 u'default_profile',
 u'is_translator']
```

Anatomy of a Tweet

```
[u'contributors',
 u'truncated',
 u'text',
 u'in_reply_to_status_id',
 u'id',
 u'favorite_count',
 u'source',
 u'retweeted',
 u'coordinates',
 u'entities',
 u'in_reply_to_screen_name',
 u'in_reply_to_user_id',
 u'retweet_count',
 u'id_str',
 u'favorited',
 u'user',
 u'geo',
 u'in_reply_to_user_id_str',
 u'possibly_sensitive',
 u'lang',
 u'created_at',
 u'in_reply_to_status_id_str',
 u'place',
 u'metadata']
```

```
u"I'm at Terminal Rodovi\xe1rio de Feira de Santana
 (Feira de Santana, BA) http://t.co/WirvdHwYMq"
```

```
u'<a href="http://foursquare.com" rel="nofollow">
  foursquare</a>'
```

```
[u'symbols',
 u'user_mentions',
 u'hashtags',
 u'urls']
```

```
[u'type',
 u'coordinates']
```

Anatomy of a Tweet

```
[u'contributors',
 u'truncated',
 u'text',
 u'in_reply_to_status_id',
 u'id',
 u'favorite_count',
 u'source',
 u'retweeted',
 u'coordinates',
 u'entities',
 u'in_reply_to_screen_name',
 u'in_reply_to_user_id',
 u'retweet_count',
 u'id_str',
 u'favorited',
 u'user',
 u'geo',
 u'in_reply_to_user_id_str',
 u'possibly_sensitive',
 u'lang',
 u'created_at',
 u'in_reply_to_status_id_str',
 u'place',
 u'metadata']
```

```
u"I'm at Terminal Rodovi\xe1rio de Feira de Santana
 (Feira de Santana, BA) http://t.co/WirvdHwYMq"
```

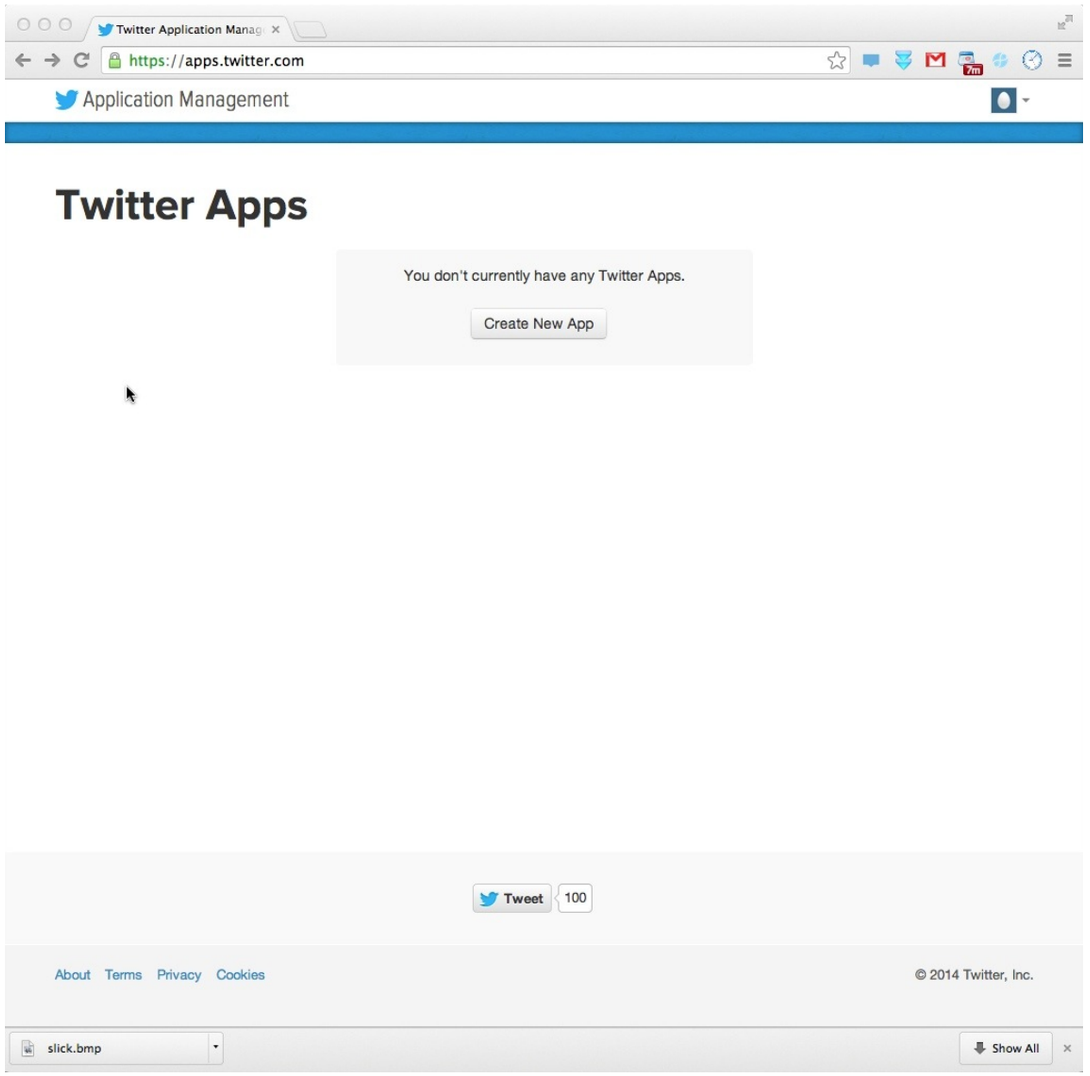
```
u'<a href="http://foursquare.com" rel="nofollow">
  foursquare</a>'
```

```
[u'symbols',
 u'user_mentions',
 u'hashtags',
 u'urls']
```

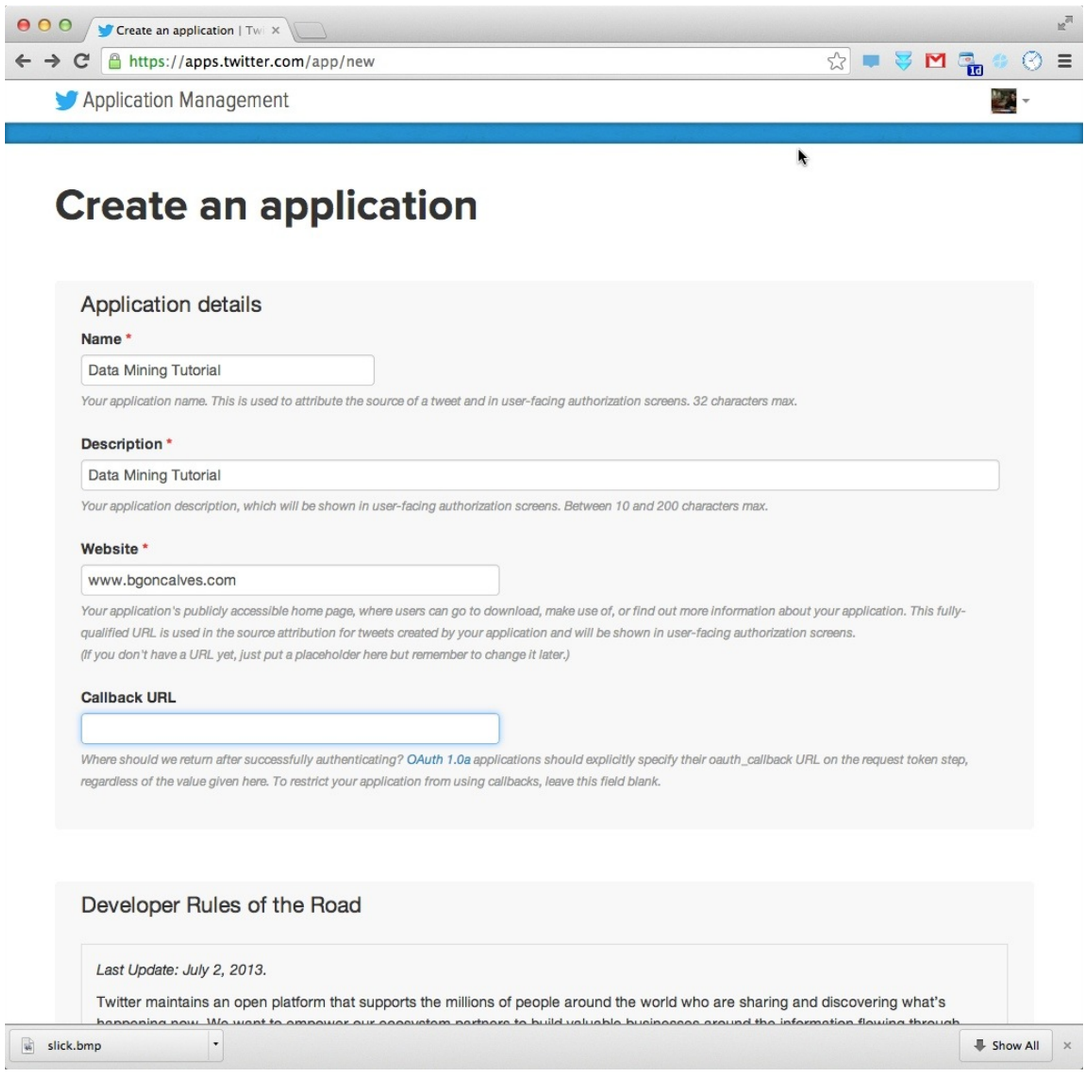
```
{u'display_url': u'4sq.com/1k5MeYF',
 u'expanded_url': u'http://4sq.com/1k5MeYF',
 u'indices': [70, 92],
 u'url': u'http://t.co/WirvdHwYMq'}
```

```
[u'type',
 u'coordinates']
```

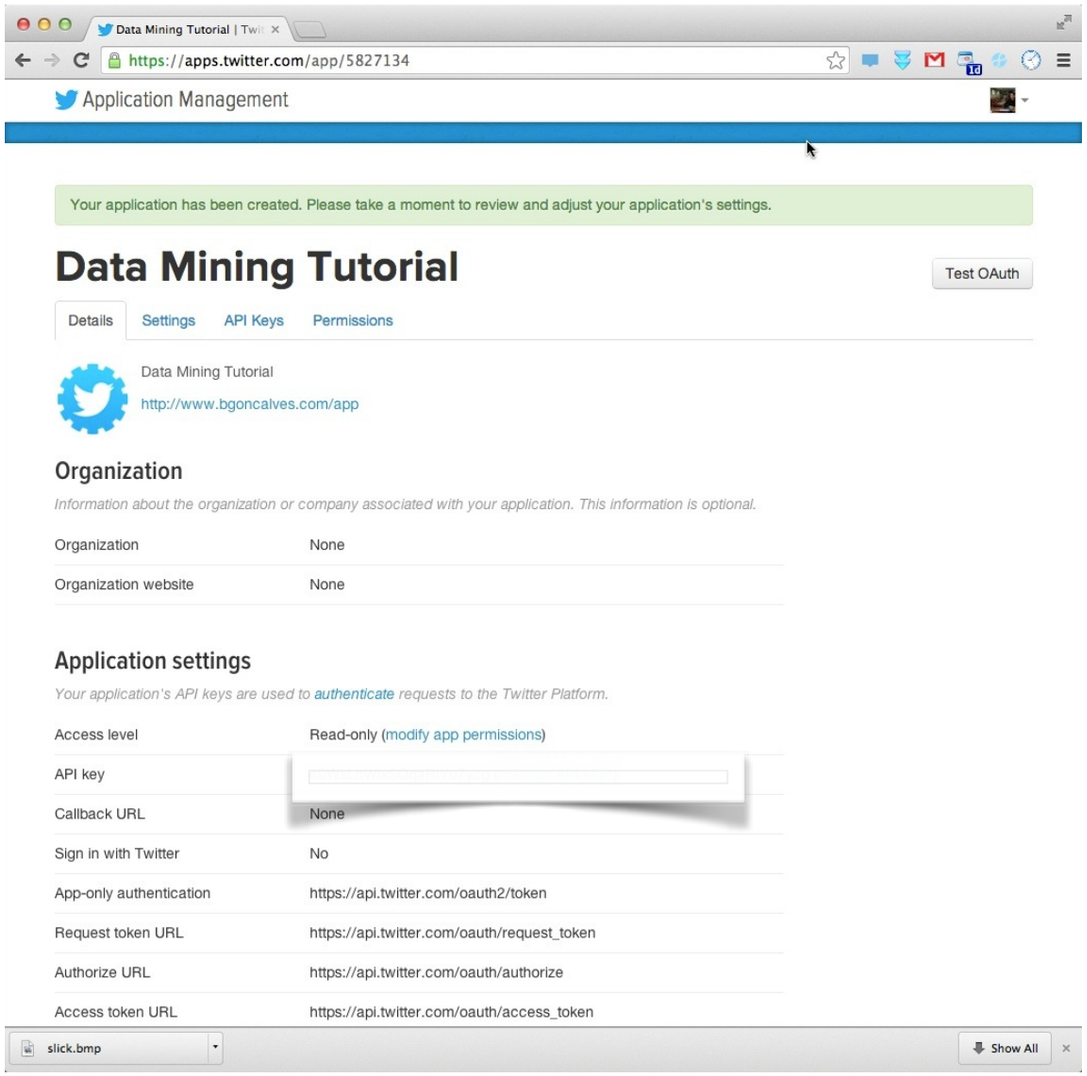
Registering an Application



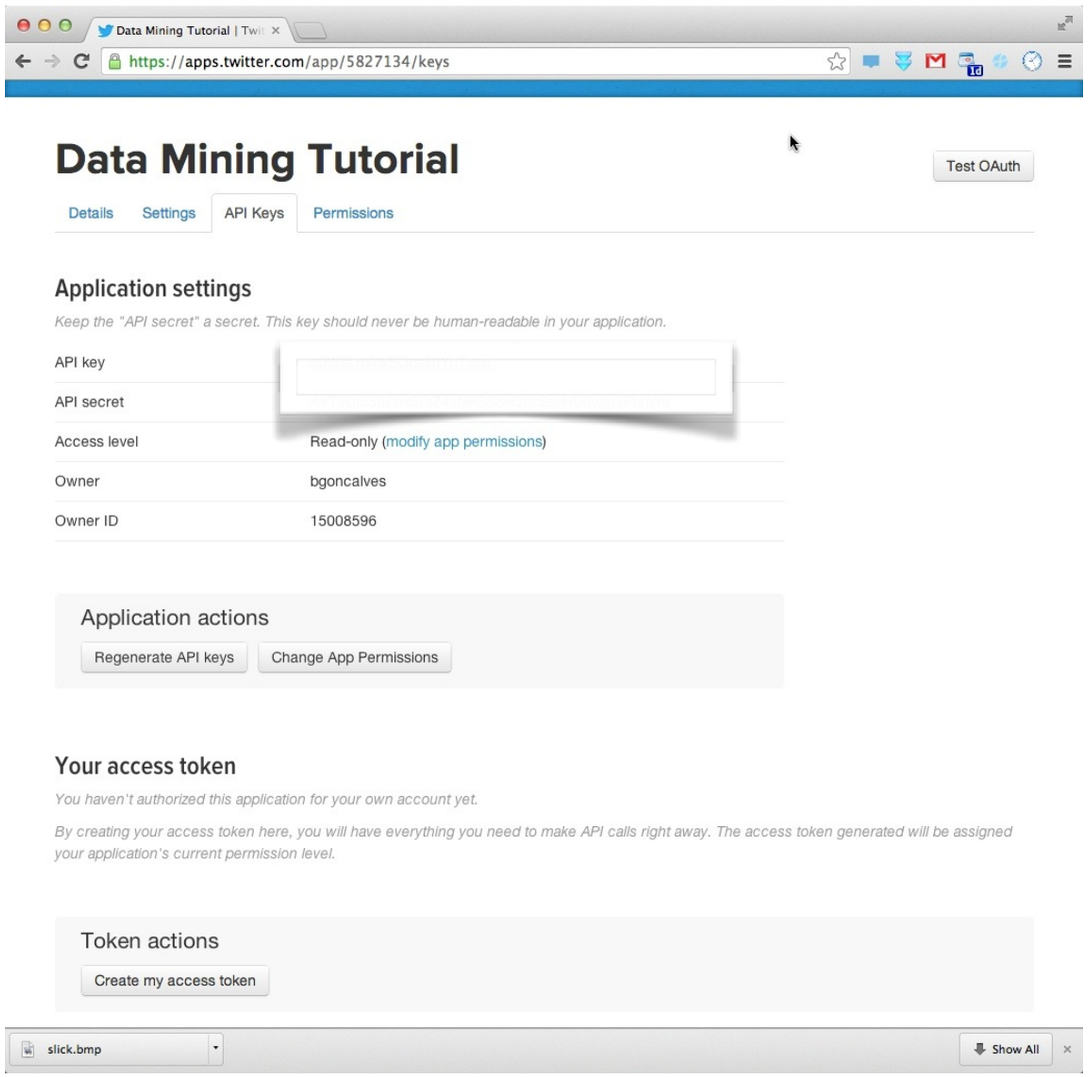
Registering an Application



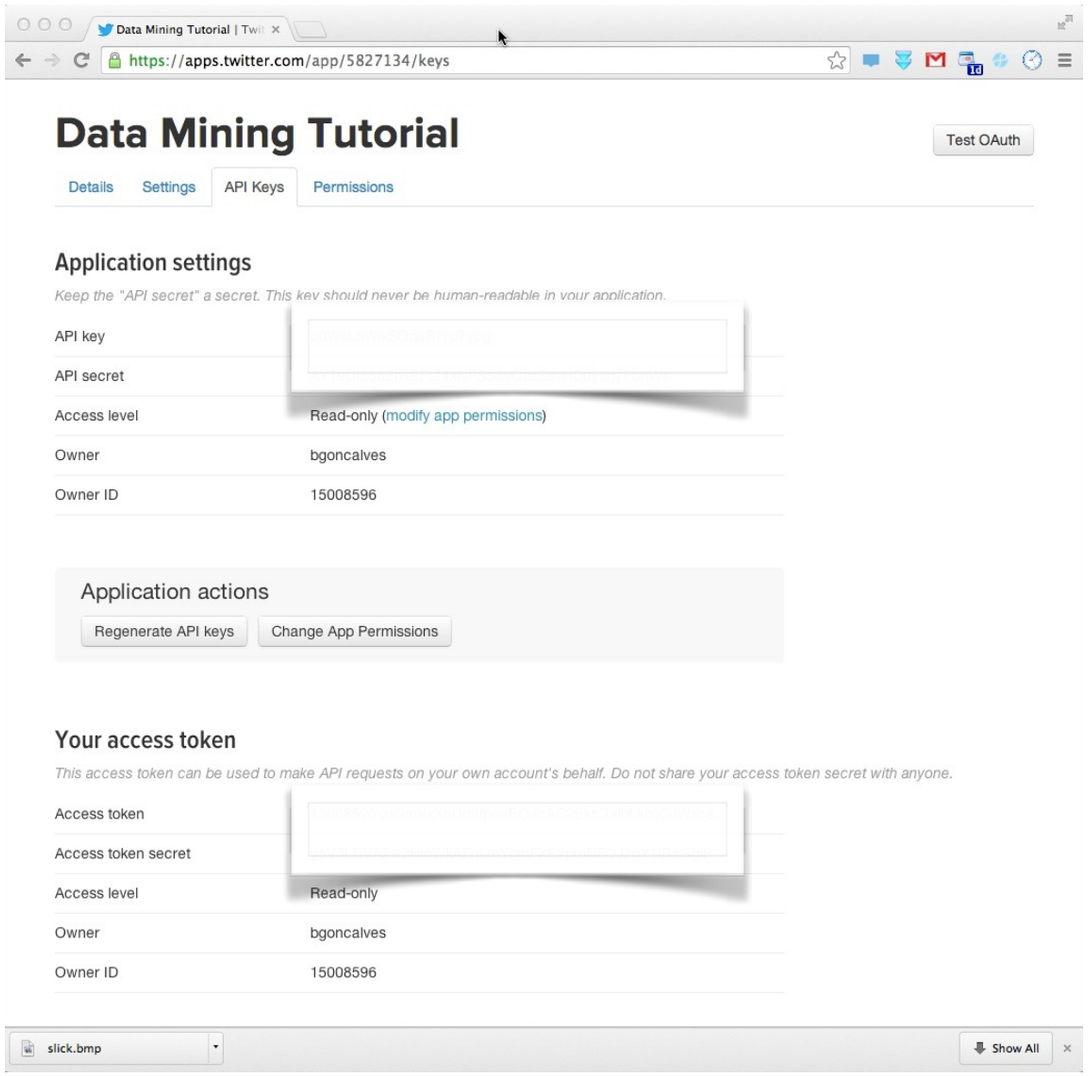
Registering an Application



Registering an Application



Registering an Application



API Basics

https://dev.twitter.com/docs

- The twitter module provides the oauth interface. We just need to provide the right credentials.

- Best to keep the credentials in a dict and parametrize our calls with the dict key. This way we can switch between different accounts easily.

- .Twitter(auth) takes an OAuth instance as argument and returns a Twitter object that we can use to interact with the API

- Twitter methods mimic API structure

- 4 basic types of objects:

    - Tweets

    - Users

    - Entities

Authenticating with the API

```
import tweepy
from twitter_accounts import accounts

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

twitter_api = twitter.Twitter(auth=auth)
```

- In the remainder of this course, the accounts dict will live inside the twitter_accounts.py file
- 4 basic types of objects:
  - Tweets
  - Users
  - Entities
  - Places

twitter_authentication.py

Searching for Tweets

https://dev.twitter.com/docs/api/1.1/get/search/tweets

- .search.tweets(query, count)
  - query is the content to search for
  - count is the maximum number of results to return
- returns dict with a list of "statuses" and "search_metadata"


```
{u'completed_in': 0.027,
u'count': 15,
u'max_id': 438088492577345536,
u'max_id_str': u'438088492577345536',
u'next_results': u'?max_id=438088485145034752&q=soccer&include_entities=1',
u'query': u'soccer',
u'refresh_url': u'?since_id=438088492577345536&q=soccer&include_entities=1',
u'since_id': 0,
u'since_id_str': u'0'}
```


- search_results["search_metadata"]["next_results"] can be used to get the next page of results

Searching for Tweets

https://dev.twitter.com/docs/api/1.1/get/search/tweets

```
query = "instagram"
count = 200

search_results = twitter_api.search.tweets(q=query, count=count)

statuses = search_results["statuses"]
tweet_count = 0

while True:
    try:
        next_results = search_results["search_metadata"]["next_results"]

        args = dict(parse.parse_qsl(next_results[1:]))

        search_results = twitter_api.search.tweets(**args)
        statuses = search_results["statuses"]

        print(search_results["search_metadata"]["max_id"])

        for tweet in statuses:
            tweet_count += 1

            if tweet_count % 10000 == 0:
                print(tweet_count,file=sys.stderr)

            print (tweet["text"])
    except:
        break
```


twitter_search.py

Streaming data

https://dev.twitter.com/docs/api/1.1/post/statuses/filter

- The Streaming api provides realtime data, subject to filters
- Use TwitterStream instead of Twitter object (.TwitterStream(auth=twitter_api.auth))
- .status.filter(track=q) will return tweets that match the query q in real time
  - Returns generator that you can iterate over

Streaming data

https://dev.twitter.com/docs/api/1.1/post/statuses/filter

```
import twitter
from twitter_accounts import accounts

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

stream_api = twitter.TwitterStream(auth=auth)

query = "bieber"

stream_results = stream_api.statuses.filter(track=query)

for tweet in stream_results:
    print(tweet["text"])
```


twitter_stream.py

User profiles

https://dev.twitter.com/docs/api/1.1/get/users/lookup

- .users.lookup() returns user profile information for a list of user_ids or screen_names

- list should be comma separated and provided as a string

```
import twitter
from twitter_accounts import accounts

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

twitter_api = twitter.Twitter(auth=auth)

screen_names = ",".join(["diunito", "giaruffo"])

search_results = twitter_api.users.lookup(screen_name=screen_names)

for user in search_results:
  print(user["screen_name"], "has", user["followers_count"], "followers")
```


twitter_users.py

Social Connections

https://dev.twitter.com/docs/api/1.1/get/friends/ids
https://dev.twitter.com/docs/api/1.1/get/followers/ids

- .friends.ids() and .followers.ids() returns a list of up to 5000 of a users friends or followers for a given screen_name or user_id

- result is a dict containing multiple fields:

```
[u'next_cursor_str',
 u'previous_cursor',
 u'ids',
 u'next_cursor',
 u'previous_cursor_str']
```

- ids are contained in results[“ids”].

- results[“next_cursor”] allows us to obtain the next page of results.

- .friends.ids(screen_name=screen_name, cursor=results[“next_cursor”]) will return the next page of results

- cursor=0 means no more results

Challenge - Social Connections

https://dev.twitter.com/docs/api/1.1/get/friends/ids
https://dev.twitter.com/docs/api/1.1/get/followers/ids

- get a list of followers of "stephen_wolfram"

```
import twitter
from twitter_accounts import accounts

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

twitter_api = twitter.Twitter(auth=auth)

screen_name = "stephen_wolfram"

cursor = -1
followers = []

while cursor != 0:
    result = twitter_api.followers.ids(screen_name=screen_name,
cursor=cursor)

    followers += result["ids"]
    cursor = result["next_cursor"]

print("Found", len(followers), "Followers")
```


twitter_followers.py

User Timeline

https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline

- .statuses.user_timeline() returns a set of tweets posted by a single user
- Important options:
  - include_rts='true' to Include retweets by this user
  - count=200 number of tweets to return in each call
  - trim_user='true' to not include the user information (save bandwidth and processing time)
  - max_id=1234 to include only tweets with an id lower than 1234
- Returns at most 200 tweets in each call. Can get all of a users tweets (up to 3200) with multiple calls using max_id

User Timeline

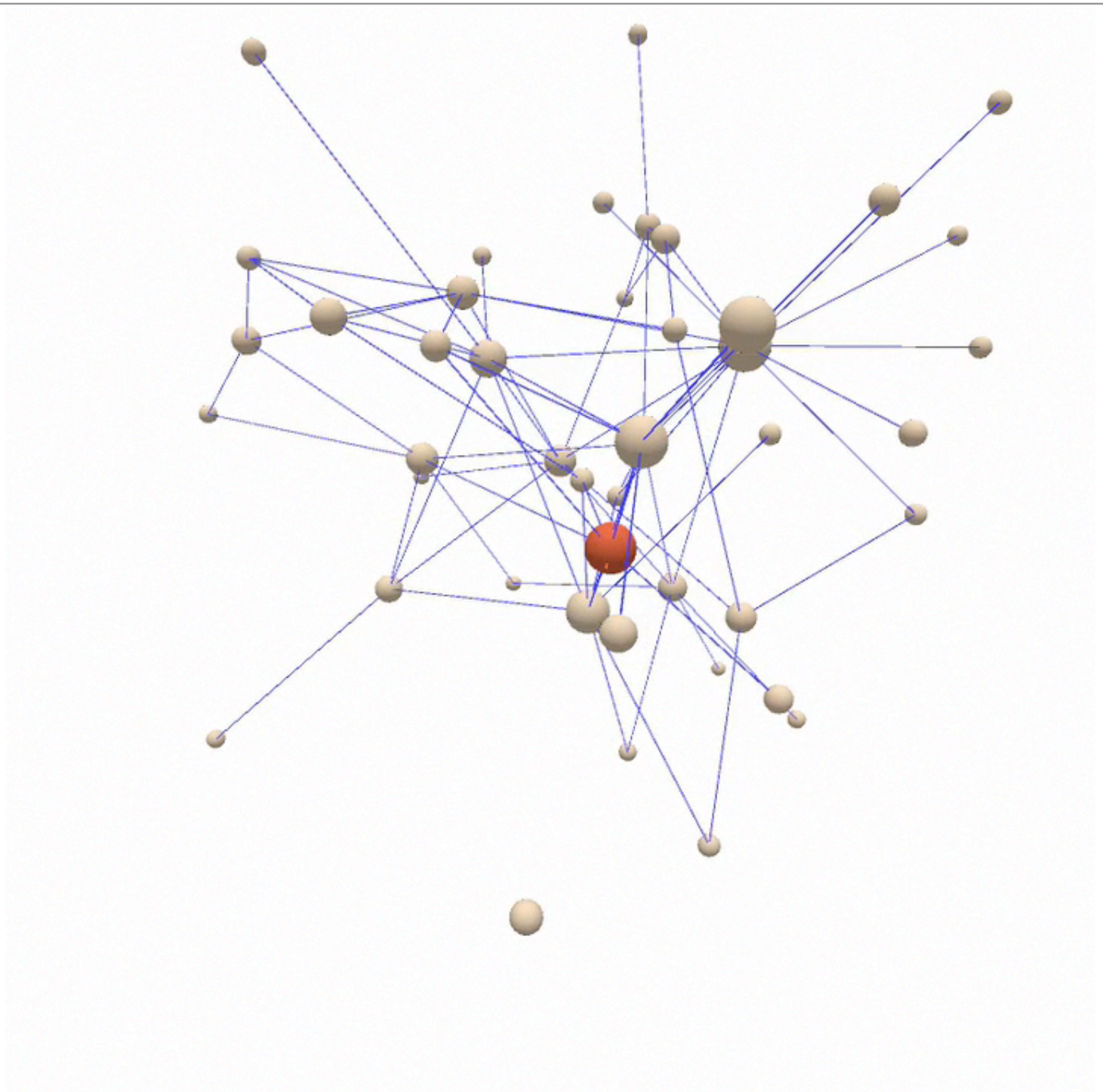
https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline

```
import twitter
from twitter_accounts import accounts

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

twitter_api = twitter.Twitter(auth=auth)
screen_name = "BarackObama"

args = { "count" : 200,
         "trim_user": "true",
         "include_rts": "true"
        }

tweets = twitter_api.statuses.user_timeline(screen_name = screen_name, **args)
tweets_new = tweets

while len(tweets_new) > 0:
    max_id = tweets[-1]["id"] - 1
    tweets_new = twitter_api.statuses.user_timeline(screen_name = screen_name, max_id=max_id, **args)
    tweets += tweets_new

print("Found", len(tweets), "tweets")
```


twitter_timeline.py

Social Interactions

http://www.youtube.com/watch?v=mIKGgsGCIMc



Challenge - Social Interactions

- Extract user-user interactions (retweets and mentions) from a users timeline

Social Interactions

```
import twitter
from twitter_accounts import accounts

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

twitter_api = twitter.Twitter(auth=auth)
screen_name = "BarackObama"
args = { "count" : 200, "trim_user": "true", "include_rts": "true"}

tweets = twitter_api.statuses.user_timeline(screen_name=screen_name, **args)
tweets_new = tweets

while len(tweets_new) > 0:
    max_id = tweets[-1]["id"] - 1
    tweets_new = twitter_api.statuses.user_timeline(screen_name=screen_name, max_id=max_id, **args)
    tweets += tweets_new

user = tweets[0]["user"]["id"]

for tweet in tweets:
    if "retweeted_status" in tweet:
        print(tweet["retweeted_status"]["user"]["id"], "->", user)
    elif tweet["in_reply_to_user_id"]:
        print(user, "->", tweet["in_reply_to_user_id"])
```


twitter_interactions.py

Challenge - Social Interactions

- Extract user-user interactions (retweets and mentions) from a users timeline
    - and build a NetworkX graph object

```
import twitter
from twitter_accounts import accounts
import networkx as NX

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

twitter_api = twitter.Twitter(auth=auth)

screen_name = "BarackObama"

args = { "count" : 200,
         "trim_user": "true",
         "include_rts": "true"}

tweets = twitter_api.statuses.user_timeline(screen_name=screen_name, **args)

tweets_new = tweets

while len(tweets_new) > 0:
    max_id = tweets[-1]["id"] - 1
    tweets_new = twitter_api.statuses.user_timeline(screen_name=screen_name, max_id=max_id, **args)
    tweets += tweets_new

user = tweets[0]["user"]["id"]
G = NX.DiGraph()

for tweet in tweets:
    if "retweeted_status" in tweet:
        G.add_edge(tweet["retweeted_status"]["user"]["id"], user)
    elif tweet["in_reply_to_user_id"]:
        G.add_edge(user, tweet["in_reply_to_user_id"])

print("Graph has", G.number_of_nodes(), "nodes,",\
                   G.number_of_edges(), "edges, and the maximum degree is",\
                   max(G.degree().values()))
```

twitter_networkx.py

Streaming Geocoded data

https://dev.twitter.com/streaming/overview/request-parameters#locations

- The Streaming api provides realtime data, subject to filters

- Use TwitterStream instead of Twitter object (.TwitterStream(auth=twitter_api.auth))

- .status.filter(track=q) will return tweets that match the query q in real time

- Returns generator that you can iterate over

- .status.filter(locations=bb) will return tweets that occur within the bounding box bb in real time

- bb is a comma separated pair of lon/lat coordinates.
  - -180,-90,180,90 - World
  - -74,40,-73,41 - NYC

Streaming Geocoded data

https://dev.twitter.com/streaming/overview/request-parameters#locations

```
import twitter
from twitter_accounts import accounts
import sys
import gzip

app = accounts["social"]

auth = twitter.oauth.OAuth(app["token"],
                           app["token_secret"],
                           app["api_key"],
                           app["api_secret"])

stream_api = twitter.TwitterStream(auth=auth)

query = "-74,40,-73,41"  # NYC
stream_results = stream_api.statuses.filter(locations=query)
tweet_count = 0

fp = gzip.open("NYC.json.gz", "a")

for tweet in stream_results:
    try:
        tweet_count += 1
        print (tweet_count, tweet["id"])
        print(tweet, file=fp)
    except:
        pass

    if tweet_count % 10000 == 0:
        print(tweet_count, file=sys.stderr)
        break
```

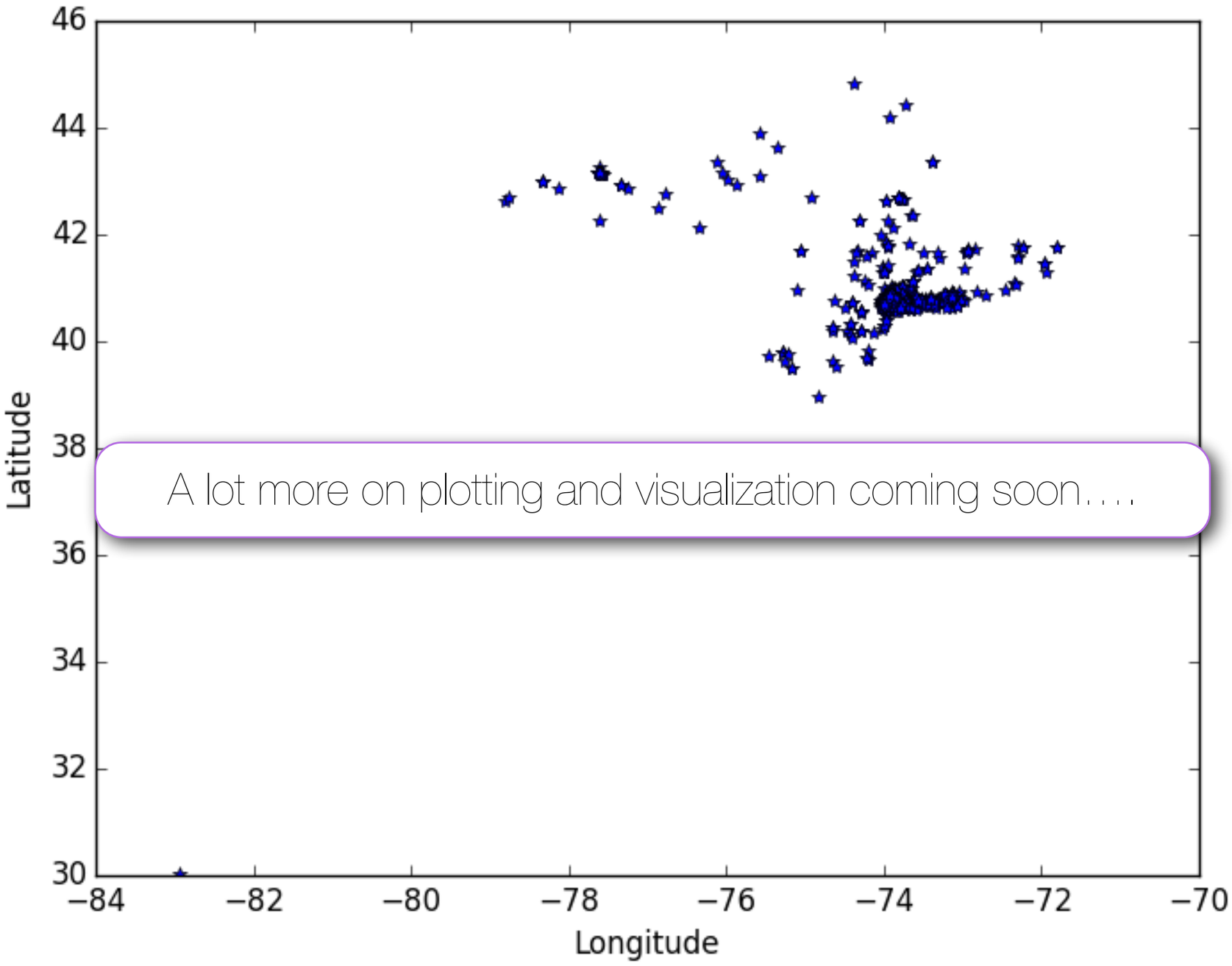
twitter_location.py

Plotting geolocated tweets

```
import sys
import gzip
import matplotlib.pyplot as plt

x = []
y = []

line_count = 0

try:
    for line in gzip.open(sys.argv[1]):
        try:
            tweet = eval(line.strip())
            line_count += 1

            if "coordinates" in tweet and tweet["coordinates"] is not None:
                x.append(tweet["coordinates"]["coordinates"][0])
                y.append(tweet["coordinates"]["coordinates"][1])
        except:
            pass
except:
    pass

print("Read", line_count, "and found", len(x), "geolocated tweets", file=sys.stderr)

plt.plot(x, y, '*')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.savefig(sys.argv[1] + '.png')
plt.close()
```

plot_tweets.py

Plotting geolocated tweets



A lot more on plotting and visualization coming soon....

Foursquare

Foursquare


12:29
Four Square for Dummies
Sponso
FAQ
Videos
Twitter
Suppor
foursquare
I'm looking for...
About Foursquare
What is Foursquare?
Foursquare is a free app that helps you and your friends make the most of where you are. When you're out and about, use Foursquare to share and save the places you visit. And, when you're looking for inspiration for what to do next, we'll give you personalized recommendations and deals based on where you, your friends, and people with your tastes have been.
Whether you're setting off on a trip around the world, coordinating a night out with friends, or trying to pick out the best dish at your local restaurant, Foursquare is the perfect companion.
Fetched

Anatomy of a Checkin

Anatomy of a Checkin

```
[u'venue',
 u'like',
 u'photos',
 u'source',
 u'visibility',
 u'entities',
 u'shout',
 u'timeZoneOffset',
 u'type',
 u'id',
 u'createdAt',
 u'likes']
```

Anatomy of a Checkin

```
[u'venue',
 u'like',
 u'photos',
 u'source',
 u'visibility',
 u'entities',
 u'shout',
 u'timeZoneOffset',
 u'type',
 u'id',
 u'createdAt',
 u'likes']
```

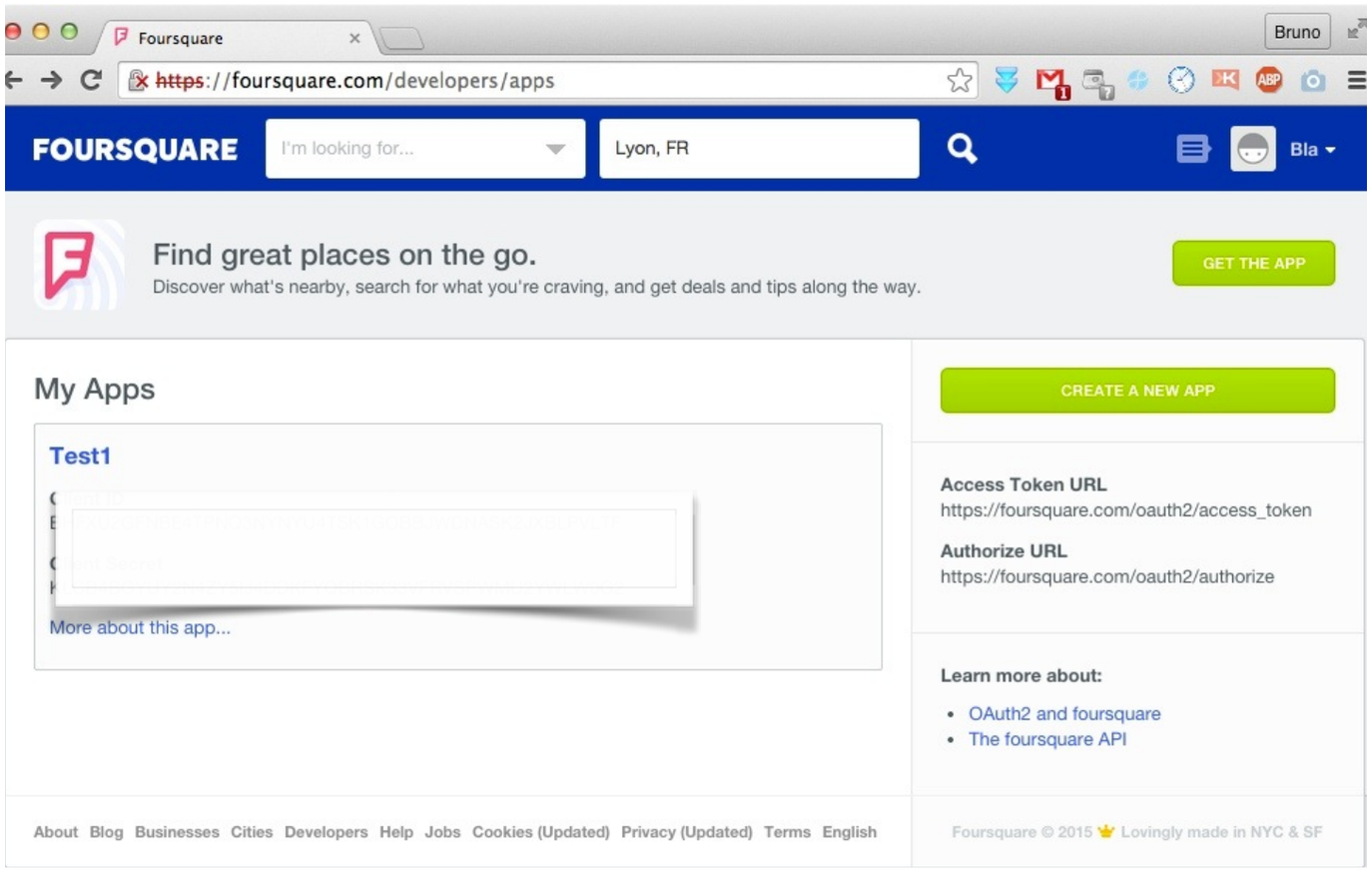
```
[u'verified',
 u'name',
 u'url',
 u'like',
 u'contact',
 u'location',
 u'stats',
 u'id',
 u'categories',
 u'likes']
```

```
[{u'indices': [112, 137],
  u'object': {u'url': u'http://go.nasa.gov/9kpN5g'},
  u'type': u'url'}]
```

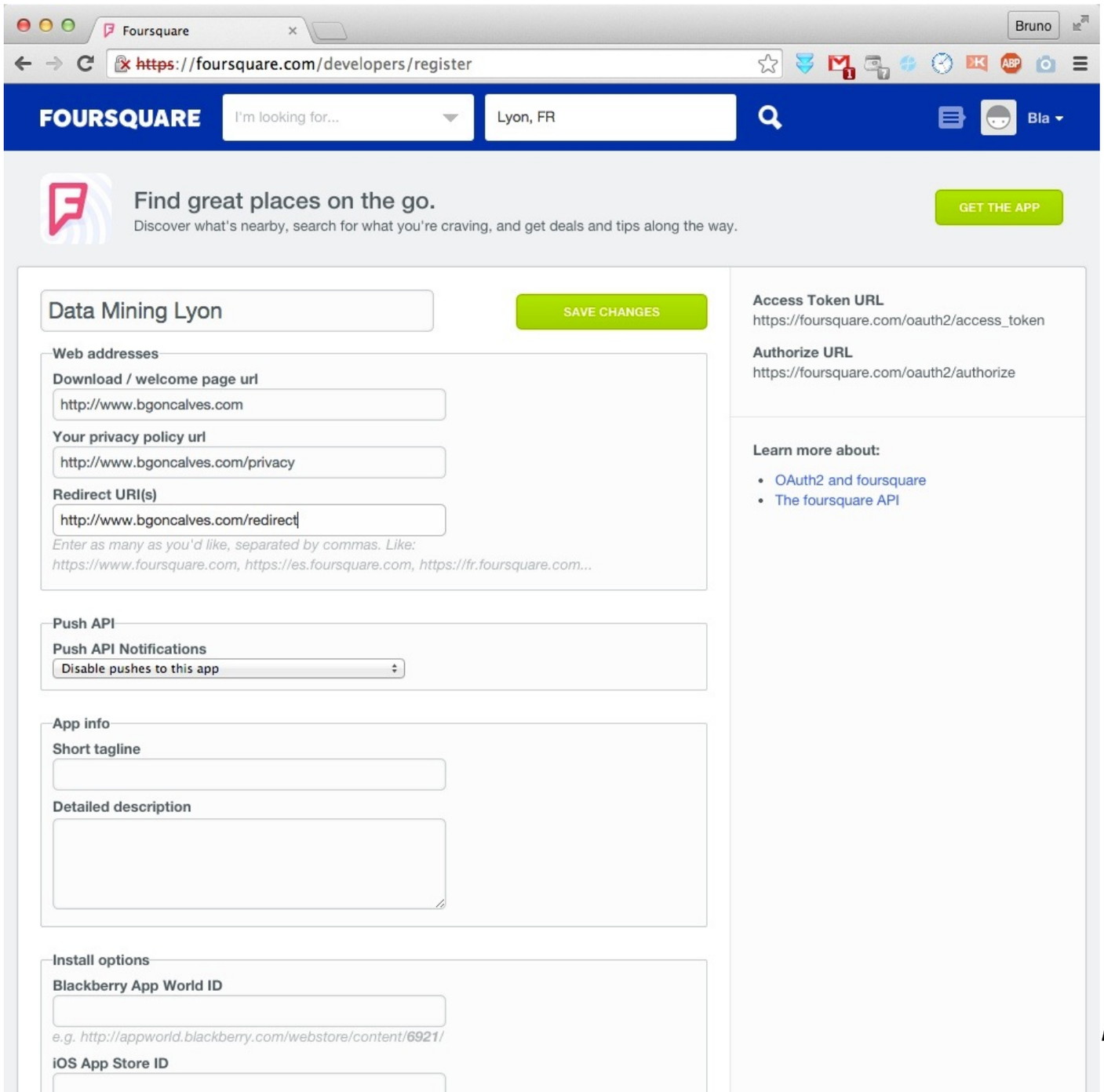
```
u"We've got a new 'Curiosity Explorer' badge. Explore your
curiosity at science museums & planetariums to earn it http://
go.nasa.gov/9kpN5g"
```

```
{u'count': 1837,
 u'groups': [{u'count': 1837, u'items': [], u'type': u'others'}],
 u'summary': u'1837 likes'}
```

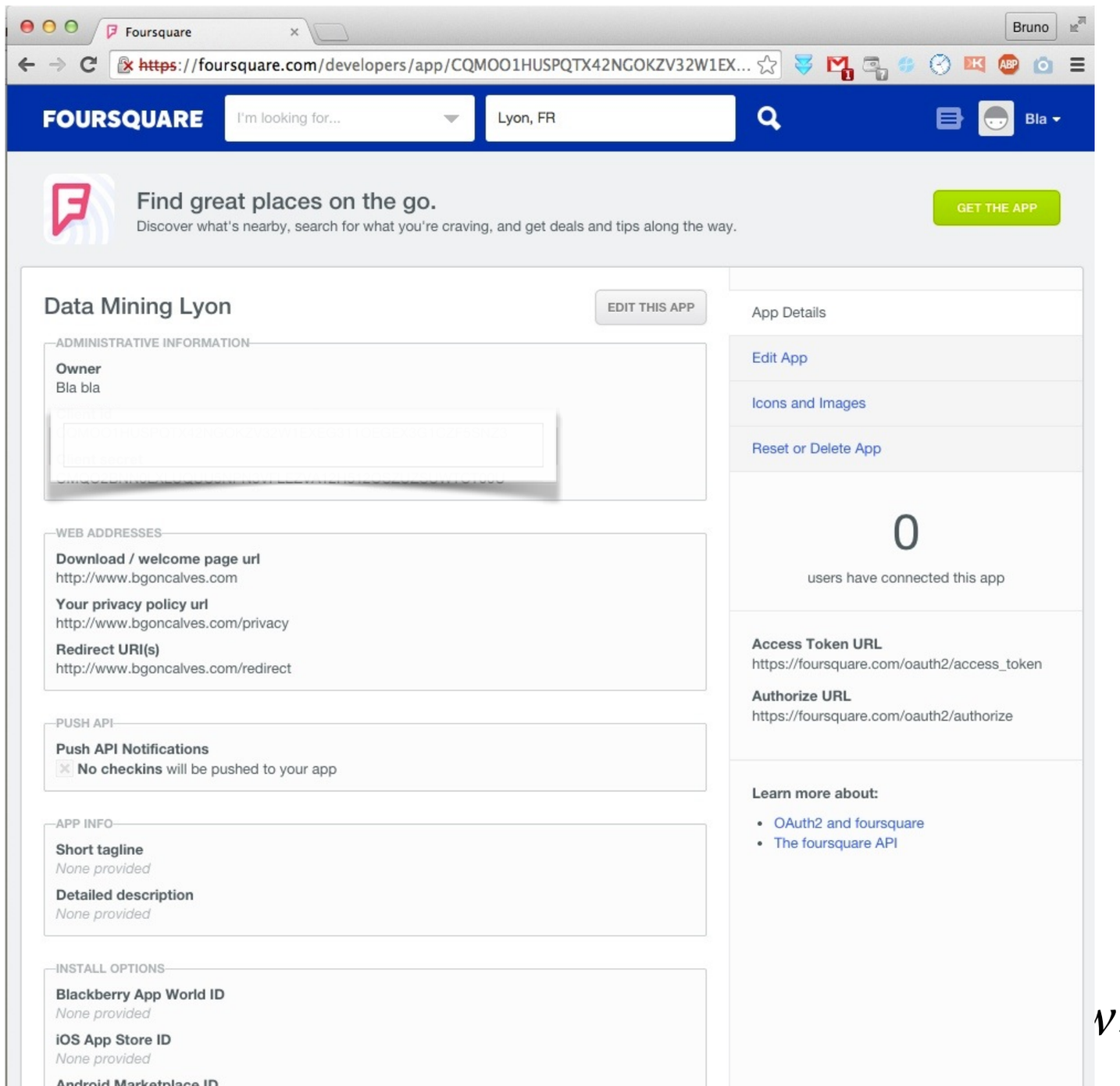
Registering An Application



Registering An Application



Registering An Application



Registering An Application

- We now have our client_id and client_secret that we can use to request an access token.
- First we request an auth_url, a URL where the "user" will be asked to login to foursquare and authorize our app

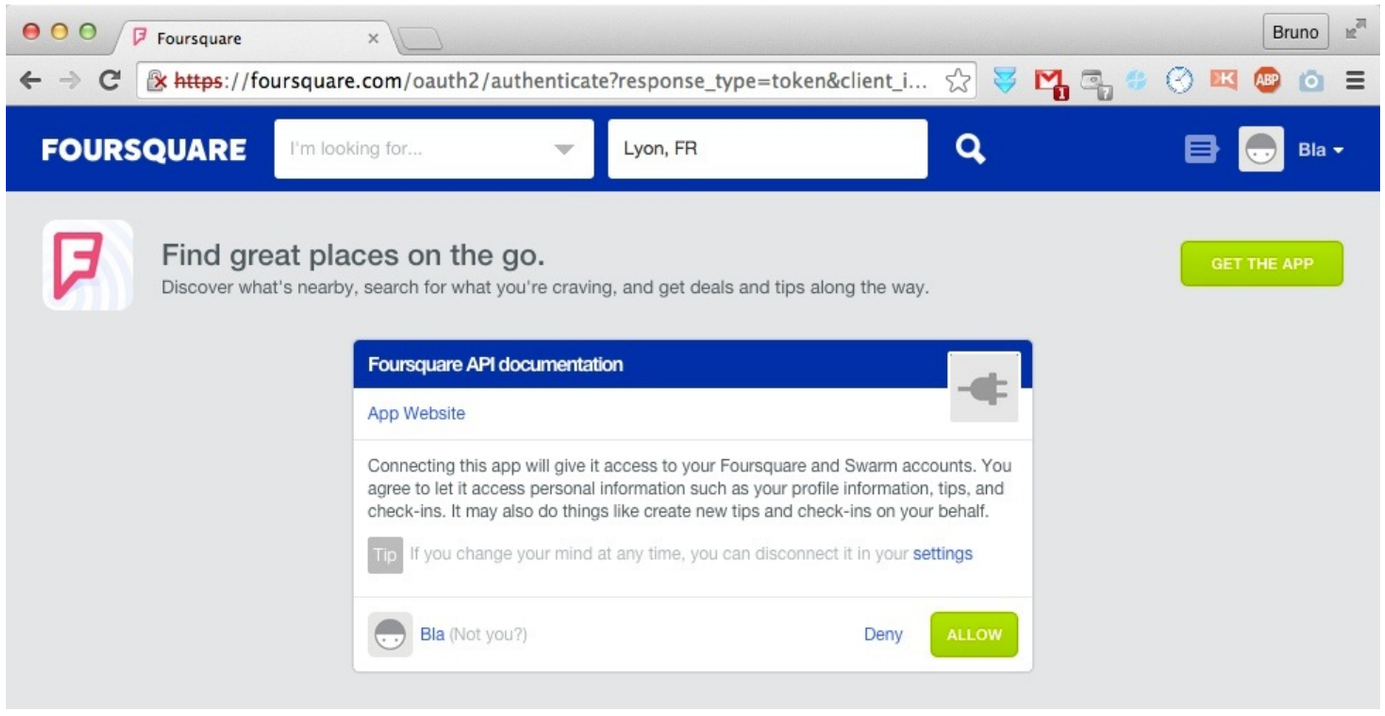
```
import foursquare

accounts = {"tutorial": {"client_id": "CLIENT_ID",
                         "client_secret": "CLIENT_SECRET",
                         "access_token": ""
                         }
            }

app = accounts["tutorial"]

client = foursquare.Foursquare(client_id=app["client_id"],
                               client_secret=app["client_secret"],
                               redirect_uri='http://www.bgoncalves.com/redirect')

auth_uri = client.oauth.auth_url()
print(auth_uri)
```

- In the remainder of this lecture, the accounts dict will live inside the foursquare_accounts.py file

foursquare_auth_uri.py

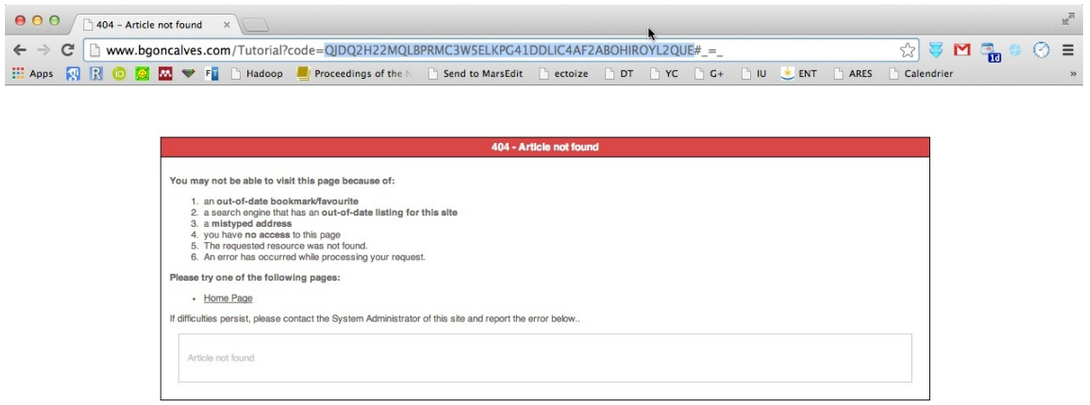
Registering An Application

- Now we must access our auth_uri in a browser, login and authorize the application



Registering An Application

- Afterwards we will be redirected automatically to out "redirect_uri" we used when registering the application



- Don't worry about the error. What we need is just the code portion of the url (the part between the "=" and the "#".

- This is the final piece of the puzzle.

Registering An Application

- with this code we can now request an auth_token which will allow us to authenticate with the Foursquare API

```
access_token = client.oauth.get_token('CODE')
```

- This will return the OAuth2 access_token that we can then use directly.

```
import foursquare
from foursquare_accounts import accounts

app = accounts["tutorial"]

client = foursquare.Foursquare(client_id=app["client_id"],

client_secret=app["client_secret"])

client.set_access_token(app["access_token"])
```

- Much simpler and intuitive
- Less prone to mistakes
- Automatically takes care of all the dirty details

foursquare_authentication.py

Objects

- Two main types of objects:
    - Users
    - Venues
- Multiple possible actions (by Users on Venues)
    - Checkin
    - Like
    - Tip

API Limitations

- Users have privacy concerns with respect to publicly sharing their location.
    - "Stalker apps"
    - "Please Rob Me"
- Privacy is a big concern for Foursquare
    - API structure reflects this
    - "Easy" to get information on users and on venues. Connecting users to venues much harder to obtain.

Venues

https://developer.foursquare.com/overview/venues.html

- Venues correspond to physical locations
- Are perhaps the most important object in the Foursquare universe
- API is particularly generous, allowing for 5000 requests per hour.
- .venues(venue_id)
  Returns a venue object
- .venues.similar(venue_id)
  Returns a list of similar venues (abbreviated)
- .venues.search({“query”:query, “near”:location})
  Searches for places matching the query (“pizza”, “Eiffel Tower”, etc) near location (“Paris”, etc).

```
[u'city',
 u'cc',
 u'country',
 u'state',
 u'address',
 u'lat',
 u'lng']
```

```
[u'count',
 u'groups',
 u'summary']
```

```
[u'rating',
 u'reasons',
 u'likes',
 u'mayor',
 u'createdAt',
 u'verified',
 u'id',
 u'shortUrl',
 u'pageUpdates',
 u'location',
 u'tips',
 u'listed',
 u'canonicalUrl',
 u'tags',
 u'photos',
 u'attributes',
 u'stats',
 u'dislike',
 u'hereNow',
 u'categories',
 u'name',
 u'like',
 u'phrases',
 u'specials',
 u'contact',
 u'popular',
 u'timeZone']
```

Challenge - Similar Venues

https://developer.foursquare.com/overview/venues.html

- Obtain the list of venues similar to:

43695300f964a5208c291fe3

```
import foursquare
from foursquare_accounts import accounts

app = accounts["tutorial"]

client = foursquare.Foursquare(client_id=app["client_id"],
                               client_secret=app["client_secret"])

client.set_access_token(app["access_token"])

venue_id = "43695300f964a5208c291fe3"
venue = client.venues(venue_id)
similar = client.venues.similar(venue_id)

print("Similar venues to", venue["venue"]["name"], "(", venue["venue"]["hereNow"]["summary"], ")")

for venue in similar["similarVenues"]["items"]:
    print(venue["name"])
```


foursquare_venues.py

Users

https://developer.foursquare.com/docs/users/users

- Users interact with venues in multiple ways:
    - checkin
    - leaving a “tip”
    - “liking”
- Users connect to each other through friendship/colocation
- .users(user_id) Returns the user object
- .users.friends(user_id) Returns the list of friends for a user
- .users.checkins(user_id) Returns list of (public) checkins
- .users.search({“twitter”:screen_name}) Search for the user with the given twitter screen_name. Returns abbreviated user object

Tips

https://developer.foursquare.com/docs/venues/tips

- Users can leave tips in venues at any time (without checking in)
- (Reduced) Tips for a venue can be accessed using .venues.tips(venue_id)
- Limited to a maximum of 500 per call, defined with the "count" parameter. Get further tips with "offset" parameter (same as for friends).
- Full Tip objets can be obtained with .tips(tip_id)
- Contain (Reduced) User object and are public, providing an easy way to connect users with venues.

Challenge - Tips

- Obtain the complete list of user_id who left a tip at

```
import foursquare
from foursquare_accounts import accounts

app = accounts["tutorial"]

client = foursquare.Foursquare(client_id = app["client_id"],
                               client_secret = app["client_secret"])

client.set_access_token(app["access_token"])

venue_id = "43695300f964a5208c291fe3"

tips = client.venues.tips(venue_id)
tips_list = tips["tips"]["items"]
tip_count = tips["tips"]["count"]

while len(tips_list) < tip_count:
    tips = client.venues.tips(venue_id, {"offset":
len(tips_list)})
    tips_list += tips["tips"]["items"]

print len(tips_list), tip_count

for tip in tips_list:
    print tip[“user”]["id"], tip["text"]
```

@bgoncalves

foursquare_tips.py

Checkins

https://developer.foursquare.com/docs/checkins/checkins.html

- Checkins are the *Raison d'être* of Foursquare.
- They connect Users with Venues providing valuable temporal and demographic information.
- .checkins(checkin_id) Returns the Checkin object
- .users.checkins(user_id) Returns the list of ***Public*** checkins for User user_id or all checkins if user_id is friends of the user using the application.

```
import foursquare
from foursquare_accounts import accounts

app = accounts["tutorial"]

client = foursquare.Foursquare(client_id=app["client_id"],

client_secret=app["client_secret"])

client.set_access_token(app["access_token"])

checkin_id = "5089b44319a9974111a6c882"

checkin = client.checkins(checkin_id)
user_name = checkin["checkin"]["user"]["firstName"]

print(checkin_id, "was made by", user_name)
```

foursquare_checkins.py

Checkins

https://developer.foursquare.com/docs/checkins/checkins.html

- Users have the option of sharing their checkins through Twitter and Facebook, making them publicly accessible
- The status text is shared along with the URL of the web version of the checkin.
- To allow Twitter and Facebook friends to access the checkin, a special "access_token", called a signature, is added to the checkin URL.
- Each signature is valid for just a single checkin and it allows anyone to access the respective checkin

Checkins

https://developer.foursquare.com/docs/checkins/checkins.html

- Signed checkin urls are of the form:

  https://foursquare.com/<user>/checkin/<checkin_id>?s=<signature>&ref=tw

- For example:

  https://foursquare.com/tyayayayaa/checkin/5304b652498e734439d8711f?
  s=ScMqmpSLg1buhGXQicDJS4A_FVY&ref=tw

- corresponds to user tyayayayaa performing checkin 5304b652498e734439d8711f and has signature ScMqmpSLg1buhGXQicDJS4A_FVY

- .checkins(checkin_id, {“signature”: signature}) can be used to query the API using the signature key to access a private checkin

Check-in Details

`https://api.foursquare.com/v2/checkins/CHECKIN_ID`

Get details of a check-in.

HTTP Method	GET
Requires Acting User	Yes (learn more)
Modes supported	`swarm` (learn more)

Parameters

All parameters are optional, unless otherwise indicated.

CHECKIN_ID	IHR8THISVNU	The ID of the check-in to retrieve additional information for.
signature	ASDJKASLJDLA	**This is now deprecated**—see the checkins/resolve endpoint for how to retrieve check-in details from public feeds. However, check-ins still shared from legacy Foursquare clients to public feeds such as Twitter will have a signature (`s=XXXXXX`) that allows users to bypass the friends-only access restriction on checkins. The same value can be used here for programmatic access to otherwise inaccessible checkins. Callers should use the bit.ly API to first expand any 4sq.com links.

Response fields

checkin	A complete check-in object.

Resolving checkins… the hard way!

```
from  foursquare_accounts import accounts
from urllib import parse
import posixpath
import requests

app = accounts["tutorial"]

url_base = "https://api.foursquare.com/v2/checkins/resolve?shortId=%s&oauth_token=%s&v=20160912"

swarm_url = "https://www.swarmapp.com/c/j0cBYuhNHki"
parsed_url = parse.urlparse(swarm_url)
short_id = posixpath.basename(parsed_url.path)

url = url_base % (short_id, app["access_token"])

req = requests.get(url)

checkin = req.json()["response"]["checkin"]
checkin_id = checkin["id"]
user_name = checkin["user"]["firstName"]

print(short_id, ":", checkin_id, "was made by", user_name)
```


foursquare_checkins_hard.py

Wikipedia

Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects

Lviv

From Wikipedia, the free encyclopedia

Coordinates: 49°49′48″N 24°00′51″E

"Lwów", "Lvov", and "Lemberg" redirect here. For other uses, see Lwów (ship), Lvov (disambiguation), and Lemberg (disambiguation).

Lviv (Ukrainian: Львів, pronounced [lʲʊ̯ʲiu̯] (listen), Polish: *Lwów*,[1] pronounced [lvuf] (listen), Russian: Львов, *Lvov*, Hellenic and Latin: *Leopolis*, see also other names) is the largest city in western Ukraine and the seventh-largest city in the country overall, with a population of around 728,350 as of 2016. Lviv is one of the main cultural centres of Ukraine.

Named in honor of the Leo, the eldest son of Rus' King Daniel of Galicia, it was the capital of the Kingdom of Galicia–Volhynia (also called Kingdom of Rus')[2] from 1272 to 1349, when it was conquered by King Casimir III the Great who then became known as the King of Poland and Rus'. From 1434, it was the regional capital of the Ruthenian Voivodeship in the Kingdom of Poland and was known as *Lwów*. In 1772, after the First partition of Poland, the city became the capital of the Habsburg Kingdom of Galicia and Lodomeria and was renamed to *Lemberg*. In 1918 in a short time was the capital of the West Ukrainian People's Republic. Between the wars, the city was known again as Lwów and was the centre of the Lwów Voivodeship in the Second Polish Republic. After World War II, it became part of the Soviet Union (by Stalin Djugashvili gift to Ukrainian SSR) with Ukrainian Peoples coming back to their Homeland and in 1991 of independent Ukraine. Administratively, Lviv serves as the administrative center of Lviv Oblast and has the status of city of oblast significance.

Lviv was the centre of the historical region of Galicia. The historical heart of the city, with its old buildings and cobblestone streets, survived Soviet and German occupations during World War II largely unscathed. The city has many industries and institutions of higher education such as Lviv University and Lviv Polytechnic. Lviv is also the home of many cultural institutions, including a philharmonic orchestra and the Lviv Theatre of Opera and Ballet. The historic city centre is on the UNESCO World Heritage List. Lviv celebrated its 750th anniversary with a *son et lumière* in the centre of the city in September 2006.

Contents

Lviv
Львів

City of regional significance

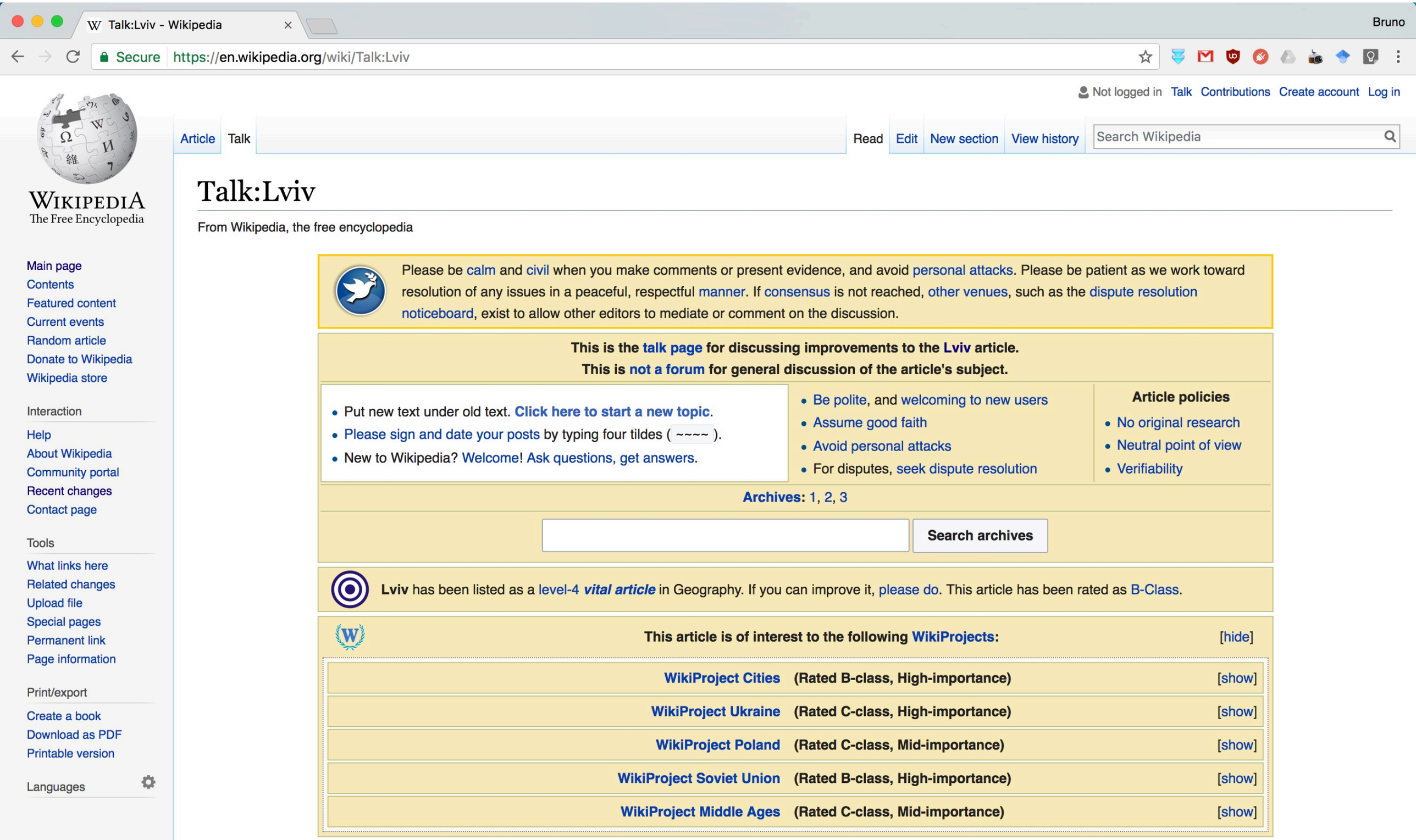
View of the Market Square

Flag

Coat of arms

Logo

Wikipedia


Talk:Lviv - Wikipedia
Bruno
Secure https://en.wikipedia.org/wiki/Talk:Lviv
Not logged in Talk Contributions Create account Log in
Article Talk Read Edit New section View history Search Wikipedia
WIKIPEDIA
The Free Encyclopedia
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store
Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page
Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Print/export
Create a book
Download as PDF
Printable version
Languages
Talk:Lviv
From Wikipedia, the free encyclopedia
Please be calm and civil when you make comments or present evidence, and avoid personal attacks. Please be patient as we work toward resolution of any issues in a peaceful, respectful manner. If consensus is not reached, other venues, such as the dispute resolution noticeboard, exist to allow other editors to mediate or comment on the discussion.
This is the talk page for discussing improvements to the Lviv article.
This is not a forum for general discussion of the article's subject.
Put new text under old text. Click here to start a new topic.
Please sign and date your posts by typing four tildes (~~~~).
New to Wikipedia? Welcome! Ask questions, get answers.
Be polite, and welcoming to new users
Assume good faith
Avoid personal attacks
For disputes, seek dispute resolution
Article policies
No original research
Neutral point of view
Verifiability
Archives: 1, 2, 3
Search archives
Lviv has been listed as a level-4 vital article in Geography. If you can improve it, please do. This article has been rated as B-Class.
This article is of interest to the following WikiProjects: [hide]
WikiProject Cities (Rated B-class, High-importance) [show]
WikiProject Ukraine (Rated C-class, High-importance) [show]
WikiProject Poland (Rated C-class, Mid-importance) [show]
WikiProject Soviet Union (Rated B-class, High-importance) [show]
WikiProject Middle Ages (Rated C-class, Mid-importance) [show]

Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
User contributions
Logs
View user groups
Upload file
Special pages
Permanent link
Page information

Print/export
Create a book
Download as PDF
Printable version

User:Kind Tennis Fan

From Wikipedia, the free encyclopedia

This editor is a **Veteran Editor IV** and is entitled to display this **Gold Editor Star**.

About me [edit]

I'm a male, born in the United Kingdom, and I've spent the vast majority of my life so far living in the south of England. My marital status is single and I currently have a girlfriend.

I registered as a Wikipedia user in July 2013, as I have many different interests and subjects that I like to read about.

Passions [edit]

Tennis

Golf

Association football (more commonly known as football or soccer.)

Rock music (In particular: melodic Alternative rock, Soft rock, Art rock and New wave music)

Pugs (Highly recommended as nice gentle pets. They have a unique character and are one of the least aggressive breeds in the world.)

Other interests [edit]

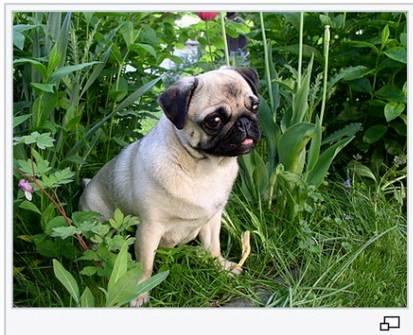
Politics

Contemporary history (Particularly the tragic conflict known as The Troubles where more than 3,500 people have been killed and over 50,000 people wounded. I would like to see the two main communities continue to work towards peace and reconciliation.)

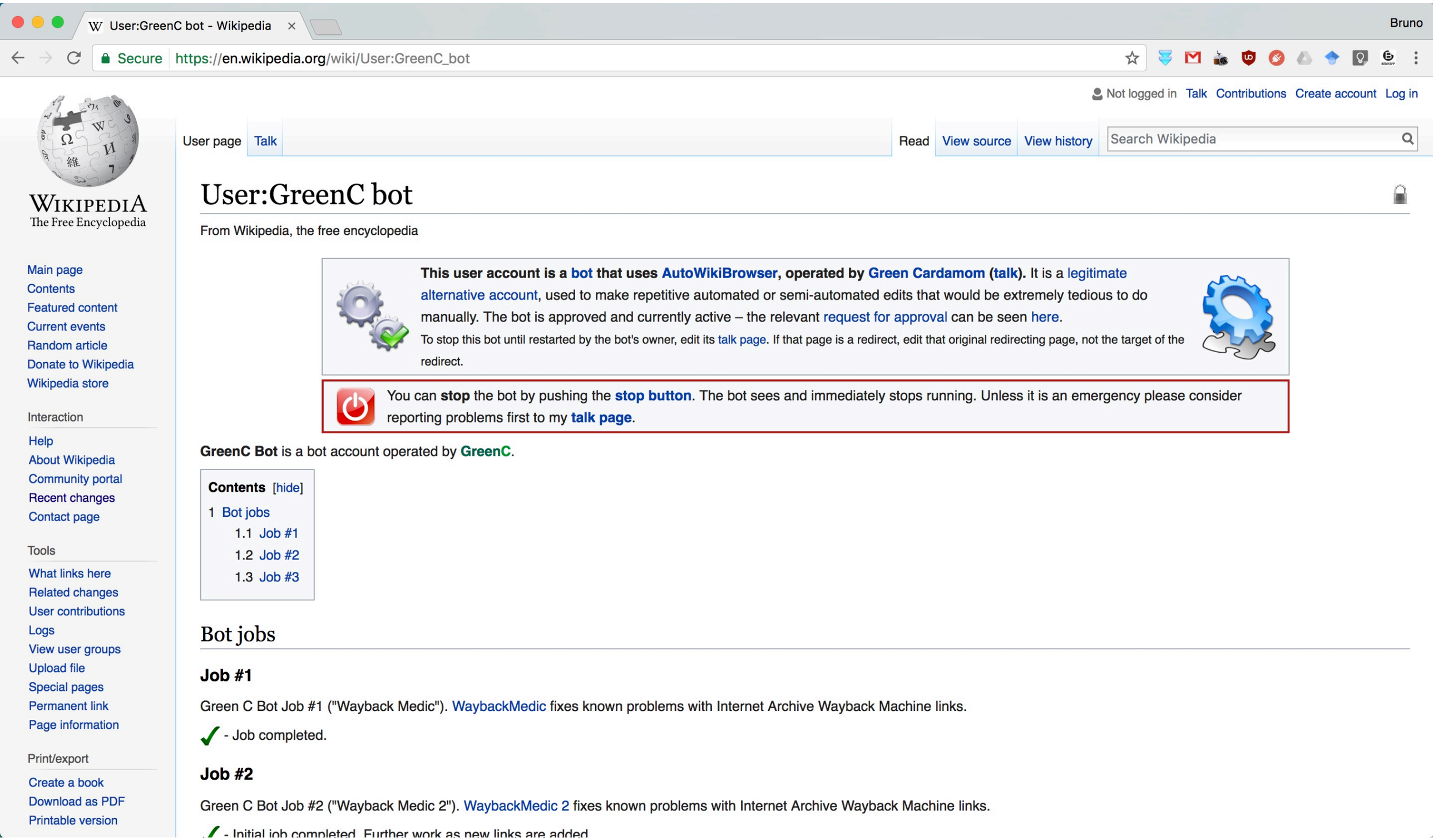
Restaurants and Cuisine

British New Wave films with an element of social realism. (Such as *Room at the Top* and *Saturday Night and Sunday Morning*.)

Psychology



Wikipedia


User:GreenC bot - Wikipedia
Bruno
Secure https://en.wikipedia.org/wiki/User:GreenC_bot
Not logged in Talk Contributions Create account Log in
User page Talk
Read View source View history
Search Wikipedia
WIKIPEDIA
The Free Encyclopedia
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store
Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page
Tools
What links here
Related changes
User contributions
Logs
View user groups
Upload file
Special pages
Permanent link
Page information
Print/export
Create a book
Download as PDF
Printable version
User:GreenC bot
From Wikipedia, the free encyclopedia
This user account is a bot that uses AutoWikiBrowser, operated by Green Cardamom (talk). It is a legitimate alternative account, used to make repetitive automated or semi-automated edits that would be extremely tedious to do manually. The bot is approved and currently active – the relevant request for approval can be seen here.
To stop this bot until restarted by the bot's owner, edit its talk page. If that page is a redirect, edit that original redirecting page, not the target of the redirect.
You can stop the bot by pushing the stop button. The bot sees and immediately stops running. Unless it is an emergency please consider reporting problems first to my talk page.
GreenC Bot is a bot account operated by GreenC.
Contents [hide]
1 Bot jobs
1.1 Job #1
1.2 Job #2
1.3 Job #3
Bot jobs
Job #1
Green C Bot Job #1 ("Wayback Medic"). WaybackMedic fixes known problems with Internet Archive Wayback Machine links.
- Job completed.
Job #2
Green C Bot Job #2 ("Wayback Medic 2"). WaybackMedic 2 fixes known problems with Internet Archive Wayback Machine links.

Wikipedia Dumps

https://dumps.wikimedia.org

Wikimedia Downloads

Bruno

Secure https://dumps.wikimedia.org

Wikimedia Downloads

If you are reading this on Wikimedia servers, please note that we have rate limited downloaders and we are capping the number of per-ip connections to 2. This will help to ensure that everyone can access the files with reasonable download times. Clients that try to evade these limits may be blocked. Our mirror sites do not have this cap.

Data downloads

The Wikimedia Foundation is requesting help to ensure that as many copies as possible are available of all Wikimedia database dumps. Please **volunteer to host a mirror** if you have access to sufficient storage and bandwidth.

Database backup dumps

A complete copy of all Wikimedia wikis, in the form of wikitext source and metadata embedded in XML. A number of raw database tables in SQL form are also available.
These snapshots are provided at the very least monthly and usually twice a month. If you are a regular user of these dumps, please consider subscribing to xmldatadumps-l for regular updates.

Mirror Sites of the XML dumps provided above

Check the complete list.

Static HTML dumps

A copy of all pages from all Wikipedia wikis, in HTML form.
These are currently not running.

DVD distributions

Wikipedia Dumps

https://dumps.wikimedia.org

- The Wikimedia foundation makes freely available regular dumps of all Wikimedia project databases.
- In particular, for the various language editions of Wikipedia, we have:
  - *.pages-articles.xml.bz2 - Complete wiki page and revision content.
  - *.stub-meta-history.xml.gz - Wiki page revision metadata
  - *.pagelinks.sql.gz - Wiki page-to-page link records
  - *.geo_tags.sql.gz - List of pages' geographical coordinates
  - *.externallinks.sql.gz - Wiki external URL link records.
  - *.page.sql.gz - Base per-page data (id, title, old restrictions, etc).
  - *.langlinks.sql.gz - Wiki interlanguage link records

Wikipedia Dumps

https://dumps.wikimedia.org

- The Wikimedia foundation makes freely available regular dumps of all Wikimedia project databases.
- In particular, for the various language editions of Wikipedia, we have:
  - *.pages-articles.xml.bz2 - Complete wiki page and revision content.
  - *.stub-meta-history.xml.gz - Wiki page revision metadata
  - *.pagelinks.sql.gz - Wiki page-to-page link records
  - *.geo_tags.sql.gz - List of pages' geographical coordinates
  - *.externallinks.sql.gz - Wiki external URL link records.
  - *.page.sql.gz - Base per-page data (id, title, old restrictions, etc).
  - *.langlinks.sql.gz - Wiki interlanguage link records

Wikipedia Dumps

https://dumps.wikimedia.org

- The Wikimedia foundation makes freely available regular dumps of all Wikimedia project databases.
- In particular, for the various language editions of Wikipedia, we have:
  - *.pages-articles.xml.bz2 - Complete wiki page and revision content.
  - *.stub-meta-history.xml.gz - Wiki page revision metadata
  - *.pagelinks.sql.gz - Wiki page-to-page link records
  - *.geo_tags.sql.gz - List of pages' geographical coordinates
  - *.externallinks.sql.gz - Wiki external URL link records.
  - *.page.sql.gz - Base per-page data (id, title, old restrictions, etc).
  - *.langlinks.sql.gz - Wiki interlanguage link records

(Wikipedia Dump "Dumping")

- I've written a simple script to easily download the most recent version of specific files for many different languages.

- You can find it in the GitHub repo: **wikidump.py**

- To customize to your needs, you just need to list the files you want in the allowed_files list and the languages you're interested in allowed_wikis

- If you want to download all the files from **acewiki** required for this tutorial, you would simply set:

```
allowed_files = ["stub-meta-history.xml.gz",
                 "geo_tags.sql.gz",
                 "langlinks.sql.gz",
                 ]

allowed_wikis = ["acewiki"]
```

- But with what you learn so far you should be able to easily write your own version 

SQL files

- Standard format, well suited for loading the data directly to a relational database (MySQL, MariaDB, PostgreSQL, etc…)

- Databases are optimized for fast querying of information, but not suitable for large scale processing where you touch all or most rows.

- mysqldump_to_csv.py - convert a wikipedia dump to a CSV file.

  - Slightly modified version of https://github.com/jamesmishra/mysqldump-to-csv

  - Available in the courses GItHub repository

  - First row is column names as defined in the SQL file

langlinks

- Just a few fields:
  - 0 - **ll_from** - The page in **this** wikipedia edition
  - 1 - **ll_lang** - The language it's linking to
  - 2 - **ll_title** - The title of the page in the **target** wikipedia edition
- This is a good example of some of the problems of working with wikipedia data, or any other self organize collaboration platform
- Many of the file formats and conventions were created in an ad hoc way, to serve one very specific need and ended up becoming adopted as "standard".
  - How can we convert the language/title pairs into a unique page_id in the target wikipedia?
  - Can we be sure that two pages didn't accidentally switch titles?
  - As pages get edited, their titles change. To when (which revision) do these titles correspond to?
  - Does a link A -> B imply a link B -> A?

Challenge - langlinks

- convert the

data/acewiki-20170420-langlinks.sql.gz

- SQL file to CSV using the mysqldum2csv.py script.

```
python mysqldump_to_csv.py data/acewiki-20170420-langlinks.sql.gz | gzip -c > data/
acewiki-20170420-langlinks.csv.gz
```

geo_tags

- Several interesting fields:
    - 0 - **gt_id** - Unique geo tag ID
    - 1 - **gt_page_id** - Corresponding Page ID
    - 2 - **gt_globe** - Not all coordinates are on Earth (Mars, Moon, Venus, Titan, etc…)
    - 4 - **gt_lat** - Latitude
    - 5 - **gt_lon** - Longitude
    - 7 - **gt_type** - city, railwaystation, landmark, airport, etc…

Challenge - geo_tags

- Convert data/enwiki-20170420-geo_tags.sql.gz to csv using mysqldump_to_csv.py

- Extract all the lat, lon pairs on planet "earth".

```
import gzip

header = {}
line_count = 0

for line in gzip.open("data/enwiki-20170420-geo_tags.csv.gz", "rt"):
    fields = line.strip().split(',')

    if line_count == 0:
        header = dict(zip(fields, range(len(fields))))

    line_count += 1

    if(fields[header["gt_globe"]] == "earth"):
        print(fields[header["gt_lat"]], fields[header["gt_lon"]])
```


wikipedia_geo_tags.py

expat - (semi) sane XML parsing

https://docs.python.org/3/library/pyexpat.html

- C library for parsing XML with bindings in most modern programming languages
- Extremely fast
- Well suited to handle large xml files:
    - Stream oriented - Reads the file line by line
    - Non-validating - doesn't check for the validity of the XML file (expensive and prone to failure)
- In Python it lives inside the xml.parsers package

```
from xml.parsers import expat
```

- The .ParserCreate() method returns a new xmlparser instance

expat - (semi) sane XML parsing

https://docs.python.org/3/library/pyexpat.html

- Defines event handlers that get called whenever it encounters something "interesting"
- The default behavior is to do nothing (very efficient!) but you can override the ones that you are interested in.
- In particular:
    - .StartElementHandler(name, attrs) - every time it encounters a <name …>
    - .EndElementHandler(name) - whenever it encounter a </name>
    - .CharacterDataHandler(data) - any textual data in between the opening and closing of a tag:
        - <name>data</name>
        - If the amount of data between these two tags is too large, it sometimes results in multiple char_data events. You should always concatenate the results as you get it
- After you overwrite the relevant methods, you can process the file by providing a file handle to .ParserFile(fp)

expat - (semi) sane XML parsing

https://docs.python.org/3/library/pyexpat.html

```python
import sys
from xml.parsers import expat

buffer = ""
level = 0

def start_element(name, attrs):
    global buffer, level
    print("\t"* level, "Opening:", name, "with attributes:", attrs)
    buffer = ""
    level += 1

def end_element(name):
    global buffer, level
    level -= 1
    print("\t"* level, "Closing:", name, "with data:", buffer)
    buffer = ""

def char_data(data):
    global buffer
    buffer += data

if __name__ == "__main__":
    p = expat.ParserCreate()
    p.StartElementHandler = start_element
    p.EndElementHandler = end_element
    p.CharacterDataHandler = char_data

    try:
        p.ParseFile(open(sys.argv[1], 'rb'))
    except Exception as e:
        print(e, file=sys.stderr)
```

expat_demo.py

Revision file format

```
<page>
  <title>Ôn Keuë</title>
  <ns>0</ns>
  <id>1</id>
  <revision>
    <id>1028</id>
    <timestamp>2008-04-13T07:53:23Z</timestamp>
    <contributor>
      <ip>125.162.38.87</ip>
    </contributor>
    <comment>New page: Jinoë droën neuh ka neutamong lam Wikipèdia  Acèh. Wikipèdia  Acèh
nyoë mantöng geu'ujoë, geukalön peuë ék na soë peudawôk peuë h'an. Meunyoë le nyang pakoë,
Wikipèdi...</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text id="815" bytes="3106" />
    <sha1>43iy7hfjh19xt1683z27ii0ie35z9am</sha1>
  </revision>
  <revision>
    <id>1029</id>
    <parentid>1028</parentid>
    <timestamp>2008-04-13T08:01:10Z</timestamp>
    <contributor>
      <username>Si Gam Acèh</username>
      <id>0</id>
    </contributor>
    <comment>Removing all content from page</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text id="816" bytes="0" />
    <sha1>phoiac9h4m842xq45sp7s6u21eteeq1</sha1>
  </revision>
</page>
```

data/page.xml

```
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.10/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mediawiki.org/xml/export-0.10/ http://www.mediawiki.org/xml/export-0.10.xsd"
version="0.10" xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <dbname>enwiki</dbname>
    <base>https://en.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.29.0-wmf.20</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</namespace>
      <namespace key="-1" case="first-letter">Special</namespace>
      <namespace key="0" case="first-letter" />
      <namespace key="1" case="first-letter">Talk</namespace>
      <namespace key="2" case="first-letter">User</namespace>
      <namespace key="3" case="first-letter">User talk</namespace>
      <namespace key="4" case="first-letter">Wikipedia</namespace>
      <namespace key="5" case="first-letter">Wikipedia talk</namespace>
      <namespace key="6" case="first-letter">File</namespace>
      <namespace key="7" case="first-letter">File talk</namespace>
      <namespace key="8" case="first-letter">MediaWiki</namespace>
      <namespace key="9" case="first-letter">MediaWiki talk</namespace>
      <namespace key="10" case="first-letter">Template</namespace>
      <namespace key="11" case="first-letter">Template talk</namespace>
      <namespace key="12" case="first-letter">Help</namespace>
      <namespace key="13" case="first-letter">Help talk</namespace>
      <namespace key="14" case="first-letter">Category</namespace>
      <namespace key="15" case="first-letter">Category talk</namespace>
      <namespace key="100" case="first-letter">Portal</namespace>
      <namespace key="101" case="first-letter">Portal talk</namespace>
      <namespace key="108" case="first-letter">Book</namespace>
      <namespace key="109" case="first-letter">Book talk</namespace>
      <namespace key="118" case="first-letter">Draft</namespace>
      <namespace key="119" case="first-letter">Draft talk</namespace>
      <namespace key="446" case="first-letter">Education Program</namespace>
      <namespace key="447" case="first-letter">Education Program talk</namespace>
      <namespace key="710" case="first-letter">TimedText</namespace>
      <namespace key="711" case="first-letter">TimedText talk</namespace>
      <namespace key="828" case="first-letter">Module</namespace>
      <namespace key="829" case="first-letter">Module talk</namespace>
      <namespace key="2300" case="first-letter">Gadget</namespace>
      <namespace key="2301" case="first-letter">Gadget talk</namespace>
      <namespace key="2302" case="case-sensitive">Gadget definition</namespace>
      <namespace key="2303" case="case-sensitive">Gadget definition talk</namespace>
    </namespaces>
  </siteinfo>
```

```
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.10/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.mediawiki.org/xml/
export-0.10/ http://www.mediawiki.org/xml/export-0.10.xsd" version="0.10" xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <dbname>enwiki</dbname>
    <base>https://en.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.29.0-wmf.20</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</namespace>
      <namespace key="-1" case="first-letter">Special</namespace>
      <namespace key="0" case="first-letter" />
      <namespace key="1" case="first-letter">Talk</namespace>
      <namespace key="2" case="first-letter">User</namespace>
      <namespace key="3" case="first-letter">User talk</namespace>
      <namespace key="4" case="first-letter">Wikipedia</namespace>
      <namespace key="5" case="first-letter">Wikipedia talk</namespace>
      <namespace key="6" case="first-letter">File</namespace>
      <namespace key="7" case="first-letter">File talk</namespace>
      <namespace key="8" case="first-letter">MediaWiki</namespace>
      <namespace key="9" case="first-letter">MediaWiki talk</namespace>
      <namespace key="10" case="first-letter">Template</namespace>
      <namespace key="11" case="first-letter">Template talk</namespace>
      <namespace key="12" case="first-letter">Help</namespace>
      <namespace key="13" case="first-letter">Help talk</namespace>
      <namespace key="14" case="first-letter">Category</namespace>
      <namespace key="15" case="first-letter">Category talk</namespace>
      <namespace key="100" case="first-letter">Portal</namespace>
      <namespace key="101" case="first-letter">Portal talk</namespace>
      <namespace key="108" case="first-letter">Book</namespace>
      <namespace key="109" case="first-letter">Book talk</namespace>
      <namespace key="118" case="first-letter">Draft</namespace>
      <namespace key="119" case="first-letter">Draft talk</namespace>
      <namespace key="446" case="first-letter">Education Program</namespace>
      <namespace key="447" case="first-letter">Education Program talk</namespa
      <namespace key="710" case="first-letter">TimedText</namespace>
      <namespace key="711" case="first-letter">TimedText talk</namespace>
      <namespace key="828" case="first-letter">Module</namespace>
      <namespace key="829" case="first-letter">Module talk</namespace>
      <namespace key="2300" case="first-letter">Gadget</namespace>
      <namespace key="2301" case="first-letter">Gadget talk</namespace>
      <namespace key="2302" case="case-sensitive">Gadget definition</namespace>
      <namespace key="2303" case="case-sensitive">Gadget definition talk</namespace>
    </namespaces>
  </siteinfo>
```

Wikipedia data structure

Namespaces

Subject namespaces		Talk namespaces	
0	(Main/Article)	Talk	1
2	User	User talk	3
4	Wikipedia	Wikipedia talk	5
6	File	File talk	7
8	MediaWiki	MediaWiki talk	9
10	Template	Template talk	11
12	Help	Help talk	13
14	Category	Category talk	15
100	Portal	Portal talk	101
108	Book	Book talk	109
118	Draft	Draft talk	119
446	Education Program	Education Program talk	447
710	TimedText	TimedText talk	711
828	Module	Module talk	829
2300	Gadget	Gadget talk	2301
2302	Gadget definition	Gadget definition talk	2303

Virtual namespaces

-1	Special
-2	Media

Revision file format

```
<page>
  <title>Ôn Keuë</title>
  <ns>0</ns>
  <id>1</id>
  <revision>
    <id>1028</id>
    <timestamp>2008-04-13T07:53:23Z</timestamp>
    <contributor>
      <ip>125.162.38.87</ip>
    </contributor>
    <comment>New page: Jinoë droën neuh ka neutamong lam Wikipèdia  Acèh. Wikipèdia  Acèh
nyoë mantöng geu'ujoë, geukalön peuë ék na soë peudawôk peuë h'an. Meunyoë le nyang pakoë,
Wikipèdi...</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text id="815" bytes="3106" />
    <sha1>43iy7hfjh19xt1683z27ii0ie35z9am</sha1>
  </revision>
  <revision>
    <id>1029</id>
    <parentid>1028</parentid>
    <timestamp>2008-04-13T08:01:10Z</timestamp>
    <contributor>
      <username>Si Gam Acèh</username>
      <id>0</id>
    </contributor>
    <comment>Removing all content from page</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text id="816" bytes="0" />
    <sha1>phoiac9h4m842xq45sp7s6u21eteeq1</sha1>
  </revision>
</page>
```

data/page.xml

Challenge - expat

- Extract the article revision information onto a csv file with the format:

page_id, revision_id, timestamp, title

In the file:

data/acewiki-20170420-stub-meta-history.xml.gz

There are 3 <id> tags. You have to keep track of which one you're in!

```
<page>
  <title>Ôn Keuë</title>
  <ns>0</ns>
  <id>1</id>
  <revision>
    <id>1028</id>
    <timestamp>2008-04-13T07:53:23Z</timestamp>
    <contributor>
      <ip>125.162.38.87</ip>
    </contributor>
    <comment>New page: Jinoë droën neuh ka neutamong
lam Wikipèdia  Acèh. Wikipèdia  Acèh nyoë mantöng
geu'ujoë, geukalön peuë ék na soë peudawôk peuë h'an
Meunyoë le nyang pakoë, Wikipèdi...</comment>
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text id="815" bytes="3106" />
    <sha1>43iy7hfjh19xt1683z27ii0ie35z9am</sha1>
  </revision>
  <revision>
    <id>1029</id>
    <parentid>1028</parentid>
    <timestamp>2008-04-13T08:01:10Z</timestamp>
    <contributor>
      <username>Si Gam Acèh</username>
      <id>0</id>
    </contributor>
    <comment>Removing all content from page</comment
    <model>wikitext</model>
    <format>text/x-wiki</format>
    <text id="816" bytes="0" />
    <sha1>phoiac9h4m842xq45sp7s6u21eteeq1</sha1>
  </revision>
</page>
```

data/page.xml

Challenge - expat

```
import sys
import gzip
from xml.parsers import expat

isContributor = False
isArticle = False
isPage = False
isData = False
buffer = u""
page_id = None
timestamp = None
revision_id = None

fields = set(["timestamp", "page", "id", "ns", "revision", "contributor"])

def start_element(name, attrs):
    global buffer, isData, isPage, isContributor

    if name in fields:
        buffer = ""

        if name == "page":
            isPage = True
        elif name == "revision":
            isPage = False
        elif name == "contributor":
            isContributor = True
        else:
            isData = True
```

wikipedia_revisions.py

Challenge - expat

```
def end_element(name):
    global buffer, isData, isPage, isArticle, isContributor, timestamp, page_id, revision_id

    if name in fields:
        if name == "ns":
            if int(buffer) == 0:
                isArticle = True
            else:
                isArticle = False
        elif name == "timestamp":
            timestamp = buffer
        elif name == "id":
            if isPage:
                page_id = buffer
                isPage = False
            elif not isContributor:
                revision_id = buffer
        elif name == "revision":
            if isArticle:
                print(",".join([page_id, revision_id, timestamp]))
        elif name == "page":
            isArticle = False
        elif name == "contributor":
            isContributor = False

    buffer = ""
    isData = False
```


wikipedia_revisions.py

Challenge - expat

```
def char_data(data):
    global isData, buffer

    if isData:
        buffer += data

if __name__ == "__main__":
    p = expat.ParserCreate()

    p.StartElementHandler = start_element
    p.EndElementHandler = end_element
    p.CharacterDataHandler = char_data

    print(",".join(["page_id", "revision_id", "timestamp", "title"]))

    try:
        p.ParseFile(gzip.open(sys.argv[1]))
    except Exception as e:
        print(e, file=sys.stderr)
```


wikipedia_revisions.py

Challenge - Matching titles

- As we saw before, matching titles and page_id is not easy. The only place where the two field appear together is in the revisions files. Fortunately, we already know how to process those.

- Even more fortunately, the Wikimedia foundation also makes available the stub-meta-current.xml.gz that have a similar format to the stub-meta-history.xml.gz files but include only the current revision of each page.

- This challenge has two parts:

    - Convert data/abwiki-20170420-stub-meta-current.xml.gz to CSV

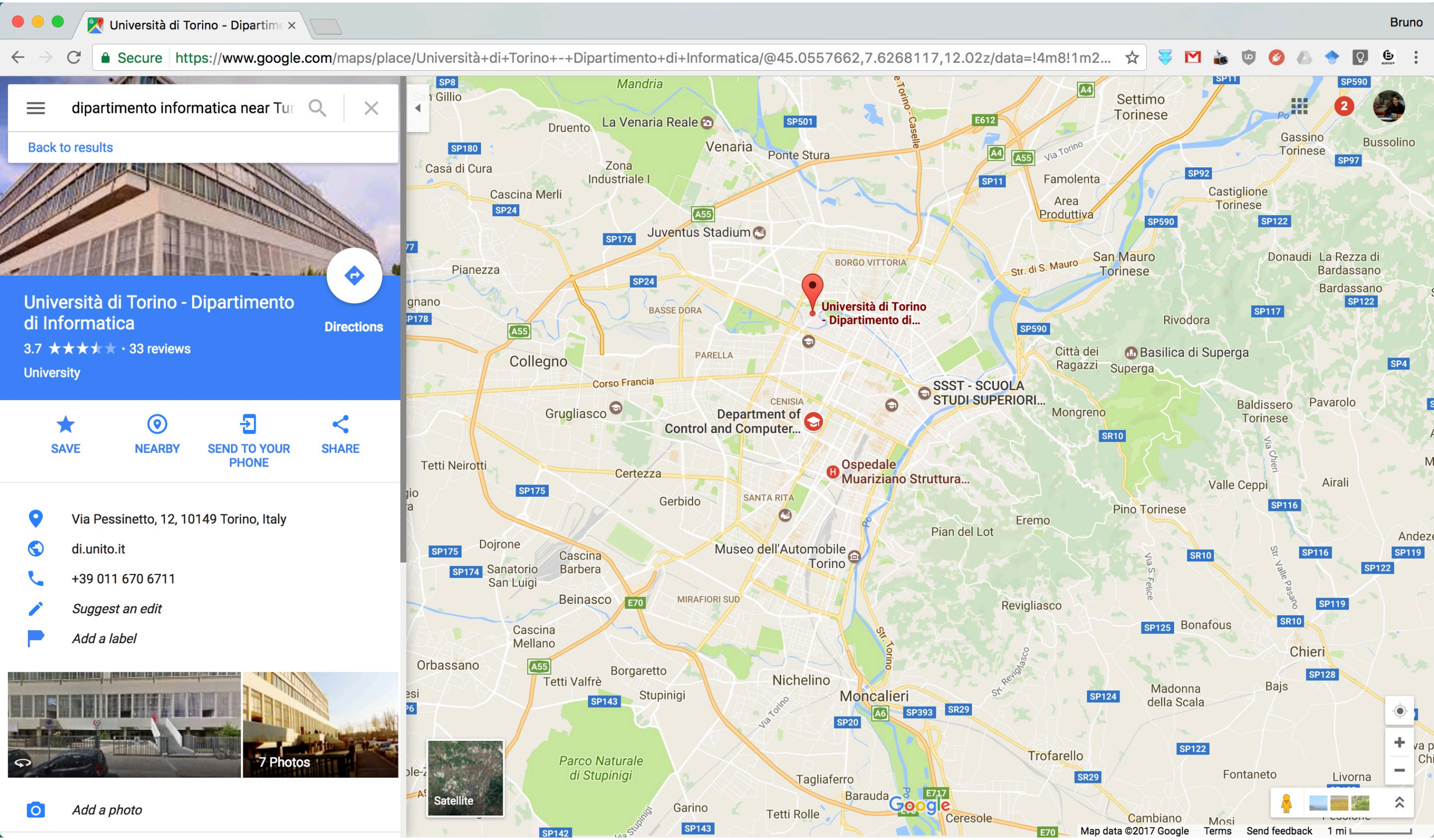
    - Use your newly generated data/abwiki-20170420-stub-meta-current.csv.gz to match the titles in data/acewiki-20170420-langlinks.csv.gz to the page_id in for the abwiki wikipedia edition.

```python
import gzip

header = {}
line_count = 0

ll_from = {}
for line in gzip.open("data/acewiki-20170420-langlinks.csv.gz", "rt"):
    fields = line.strip().split(',')

    if line_count == 0:
        header = dict(zip(fields, range(len(fields))))

    line_count += 1

    if fields[header["ll_lang"]] == "ab":
        key = fields[header["ll_title"]]

        ll_from[key] = fields[header["ll_from"]]

header = {}
line_count = 0

for line in gzip.open("data/abwiki-20170420-stub-meta-current.csv.gz", "rt"):
    fields = line.strip().split(',')

    if line_count == 0:
        header = dict(zip(fields, range(len(fields))))

    line_count += 1

    title = fields[header["title"]]

    if title in ll_from:
        print("ace", ll_from[title], "ab", fields[header["page_id"]])
```

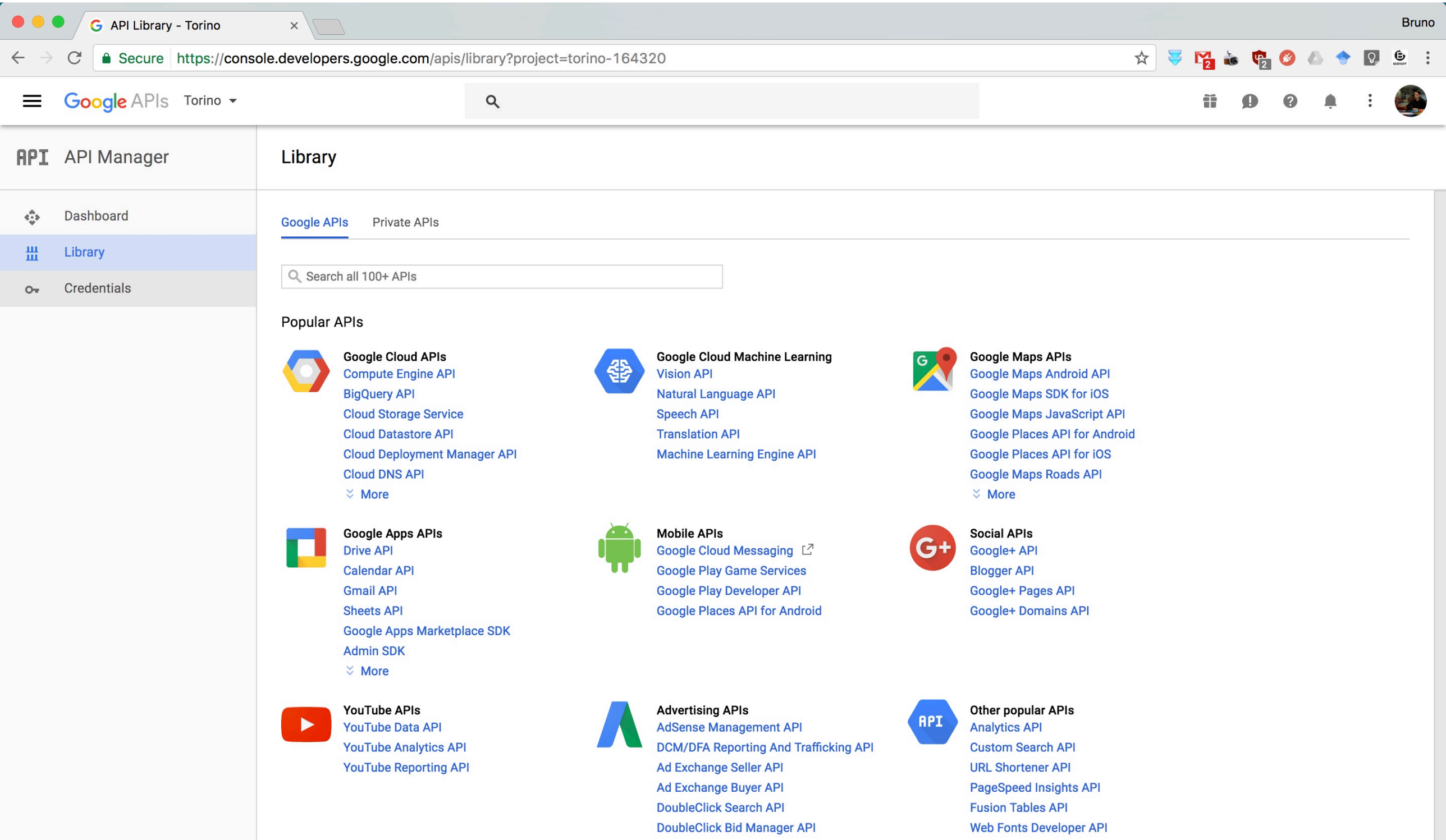
wikipedia_langlinks.py

Google Maps APIs

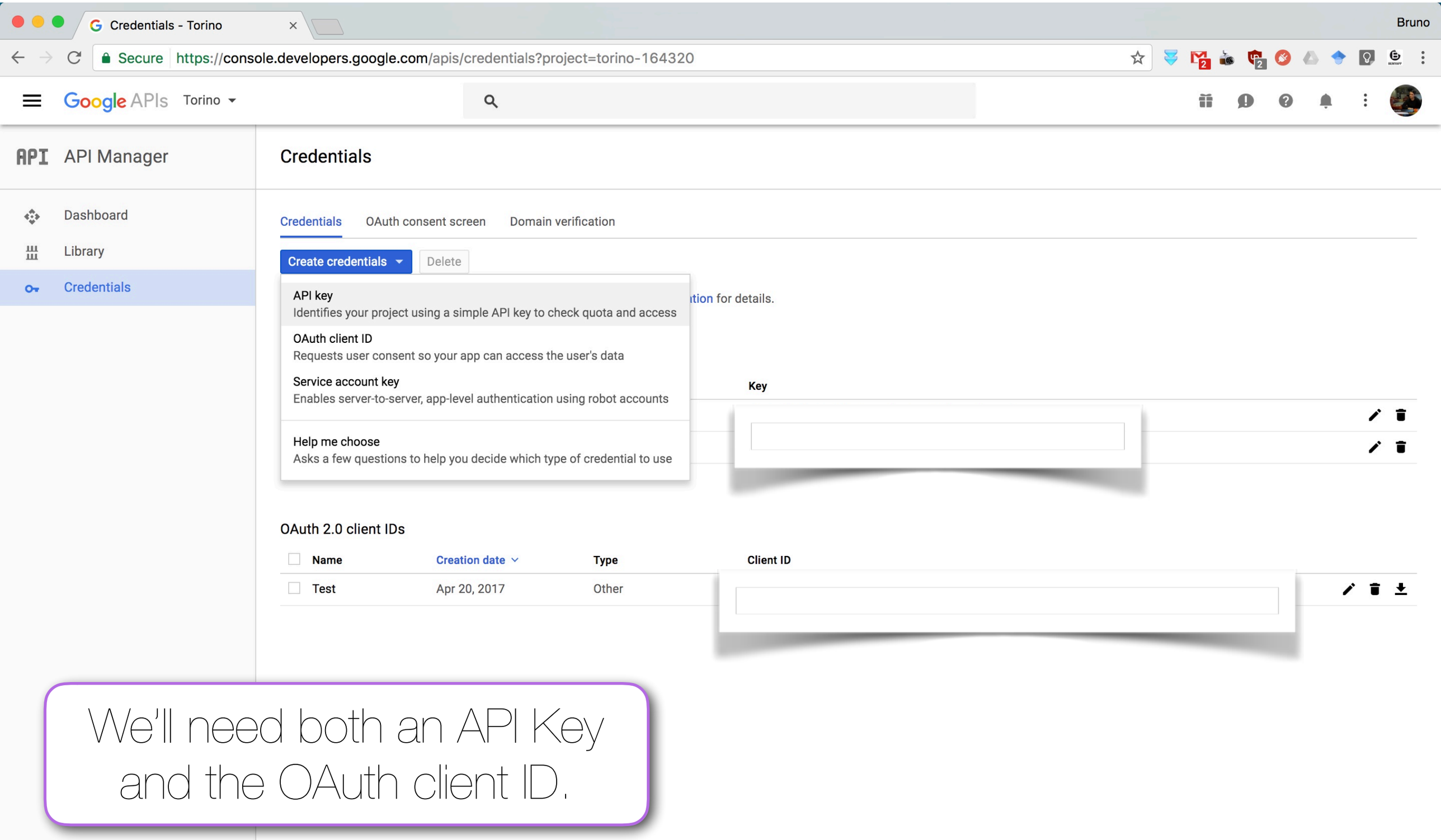
Google Maps


Università di Torino - Dipartime
Bruno
Secure
https://www.google.com/maps/place/Università+di+Torino+-+Dipartimento+di+Informatica/@45.0557662,7.6268117,12.02z/data=!4m8!1m2...
dipartimento informatica near Tur
Back to results
Università di Torino - Dipartimento di Informatica
3.7 · 33 reviews
University
Directions
SAVE
NEARBY
SEND TO YOUR PHONE
SHARE
Via Pessinetto, 12, 10149 Torino, Italy
di.unito.it
+39 011 670 6711
Suggest an edit
Add a label
7 Photos
Add a photo
Satellite
Mandria
La Venaria Reale
Venaria
Ponte Stura
Druento
Zona Industriale I
Casa di Cura
Cascina Merli
Juventus Stadium
Pianezza
BORGO VITTORIA
Università di Torino - Dipartimento di...
BASSE DORA
PARELLA
Collegno
Corso Francia
Grugliasco
CENISIA
Department of Control and Computer...
SSST - SCUOLA STUDI SUPERIORI...
Tetti Neirotti
Certezza
Gerbido
SANTA RITA
Ospedale Mauriziano Struttura...
Dojrone
Cascina Barbera
Sanatorio San Luigi
Beinasco
MIRAFIORI SUD
Museo dell'Automobile Torino
Cascina Mellano
Orbassano
Borgaretto
Tetti Valfrè
Stupinigi
Nichelino
Moncalieri
Parco Naturale di Stupinigi
Tagliaferro
Barauda
Garino
Tetti Rolle
Ceresole
Settimo Torinese
Gassino Torinese
Bussolino
Famolenta
Area Produttiva
Castiglione Torinese
San Mauro Torinese
Donaudi
La Rezza di Bardassano
Bardassano
Rivodora
Città dei Ragazzi
Basilica di Superga
Superga
Mongreno
Baldissero Torinese
Pavarolo
Valle Ceppi
Airali
Pino Torinese
Eremo
Pian del Lot
Revigliasco
Bonafous
Chieri
Bajs
Madonna della Scala
Trofarello
Fontaneto
Livorna
Cambiano
Mosi
Andezeno
Via Torino
Str. di S. Mauro
Str. Torino
Str. Revigliasco
Via S. Felice
Str. Valle Pasano
Via Chieri
Po
Google
Map data ©2017 Google
Terms
Send feedback
1 mi

Google APIs

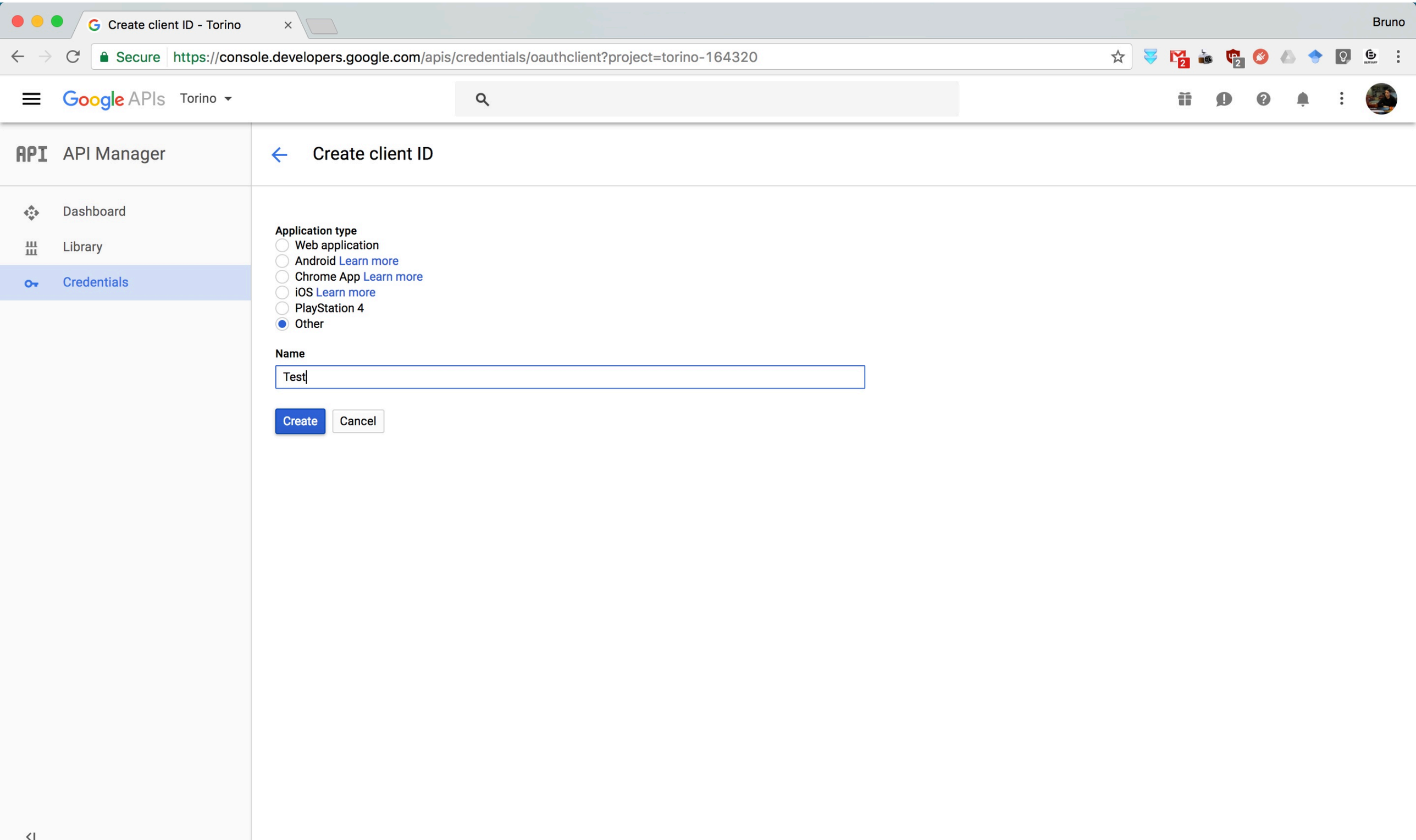

API Library - Torino
Bruno
Secure https://console.developers.google.com/apis/library?project=torino-164320
Google APIs Torino
API Manager
Library
Dashboard
Library
Credentials
Google APIs
Private APIs
Search all 100+ APIs
Popular APIs
Google Cloud APIs
Compute Engine API
BigQuery API
Cloud Storage Service
Cloud Datastore API
Cloud Deployment Manager API
Cloud DNS API
More
Google Cloud Machine Learning
Vision API
Natural Language API
Speech API
Translation API
Machine Learning Engine API
Google Maps APIs
Google Maps Android API
Google Maps SDK for iOS
Google Maps JavaScript API
Google Places API for Android
Google Places API for iOS
Google Maps Roads API
More
Google Apps APIs
Drive API
Calendar API
Gmail API
Sheets API
Google Apps Marketplace SDK
Admin SDK
More
Mobile APIs
Google Cloud Messaging
Google Play Game Services
Google Play Developer API
Google Places API for Android
Social APIs
Google+ API
Blogger API
Google+ Pages API
Google+ Domains API
YouTube APIs
YouTube Data API
YouTube Analytics API
YouTube Reporting API
Advertising APIs
AdSense Management API
DCM/DFA Reporting And Trafficking API
Ad Exchange Seller API
Ad Exchange Buyer API
DoubleClick Search API
DoubleClick Bid Manager API
Other popular APIs
Analytics API
Custom Search API
URL Shortener API
PageSpeed Insights API
Fusion Tables API
Web Fonts Developer API

Google Credentials

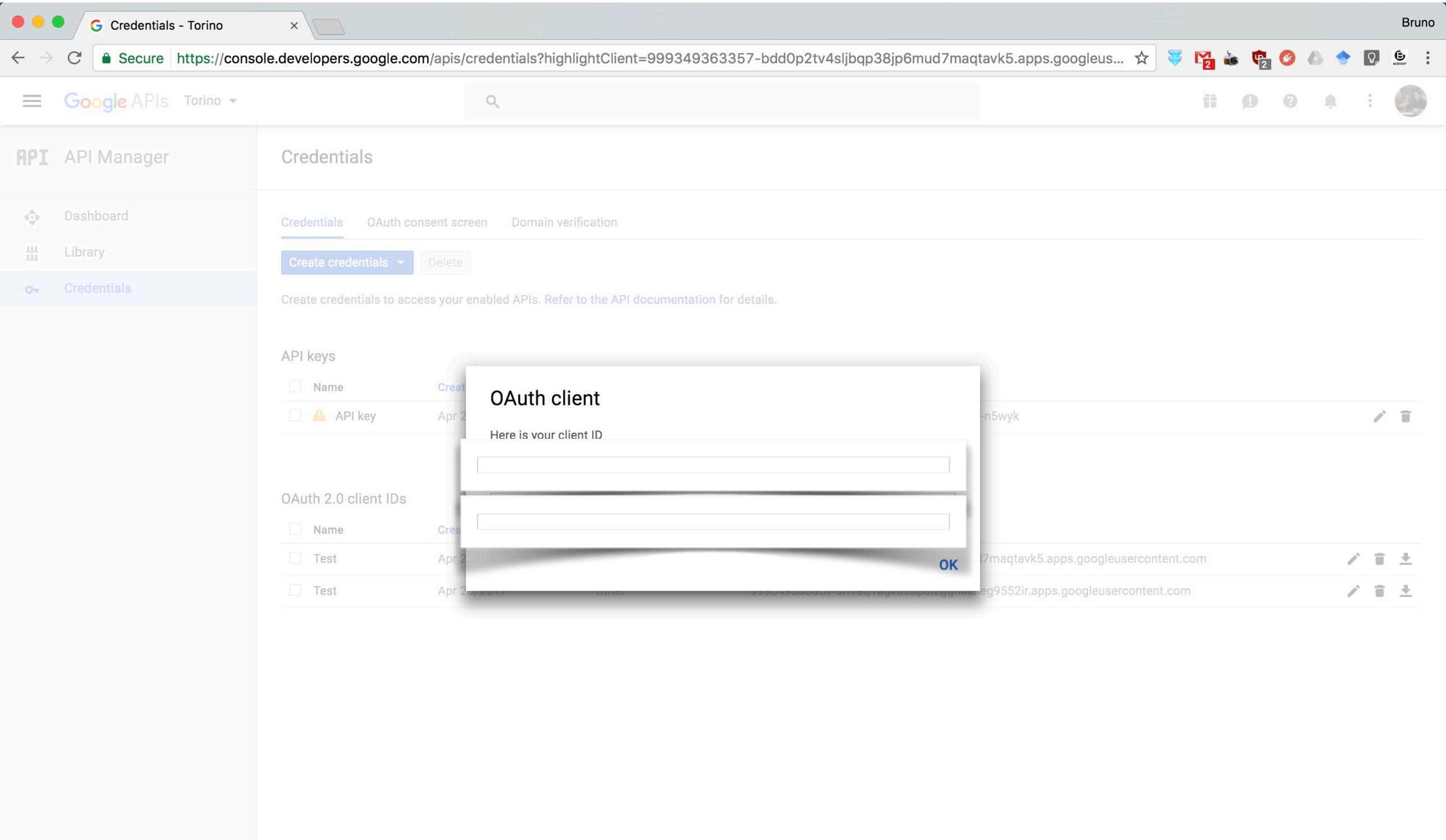


We'll need both an API Key and the OAuth client ID.

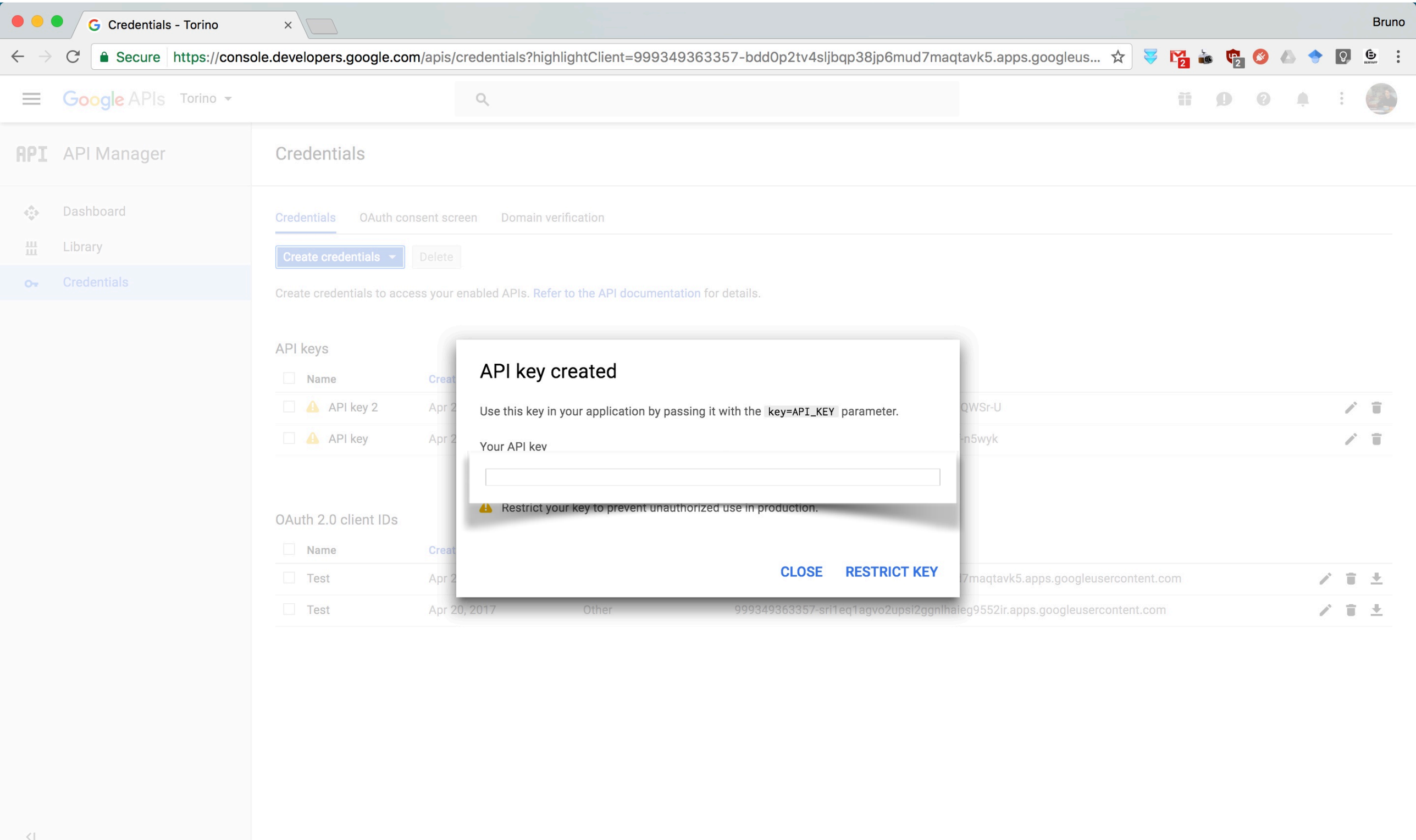
Google Credentials



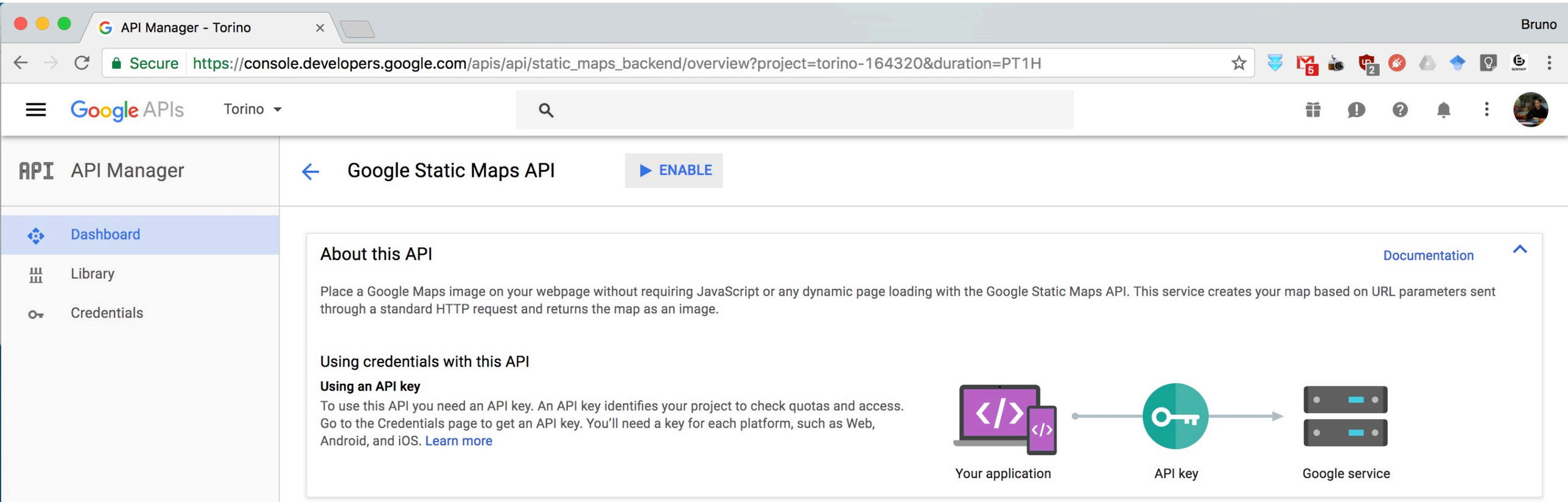
Google Credentials



Google Credentials

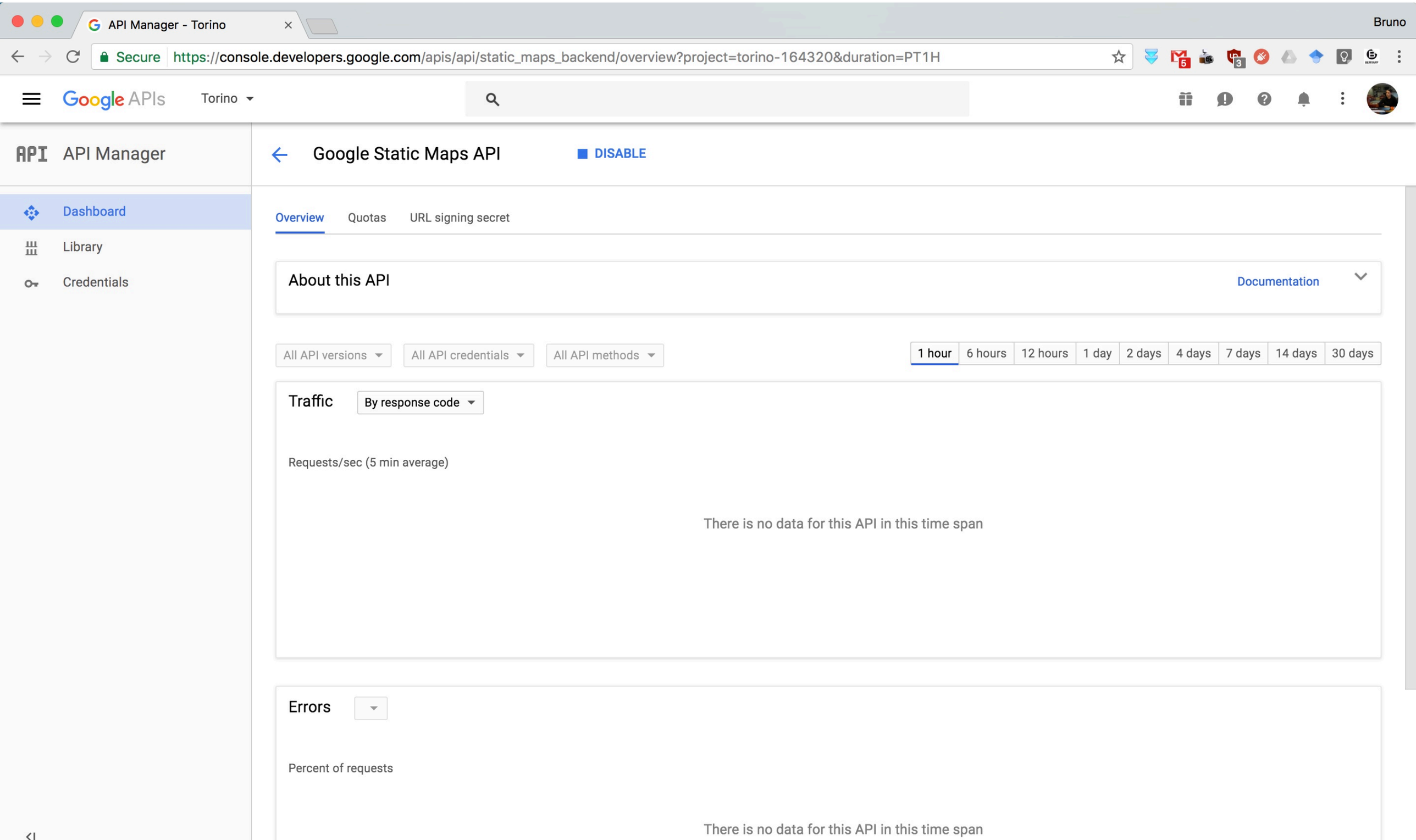


API Authorization



Each API has to be "Enabled" for this specific set of credentials

API Authorization



Google Maps APIs

https://github.com/googlemaps/google-maps-services-python

- As before, we store our credentials in a dictionary kept in an external file

```
accounts = {
    "torino": {
        "api_key": "API_KEY",
        "client_id": "CLIENT_ID",
        "client_secret": "CLIENT_SECRET"
    }
}
```

- Google provides dozens of different APIs for various purposes.
- The googlemaps Python library provides easy access to some of the GIS related ones:
  - Directions API
  - Distance Matrix API
  - Elevation API
  - Geocoding API
  - Geolocation API
  - Time Zone API
  - Roads API
  - Places API

google_accounts_STUB.py

Google Maps APIs

https://github.com/googlemaps/google-maps-services-python

- As before, we store our credentials in a dictionary kept in an external file

```
accounts = {
    "torino": {
        "api_key": "API_KEY",
        "client_id": "CLIENT_ID",
        "client_secret": "CLIENT_SECRET"
    }
}
```

- Google provides dozens of different APIs for various purposes.

- The googlemaps Python library provides easy access to some of the GIS related ones:
  - Directions API -
  - Distance Matrix API
  - Elevation API
  - Geocoding API
  - Geolocation API
  - Time Zone API
  - Roads API
  - Places API

google_accounts_STUB.py

Google Geocoding API

https://developers.google.com/maps/documentation/geocoding/start

- Allows us to obtain the latitude and longitude of a specific location.

```
import googlemaps
from google_accounts import accounts

app = accounts["torino"]
gmaps = googlemaps.Client(key=app["api_key"])

geocode_result = gmaps.geocode('JFK Airport')

print(geocode_result[0]["geometry"]["location"])
```

- As with previous APIs, it returns a JSON object with a list of results.
- Each result provides detailed information such as the address, type of location and a unique id.
- Within the "geometry" field there is:
  - a "location" field with the latitude and longitude.
  - a "viewport" field with the northeast and southwest coordinates of the respective bbox.
- The "type_id" can be used with the Places API to obtain further information, reviews, etc…

google_geocode.py

Google Geocoding API

https://developers.google.com/maps/documentation/geocoding/start

```
[{'address_components': [{'long_name': 'John F. Kennedy International Airport',
                          'short_name': 'John F. Kennedy International Airport',
                          'types': ['airport',
                                    'establishment',
                                    'point_of_interest']},
                         {'long_name': 'Queens',
                          'short_name': 'Queens',
                          'types': ['political',
                                    'sublocality',
                                    'sublocality_level_1']},
                         {'long_name': 'Queens County',
                          'short_name': 'Queens County',
                          'types': ['administrative_area_level_2',
                                    'political']},
                         {'long_name': 'New York',
                          'short_name': 'NY',
                          'types': ['administrative_area_level_1',
                                    'political']},
                         {'long_name': 'United States',
                          'short_name': 'US',
                          'types': ['country', 'political']},
                         {'long_name': '11430',
                          'short_name': '11430',
                          'types': ['postal_code']}],
'formatted_address': 'John F. Kennedy International Airport, Queens, NY '
                     '11430, USA',
'geometry': {'location': {'lat': 40.6413111, 'lng': -73.77813909999999},
             'location_type': 'APPROXIMATE',
             'viewport': {'northeast': {'lat': 40.64266008029149,
                                        'lng': -73.7767901197085},
                          'southwest': {'lat': 40.63996211970849,
                                        'lng': -73.7794880802915}}},
'place_id': 'ChIJR0lA1VBmwokR8BGfSBOyT-w',
'types': ['airport', 'establishment', 'point_of_interest']}]
```

Google Geocoding API

https://developers.google.com/maps/documentation/geocoding/start

```
[{'address_components': [{'long_name': 'John F. Kennedy International Airport',
                          'short_name': 'John F. Kennedy International Airport',
                          'types': ['airport',
                                    'establishment',
                                    'point_of_interest']},
                         {'long_name': 'Queens',
                          'short_name': 'Queens',
                          'types': ['political',
                                    'sublocality',
                                    'sublocality_level_1']},
                         {'long_name': 'Queens County',
                          'short_name': 'Queens County',
                          'types': ['administrative area level 2',
                                    'political']},
                         {'long_name': 'New York',
                          'short_name': 'NY',
                          'types': ['administrative_area_level_1',
                                    'political']},
                         {'long_name': 'United States',
                          'short_name': 'US',
                          'types': ['country', 'political']},
                         {'long_name': '11430',
                          'short_name': '11430',
                          'types': ['postal_code']}],
'formatted_address': 'John F. Kennedy International Airport, Queens, NY '
                     '11430, USA',
'geometry': {'location': {'lat': 40.6413111, 'lng': -73.77813909999999},
             'location_type': 'APPROXIMATE',
             'viewport': {'northeast': {'lat': 40.64266008029149,
                                        'lng': -73.7767901197085},
                          'southwest': {'lat': 40.63996211970849,
                                        'lng': -73.7794880802915}}},
'place_id': 'ChIJR0lA1VBmwokR8BGfSBOyT-w',
'types': ['airport', 'establishment', 'point_of_interest']}]
```

@bgon

Google Reverse Geocoding

https://developers.google.com/maps/documentation/geocoding/start

- Allows us to discover what a given set of coordinates represents
- .reverse_geocode((lat, lon)) - Just requires a lat/lon tuple.
- Google has access to many layers of GIS information and the results reflect this. The output is a list of results at various degrees of precision.
    - Building,
    - Nearest Road
    - County
    - Zip code
    - State
    - Country
    - etc…

Google Reverse Geocoding

https://developers.google.com/maps/documentation/geocoding/start


```
{'address_components': [{'long_name': 'New York',
                         'short_name': 'New York',
                         'types': ['locality', 'political']},
                        {'long_name': 'New York',
                         'short_name': 'NY',
                         'types': ['administrative_area_level_1', 'political']},
                        {'long_name': 'United States',
                         'short_name': 'US',
                         'types': ['country', 'political']}],
'formatted_address': 'New York, NY, USA',
'geometry': {'bounds': {'northeast': {'lat': 40.9175771,
                                      'lng': -73.70027209999999},
                        'southwest': {'lat': 40.4773991,
                                      'lng': -74.25908989999999}},
             'location': {'lat': 40.7127837, 'lng': -74.0059413},
             'location_type': 'APPROXIMATE',
             'viewport': {'northeast': {'lat': 40.9152555,
                                        'lng': -73.70027209999999},
                          'southwest': {'lat': 40.4960439,
                                        'lng': -74.2557349}}},
'place_id': 'ChIJOwg_06VPwokRYv534QaPC8g',
'types': ['locality', 'political']}
```

Google Reverse Geocoding

https://developers.google.com/maps/documentation/geocoding/start


```
{'address_components': [{'long_name': 'New York',
                         'short_name': 'New York',
                         'types': ['locality', 'political']},
                        {'long_name': 'New York',
                         'short_name': 'NY',
                         'types': ['administrative_area_level_1', 'political']},
                        {'long_name': 'United States',
                         'short_name': 'US',
                         'types': ['country', 'political']}],
'formatted_address': 'New York, NY, USA',
'geometry': {'bounds': {'northeast': {'lat': 40.9175771,
                                      'lng': -73.70027209999999},
                        'southwest': {'lat': 40.4773991,
                                      'lng': -74.25908989999999}},
             'location': {'lat': 40.7127837, 'lng': -74.0059413},
             'location_type': 'APPROXIMATE',
             'viewport': {'northeast': {'lat': 40.9152555,
                                        'lng': -73.70027209999999},
                          'southwest': {'lat': 40.4960439,
                                        'lng': -74.2557349}}},
'place_id': 'ChIJOwg_06VPwokRYv534QaPC8g',
'types': ['locality', 'political']}
```

Challenge - Google Reverse Geocoding

https://developers.google.com/maps/documentation/geocoding/start

- Reverse Geocode the coordinates:

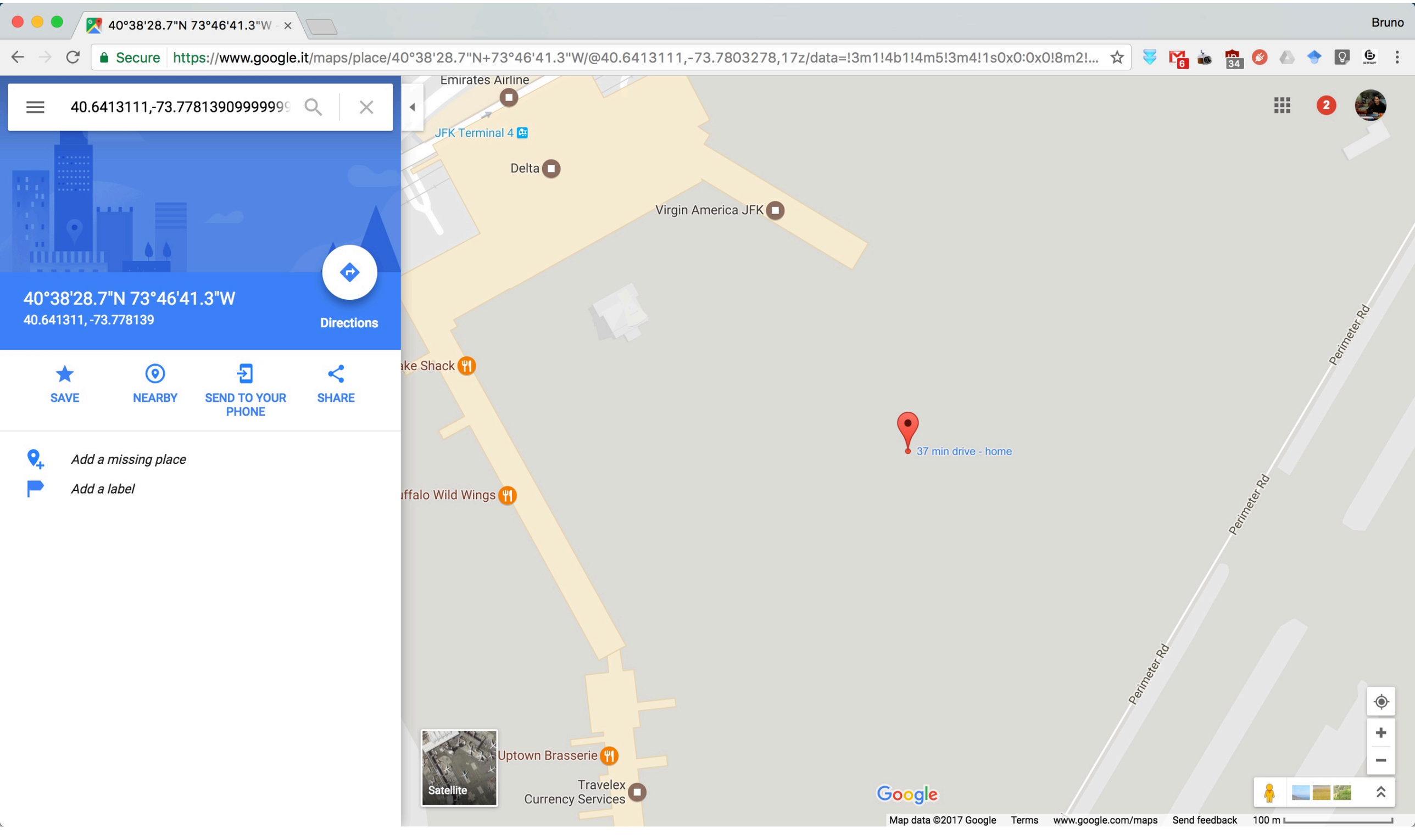
40.6413111,-73.77813909999999

- and print the “types” and “formated_address” of each of the results

```python
import googlemaps
from google_accounts import accounts

app = accounts["torino"]
gmaps = googlemaps.Client(key=app["api_key"])

reverse_result = gmaps.reverse_geocode((40.6413111,-73.77813909999999))

for result in reverse_result:
    print(result["types"], result["formatted_address"])
```

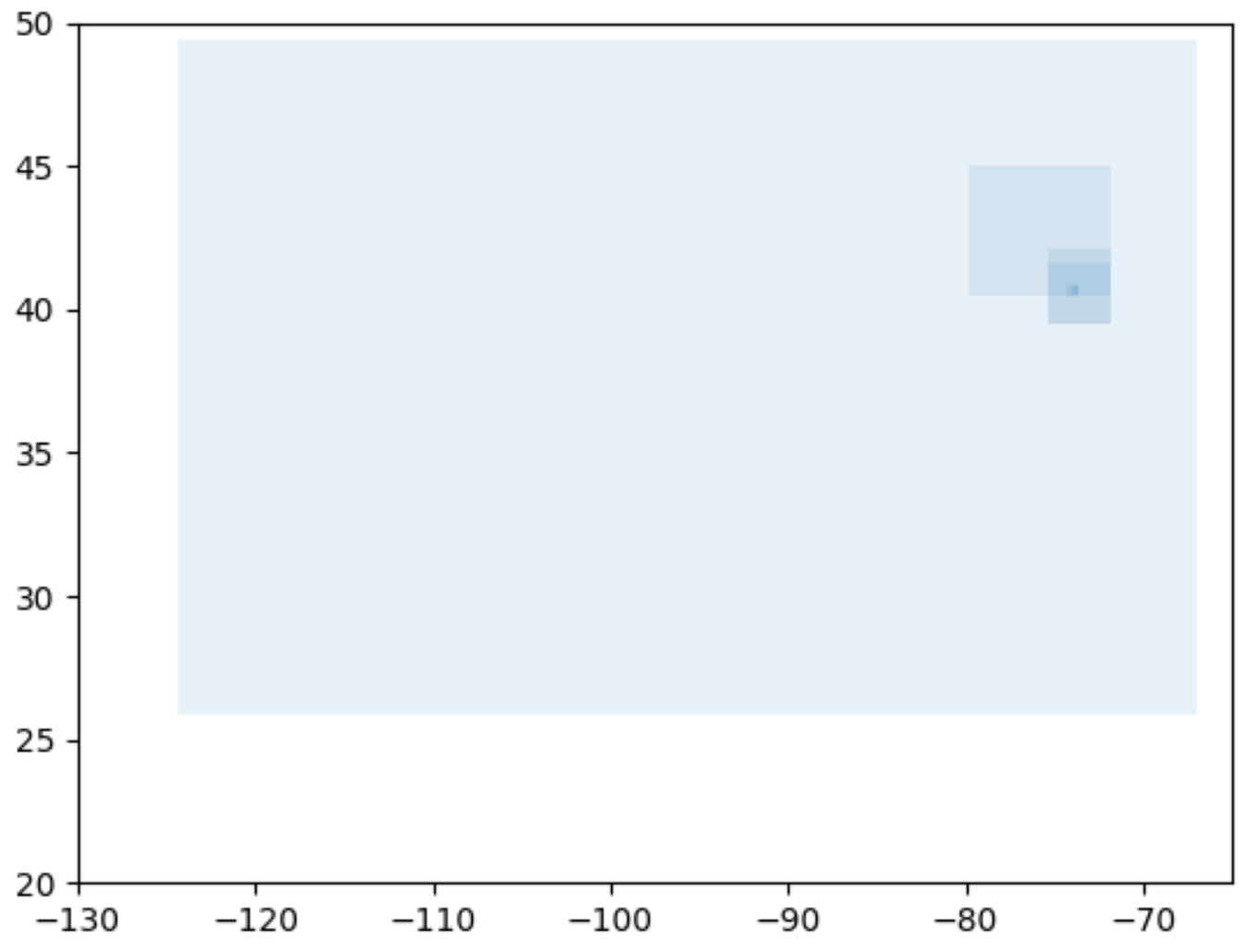

google_reverse_geocode.py

Google Reverse Geocoding



Google Reverse Geocoding

```
impo
impo
impo

reve

plt.
plt.
plt.
curr

for

plt.
```




```
))
```

plot_rectangles.py

Google Places

https://developers.google.com/places/

- Provides information about specific venues and locations
- .place(place_id) - Information about a specific place_id
- .places(query, location=None, radius=None, page_token=None) - query google maps for places within a certain radius of a given location
    - location - lat/long coordinates
    - radius - distance in meters
    - page_token - the token to paginate through the results
- Other options are also available to define price range, result language, type of result, etc…

```
['html_attributions',
 'results',
 'status',
 'next_page_token']
```

Google Places

https://developers.google.com/places/

- Provides information about specific venues and locations
- .place(place_id) - Information about a specific place_id
- .places(query, location=None, radius=None, page_token=None) - query google maps for places within a certain radius of a given location
    - location - lat/long coordinates
    - radius - distance in meters
    - page_token - the token to paginate through the results
- Other options are also available to define price range, result language, type of result, etc…

```
['html_attributions',
 'results',
 'status',
 'next_page_token']
```

Google Places

https://developers.google.com/places/

- Provides information about specific venues and locations
- .place(place_id) - Information about a specific place_id
- .places(query, location=None, radius=None, page_token=None) - query google maps for places within a certain radius of a given location
    - location - lat/long coordinates
    - radius - distance in meters
    - page_token - the token to paginate through the results
- Other options are also available to define price range, result language, type of result, etc…

```
['html_attributions',
 'results',
 'status',
 'next_page_token']
```

Google Places

https://developers.google.com/places/

```
['types',
 'id',
 'place_id',
 'address_components',
 'international_phone_number',
 'name',
 'reviews',
 'adr_address',
'vicinity',
'url',
'utc_offset',
'website',
'geometry',
'reference',
'rating',
'scope',
'formatted_address',
'photos',
'icon',
'formatted_phone_number']
```

Google Places

https://developers.google.com/places/

```
['types',                          ['airport', 'point_of_interest', 'establishment']
 'id',
 'place_id',
 'address_components',
 'international_phone_number',
 'name',                           'John F. Kennedy International Airport'
 'reviews',
 'adr_address',
 'vicinity',
 'url',
 'utc_offset',
 'website',
 'geometry',                      {'location': {'lat': 40.64131109999999, 'lng': -73.77813909999999},
 'reference',                      'viewport': {'northeast': {'lat': 40.67117114999999, 'lng': -73.76339675},
 'rating',                                      'southwest': {'lat': 40.63135774999999,
 'scope',                                                     'lng': -73.82236615000001}}}
 'formatted_address',
 'photos',
 'icon',
 'formatted_phone_number']
```

Challenge - Google Places

https://developers.google.com/places/

- Search for “pizza” within a 10km radius of

45.090219, 7.659144

- and print the formatted address of the second page of results.

```python
import googlemaps
from google_accounts import accounts
import time

app = accounts["torino"]
gmaps = googlemaps.Client(key=app["api_key"])

geocode_result = gmaps.places("pizza", (45.090219, 7.659144), 10000)
result = geocode_result["results"][0]
print(result["formatted_address"])

time.sleep(1)
page_token = geocode_result["next_page_token"]
geocode_result = gmaps.places(None, page_token=page_token)
result = geocode_result["results"][0]

print(result["formatted_address"])
```

google_place_search.py

Challenge - Google Places

https://developers.google.com/places/

```
{'formatted_address': 'Corso Peschiera, 312/A, 10139 Torino, Italy',
 'geometry': {'location': {'lat': 45.07176469999999, 'lng': 7.6317175},
              'viewport': {'northeast': {'lat': 45.07308433029149,
                                         'lng': 7.633044980291503},
                           'southwest': {'lat': 45.0703863697085,
                                         'lng': 7.630347019708498}}},
 'icon': 'https://maps.gstatic.com/mapfiles/place_api/icons/generic_business-71.png',
 'id': '644f3c75d97ae5ec66ffca7a1d2a34a5a98abad4',
 'name': 'Pizza Loca',
 'opening_hours': {'open_now': True, 'weekday_text': []},
 'place_id': 'ChIJnZx7tJJsiEcR4bCZfBz3VSs',
 'rating': 5,
 'reference': 'CmRRAAAAFdoEAMs_mvHc7wP6-2pJ_Qzs9Yw4u7aMi51cyk05rfITav2I-
r6gK_pDUAsfjxSJAf0YvIL0xaPZ7R_1XiFLJNyNiQ2TArhZMumL7J3B55CroVEQdNnbcvHwsJmgT53DEhDkbpkUbiFn4Ntnj3G
3K_tgGhQNSiQrCbG8gAM1ZQii-zshdQgmbw',
 'types': ['meal_delivery',
           'meal_takeaway',
           'restaurant',
           'food',
           'point_of_interest',
           'establishment']}
```

@bgoncalves

Challenge - Google Places

https://developers.google.com/places/

```
{'formatted_address': 'Corso Peschiera, 312/A, 10139 Torino, Italy',
 'geometry': {'location': {'lat': 45.07176469999999, 'lng': 7.6317175},
              'viewport': {'northeast': {'lat': 45.07308433029149,
                                         'lng': 7.633044980291503},
                           'southwest': {'lat': 45.0703863697085,
                                         'lng': 7.630347019708498}}},
 'icon': 'https://maps.gstatic.com/mapfiles/place_api/icons/generic_business-71.png',
 'id': '644f3c75d97ae5ec66ffca7a1d2a34a5a98abad4',
 'name': 'Pizza Loca',
 'opening_hours': {'open_now': True, 'weekday_text': []},
 'place_id': 'ChIJnZx7tJJsiEcR4bCZfBz3VSs',
 'rating': 5,
 'reference': 'CmRRAAAAFdoEAMs_mvHc7wP6-2pJ_Qzs9Yw4u7aMi51cyk05rfITav2I-
r6gK_pDUAsfjxSJAf0YvIL0xaPZ7R_1XiFLJNyNiQ2TArhZMumL7J3B55CroVEQdNnbcvHwsJmgT53DEhDkbpkUbiFn4Ntnj3G
3K_tgGhQNSiQrCbG8gAM1ZQii-zshdQgmbw',
 'types': ['meal_delivery',
           'meal_takeaway',
           'restaurant',
           'food',
           'point_of_interest',
           'establishment']}
```

@bgoncalves

Google Directions

https://developers.google.com/maps/documentation/directions/

- Provides step by step directions from point A to point B.
- Directions are always for the shortest path
- .directions((start_lat, start_lon), (end_lat, end_lon)) - Directions between two sets of GPS coordinates
- .directions("start_address", "end_address") - Geolocate each address and provide directions from start_address to end_address.
- Each step of the journey comes with start_location and end_location GPS coordinates as well as distance, duration and html_instructions.

```
{'distance': {'text': '0.5 km', 'value': 464},
 'duration': {'text': '2 mins', 'value': 98},
 'end_location': {'lat': 45.5014209, 'lng': 9.1882349},
 'html_instructions': 'Continue onto <b>Via Valassina</b>',
 'polyline': {'points':
'{eutGagaw@YGSG_AWq@Sc@MkFaBeDeAgDiAEA[M'},
 'start_location': {'lat': 45.4974178, 'lng': 9.1865724},
 'travel mode': 'DRIVING'}
```

Challenge - Google Directions

https://developers.google.com/maps/documentation/directions/

- Obtain the driving directions from coordinates:

45.067450, 7.685880

- to coordina
- and print ou

```python
import googlemaps
from google_accounts import accounts

app = accounts["torino"]
gmaps = googlemaps.Client(key=app["api_key"])

directions = gmaps.directions((45.067450, 7.685880), (45.485895, 9.204283))

steps = directions[0]["legs"][0]["steps"]

for step in steps:
    print(step["html_instructions"])
```

google_directions.py

Google Directions

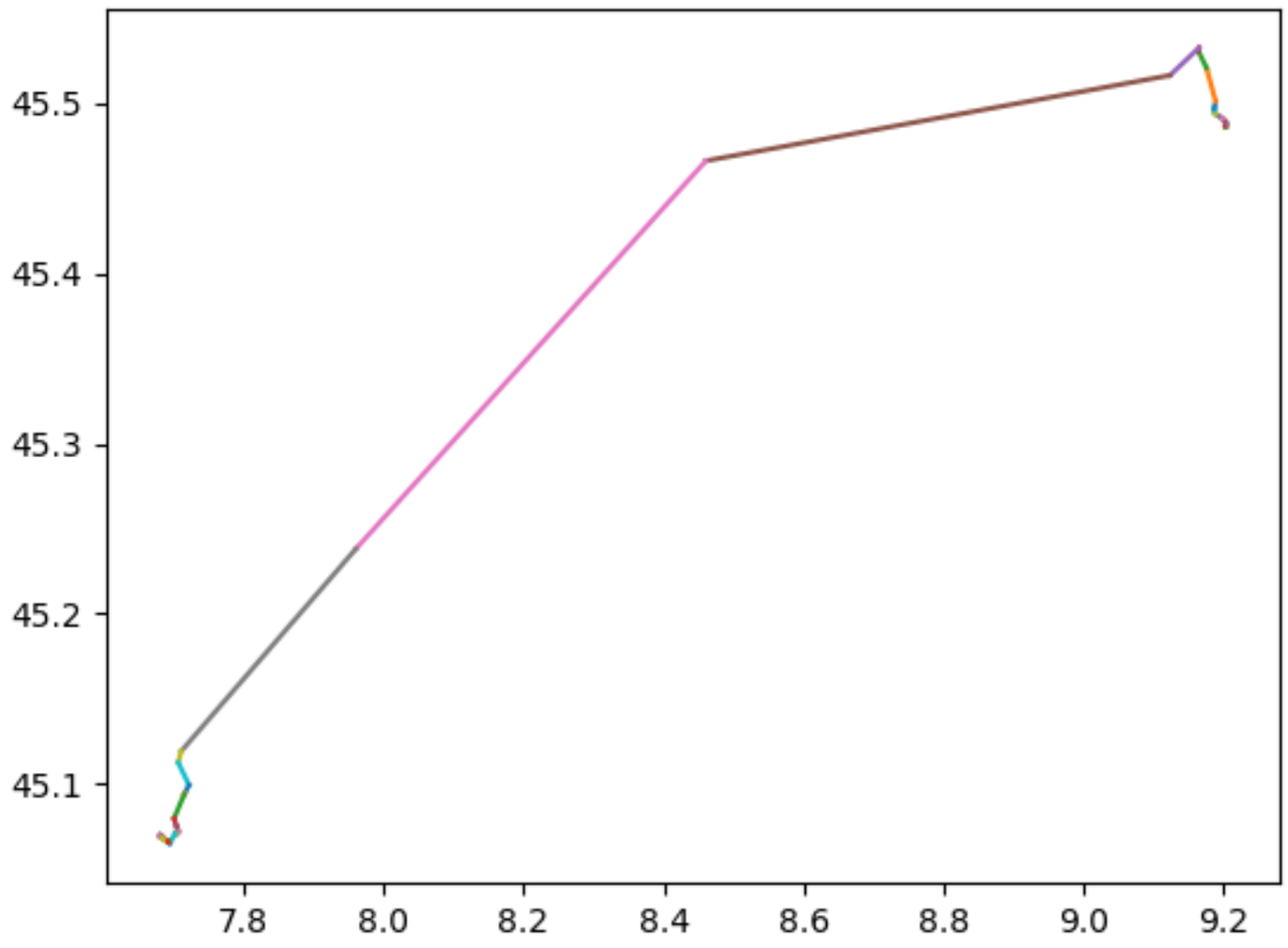
https://developers.google.com/maps/documentation/directions/

```python
import googlemaps
from google_accounts import accounts
import matplotlib.pyplot as plt

app = accounts["torino"]
gmaps = googlemaps.Client(key=app["api_key"])

directions = gmaps.directions((45.485895, 9.204283),(45.067450, 7.685880))

steps = directions[0]["legs"][0]["steps"]

for step in steps:
    start = (step["start_location"]["lng"], step["start_location"]["lat"])
    stop = (step["end_location"]["lng"], step["end_location"]["lat"])

    plt.plot([start[0], stop[0]], [start[1], stop[1]])

plt.savefig('MiTo.png')
```


plot_google_directions.py

Google Directions

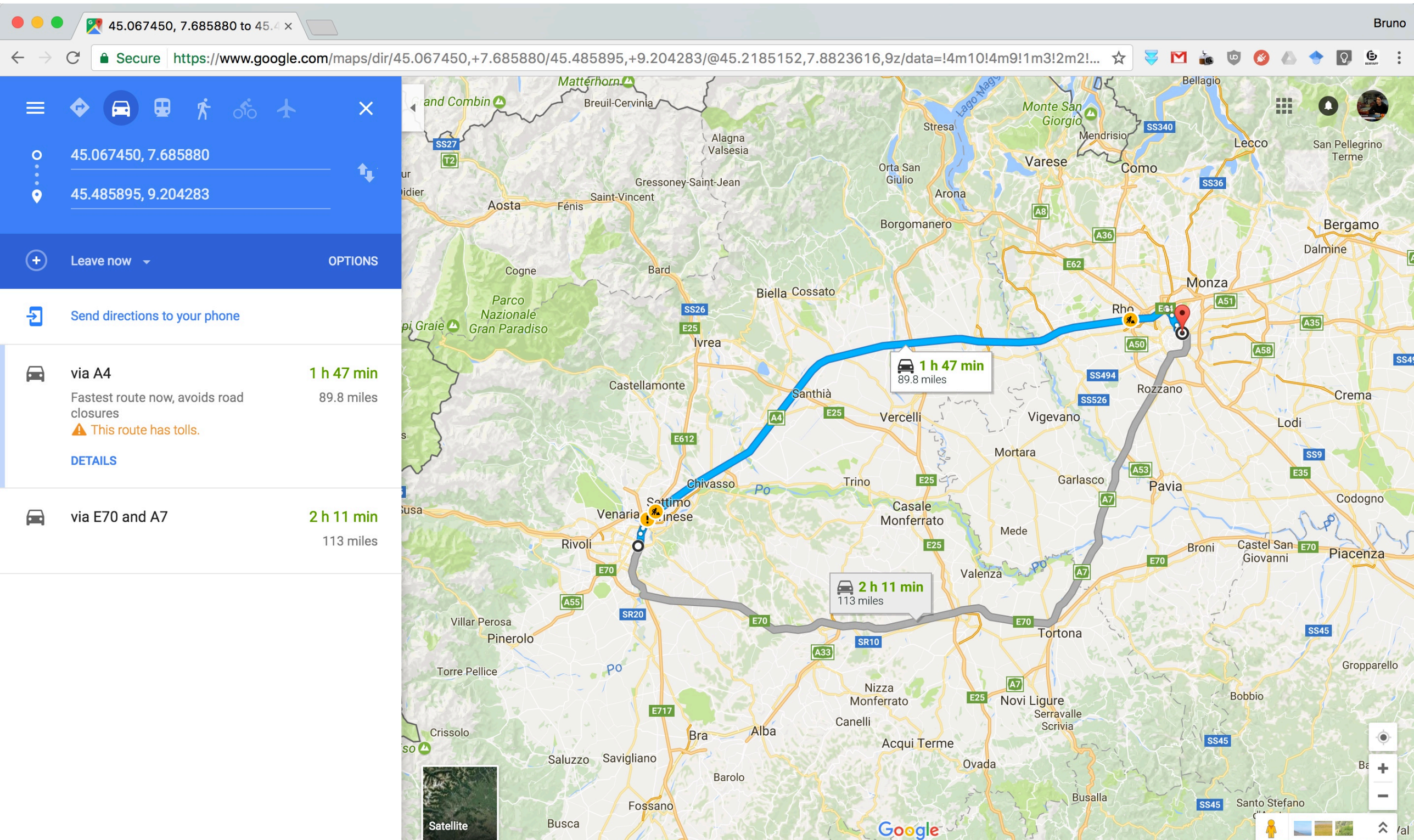
https://developers.google.com/maps/documentation/directions/



@bgoncalves

Google Directions

https://developers.google.com/maps/documentation/directions/



Google Distance Matrix

https://developers.google.com/maps/documentation/distance-matrix/

- Returns the travel time and distance for all pairs of origins and destinations
- Results are returned in rows, each row containing one origin paired with each destination.
- .distance_matrix(origins, destinations) - Both origins and destinations can be listed as either lat/long pairs or addresses
- Options for **mode** of transportation, **departure** or **arrival** time, etc…

Challenge - Google Distance Matrix

https://developers.google.com/maps/documentation/distance-matrix/

- Obtain the distance and time matrix from these origins:

(45.485895, 9.204283), (41.912199, 12.499857)

- to these destinations:

(45.067450, 7.685880), (43.775475, 11.257102)

- And print out the resulting object

Challenge - Google Distance Matrix

https://developers.google.com/maps/documentation/distance-matrix/

```python
import googlemaps
from google_accounts import accounts
from pprint import pprint

app = accounts["torino"]
gmaps = googlemaps.Client(key=app["api_key"])

origins = [(45.485895, 9.204283), (41.912199, 12.499857)]
destinations = [(45.067450, 7.685880), (43.775475, 11.257102)]

matrix = gmaps.distance_matrix(origins, destinations)

pprint(matrix)
```


google_distance.py

Google Distance Matrix

https://developers.google.com/maps/documentation/distance-matrix/


```
{'destination_addresses': ['Via Maria Vittoria, 12, 10123 Torino, Italy',
                           'Via Ricasoli, 26-30, 50122 Firenze, Italy'],
 'origin_addresses': ['Piazza Quattro Novembre, 581, 20124 Milano, Italy',
                      'Via Brescia, 23, 00198 Roma, Italy'],
 'rows': [{'elements': [{'distance': {'text': '147 km', 'value': 147321},
                         'duration': {'text': '1 hour 54 mins', 'value': 6868},
                         'status': 'OK'},
                        {'distance': {'text': '307 km', 'value': 306952},
                         'duration': {'text': '3 hours 23 mins','value': 12167},
                         'status': 'OK'}]},
          {'elements': [{'distance': {'text': '690 km', 'value': 689739},
                         'duration': {'text': '6 hours 49 mins','value': 24521},
                         'status': 'OK'},
                        {'distance': {'text': '276 km', 'value': 275684},
                         'duration': {'text': '3 hours 6 mins', 'value': 11163},
                         'status': 'OK'}]}],
 'status': 'OK'}
```


@bgoncalves

Google Distance Matrix

https://developers.google.com/maps/documentation/distance-matrix/


```
{'destination_addresses': ['Via Maria Vittoria, 12, 10123 Torino, Italy',
                           'Via Ricasoli, 26-30, 50122 Firenze, Italy'],
 'origin_addresses': ['Piazza Quattro Novembre, 581, 20124 Milano, Italy',
                      'Via Brescia, 23, 00198 Roma, Italy'],
 'rows': [{'elements': [{'distance': {'text': '147 km', 'value': 147321},
                         'duration': {'text': '1 hour 54 mins', 'value': 6868},
                         'status': 'OK'},
                        {'distance': {'text': '307 km', 'value': 306952},
                         'duration': {'text': '3 hours 23 mins','value': 12167},
                         'status': 'OK'}]},
          {'elements': [{'distance': {'text': '690 km', 'value': 689739},
                         'duration': {'text': '6 hours 49 mins','value': 24521},
                         'status': 'OK'},
                        {'distance': {'text': '276 km', 'value': 275684},
                         'duration': {'text': '3 hours 6 mins', 'value': 11163},
                         'status': 'OK'}]}],
 'status': 'OK'}
```


@bgoncalves