https://bmtgoncalves.github.io/EABDA17/

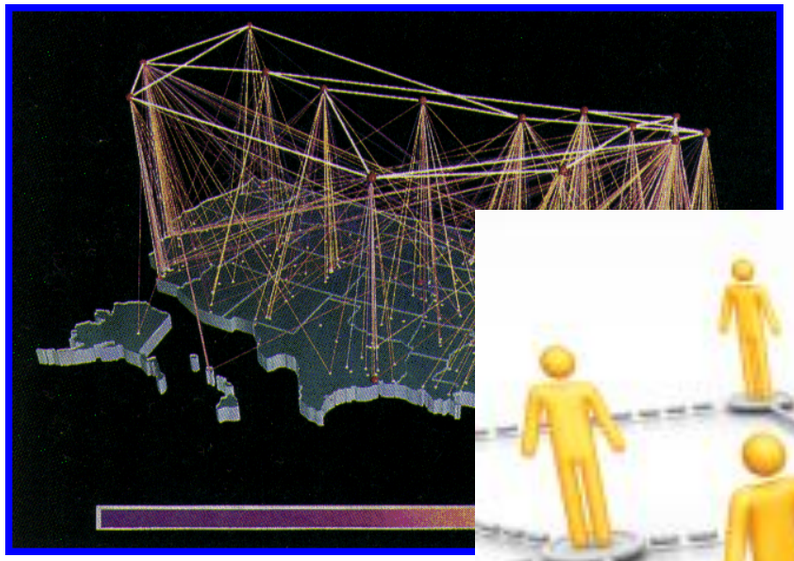
# Data Mining and Spatial Analysis

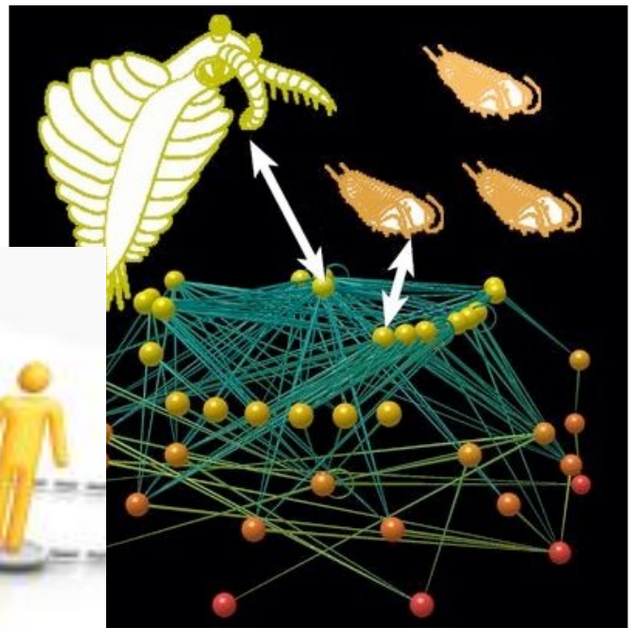
Bruno Gonçalves
*www.bgoncalves.com*

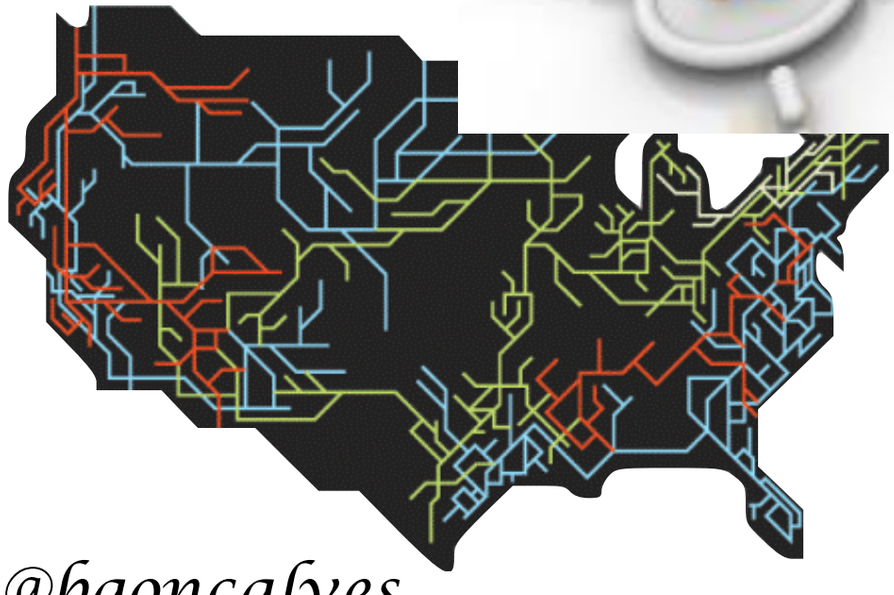
# Requirements

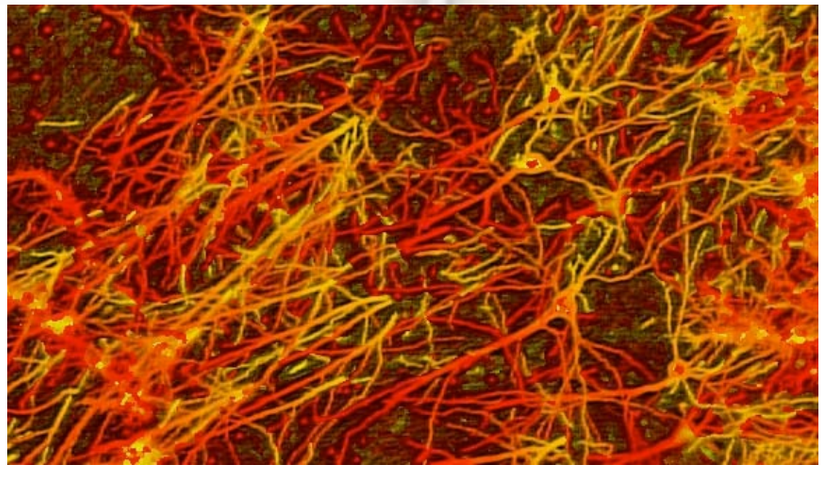
https://bmtgoncalves.github.io/EABDA17/

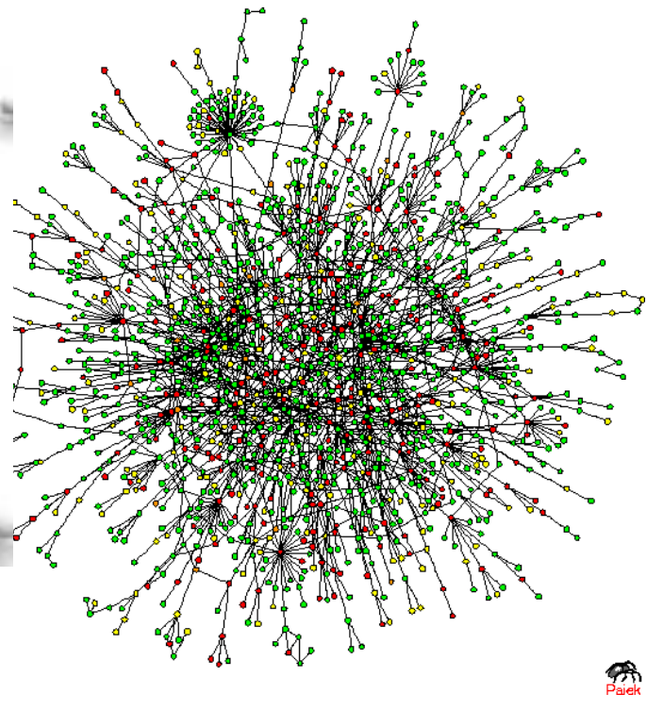




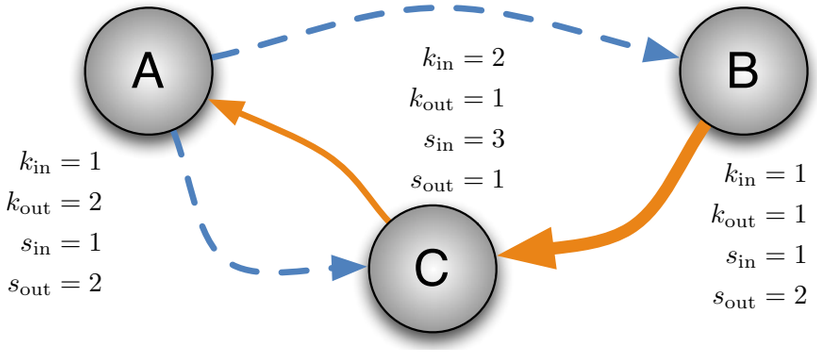


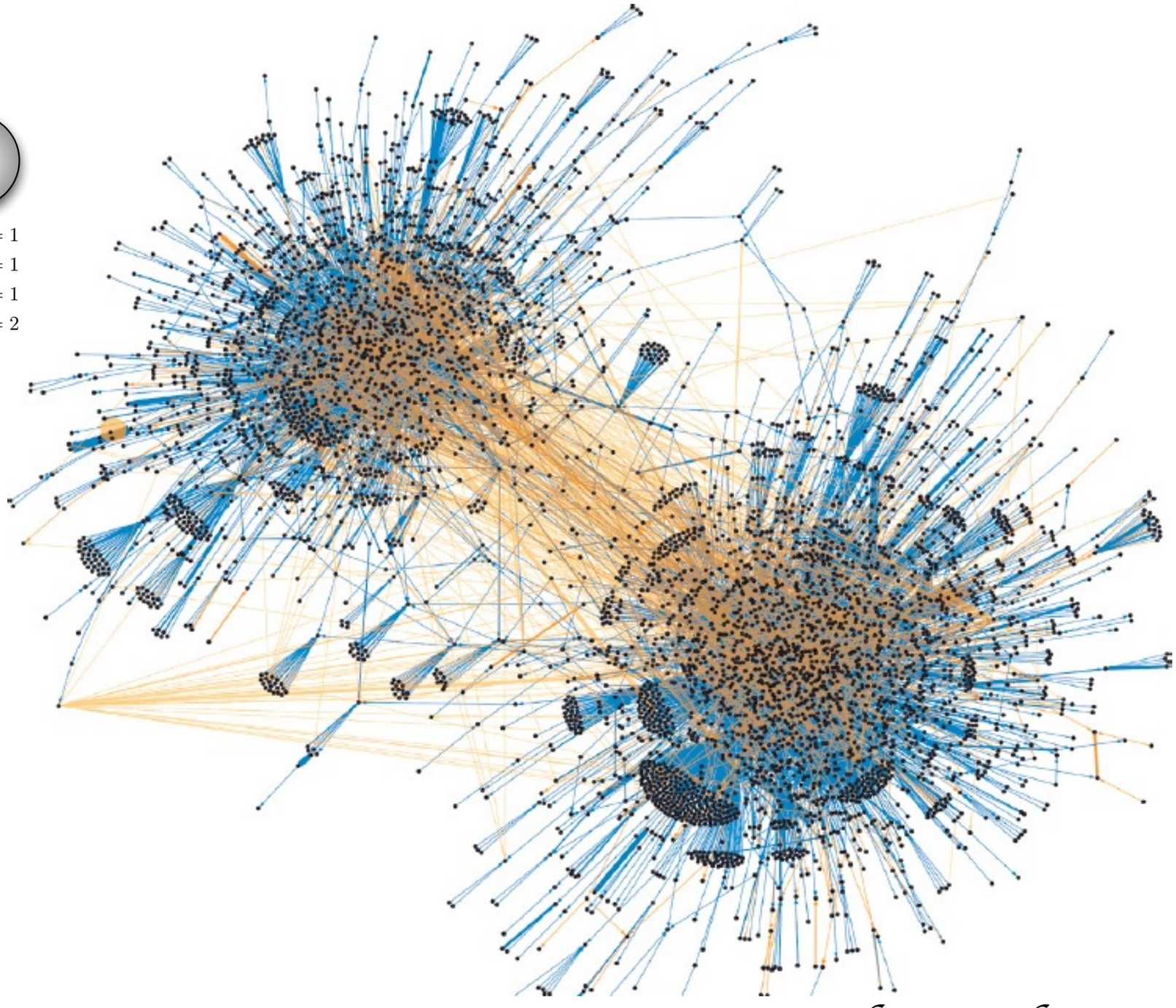





Palek

# Information Flow



# NetworkX

- High productivity software for complex networks
- Simple Python interface
- Four types of graphs supported:
    - Graph - UnDirected
    - DiGraph - Directed
    - 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

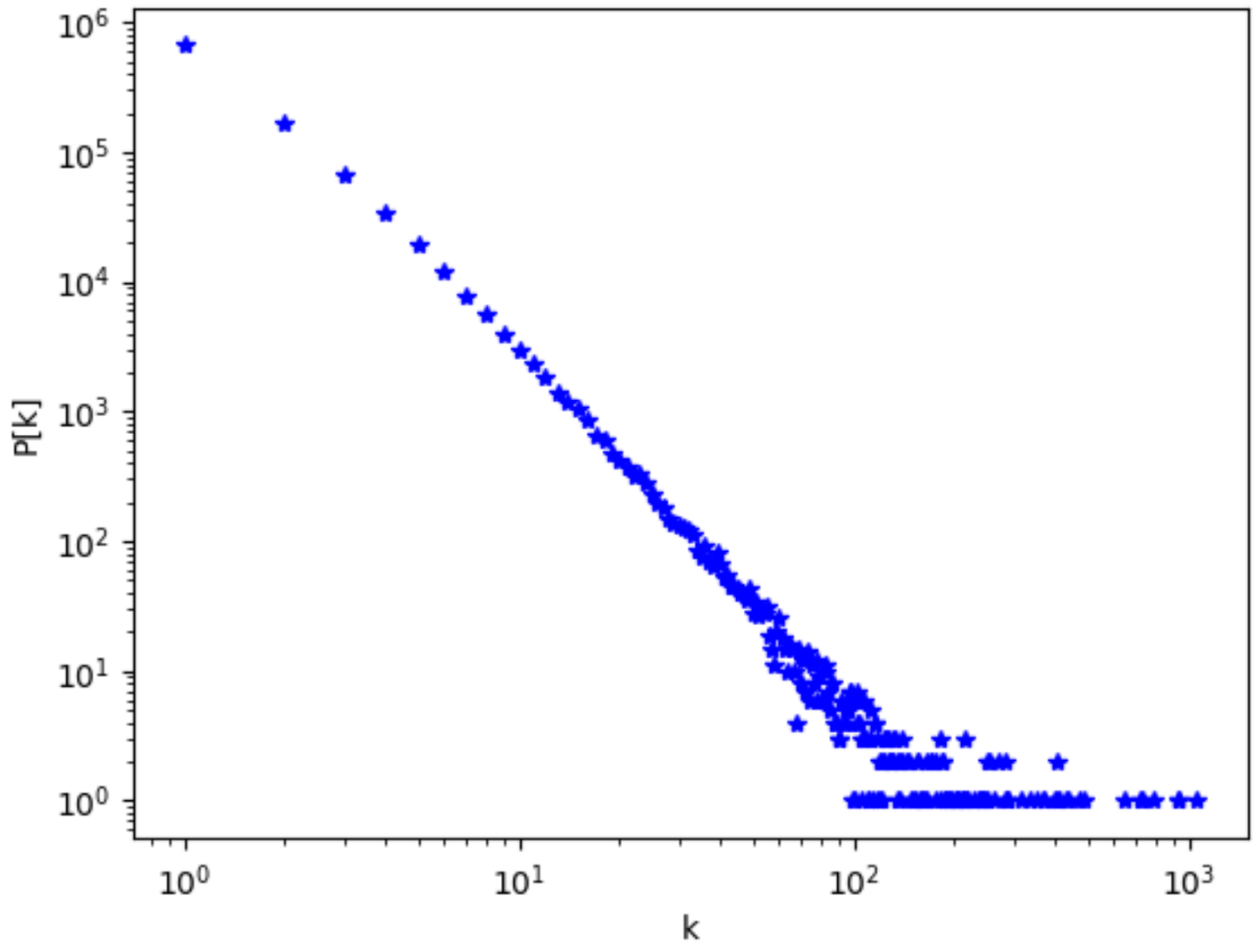
# NetworkX - Example

```
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

# NetworkX - Example



# 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

# Snowball Sampling

```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

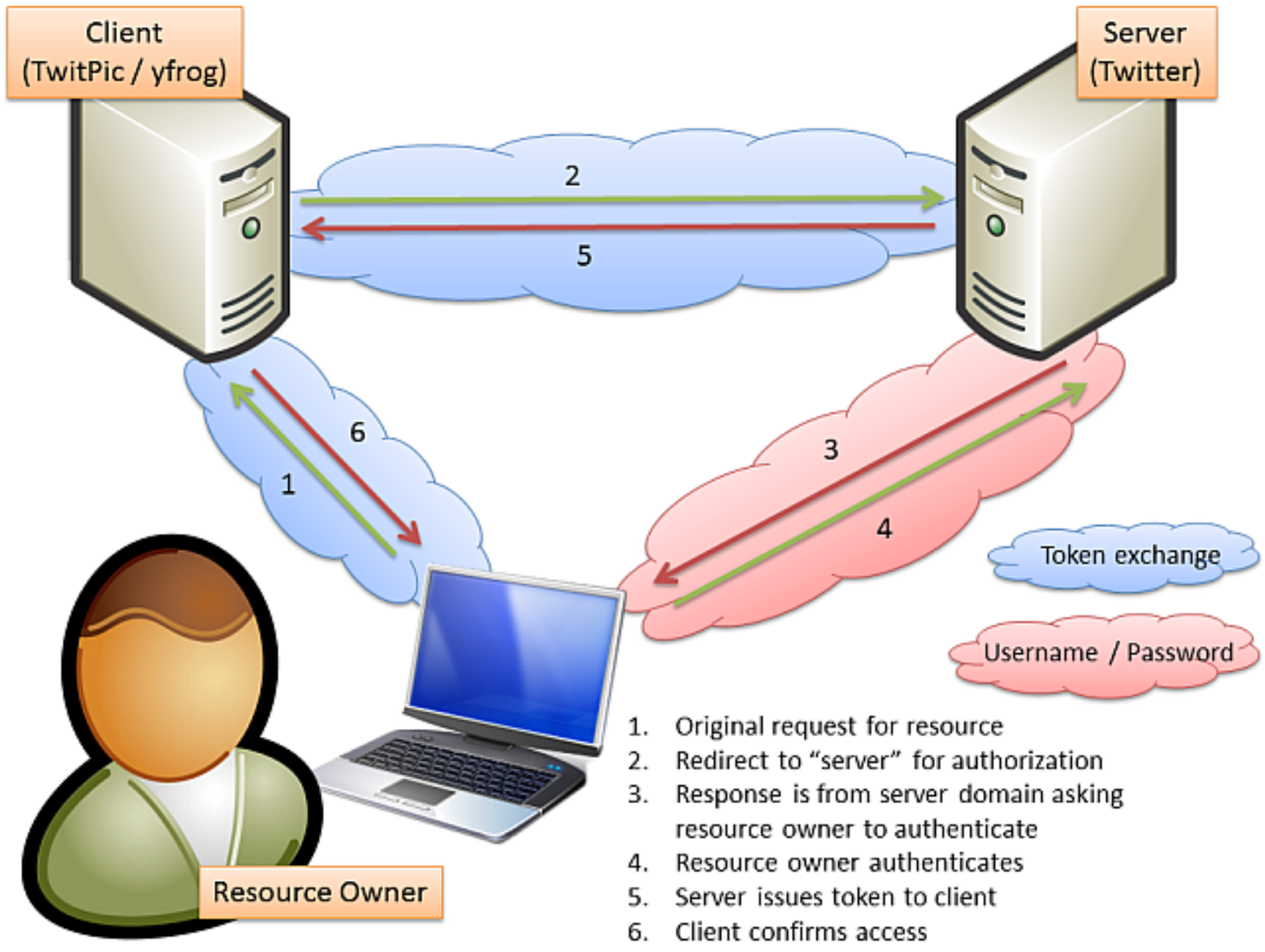
http://hueniverse.com/oauth/
https://tools.ietf.org/html/rfc5849

- "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



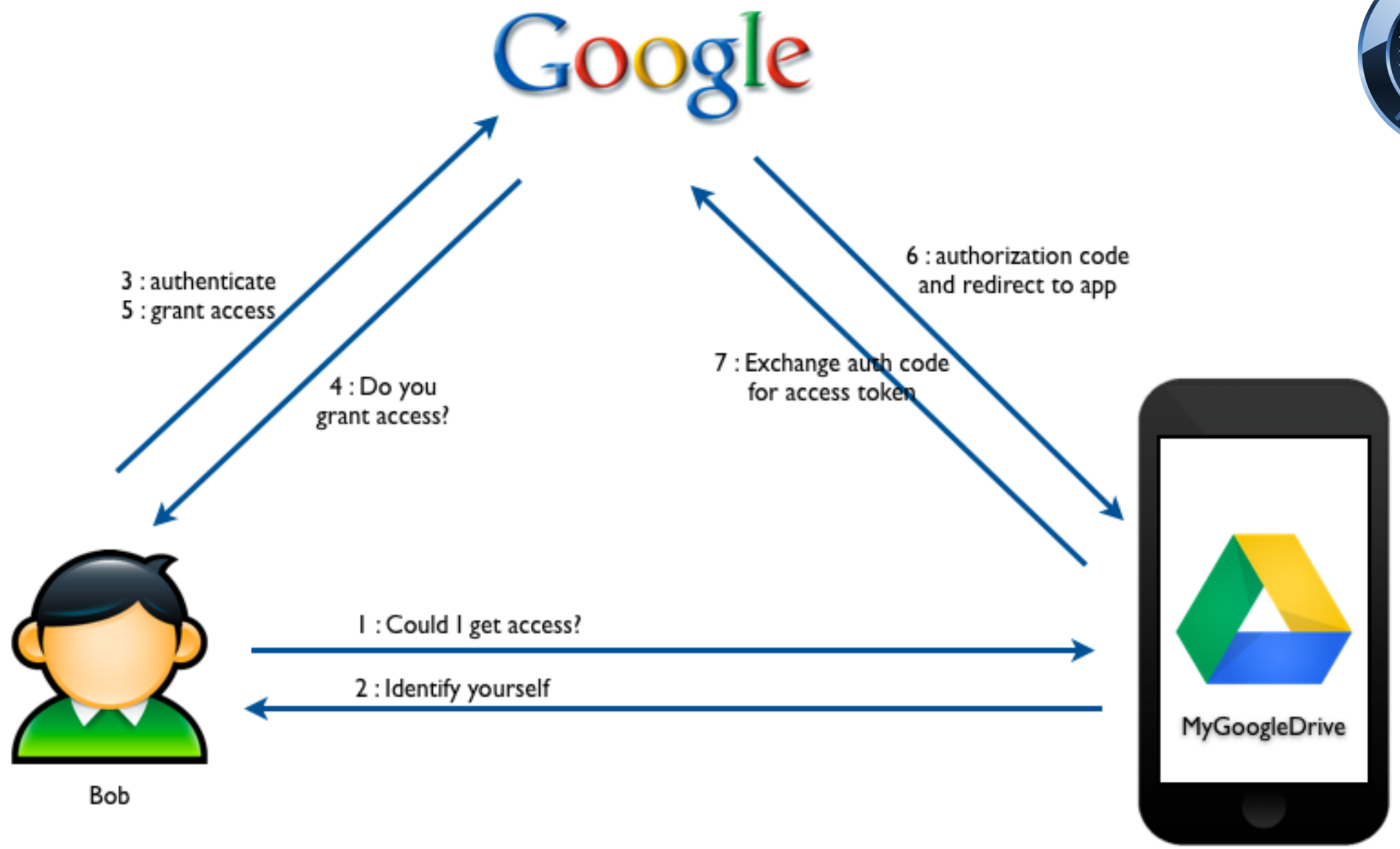
# OAuth 1

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



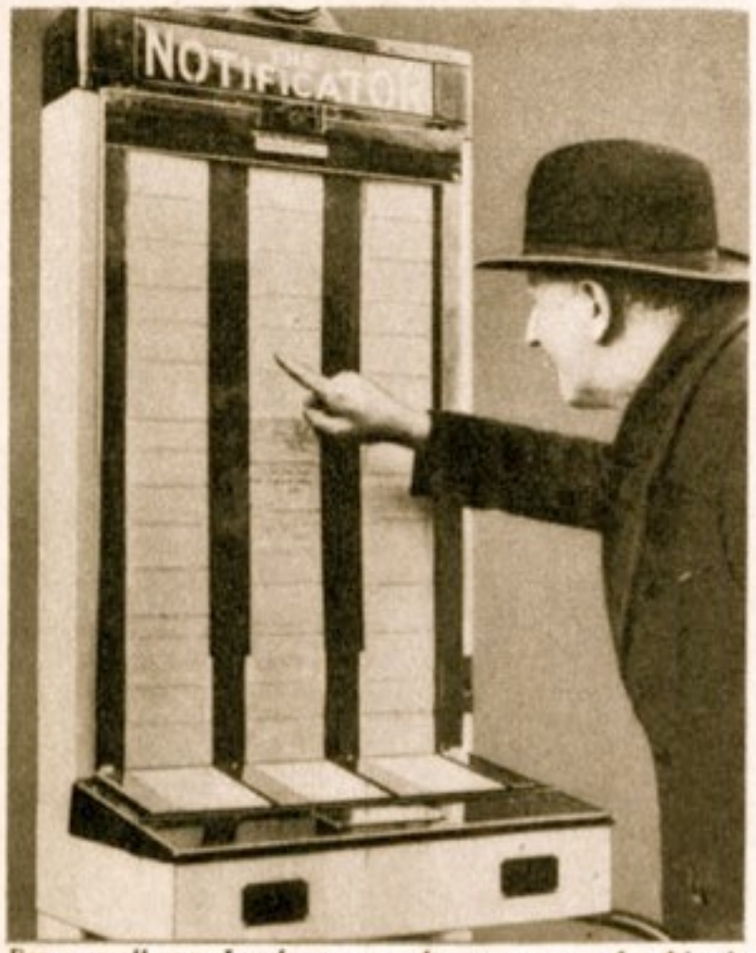
# OAuth 2

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

## 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"
```

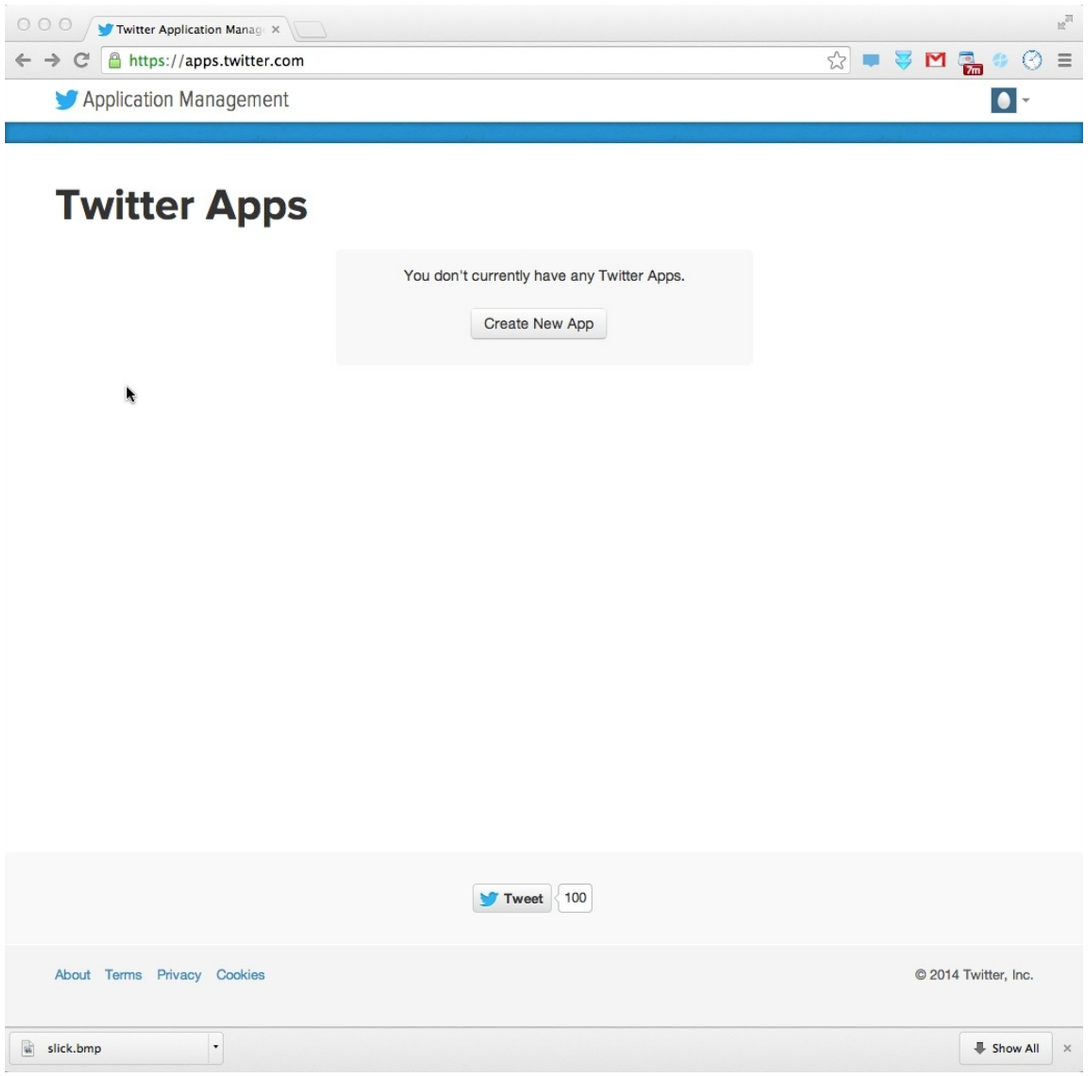
```
u'<a href="http://foursquare.com" rel="nofollow">
  foursquare</a>'
```

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

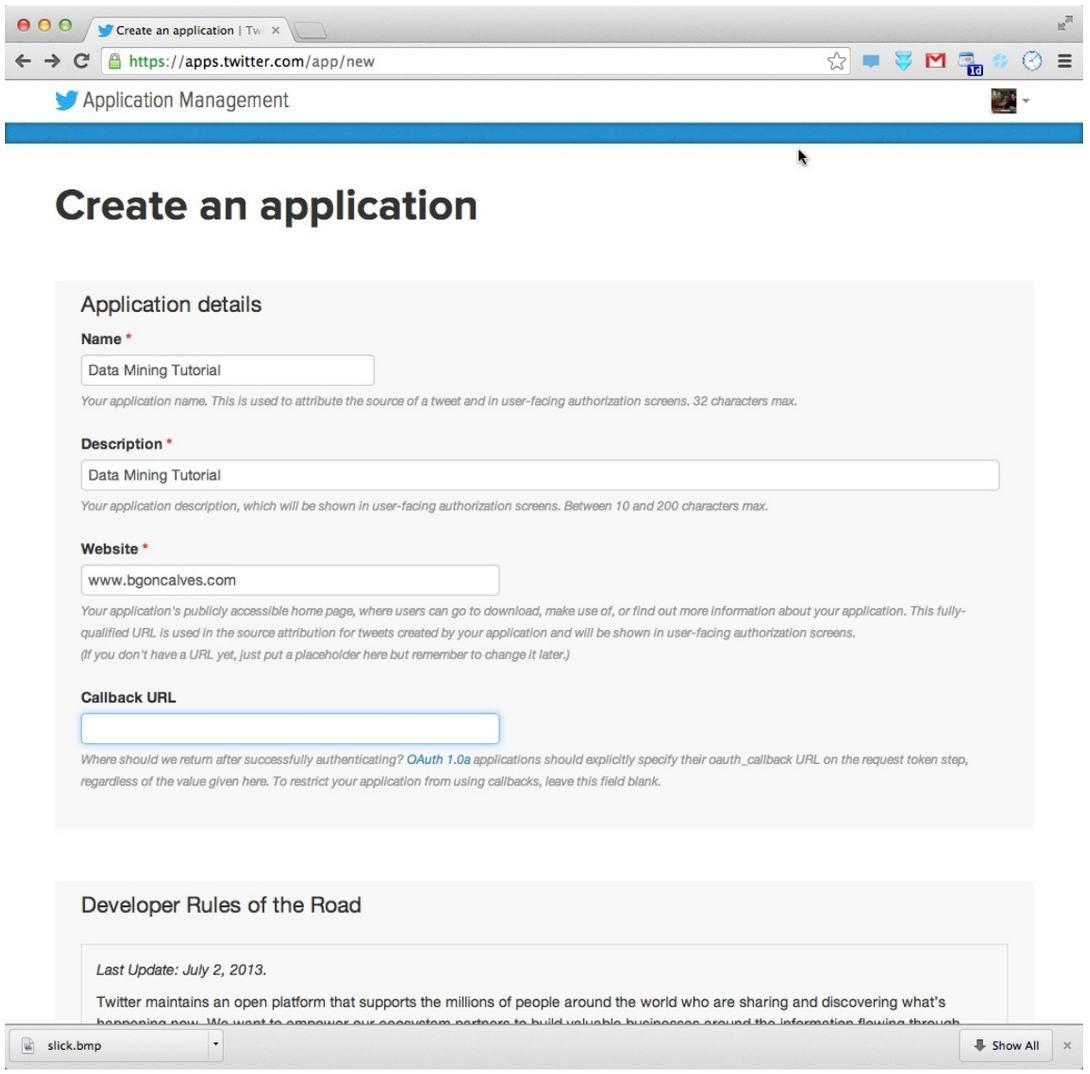
```
{u'display_url': u'4sq.com/1k5MeYF',
 u'expanded_url': u'http://4sq.com/1k5MeYF',
 u'indices': [70, 92],
 u'url': u'http://t.co/WirvdHwYMq'}
```

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

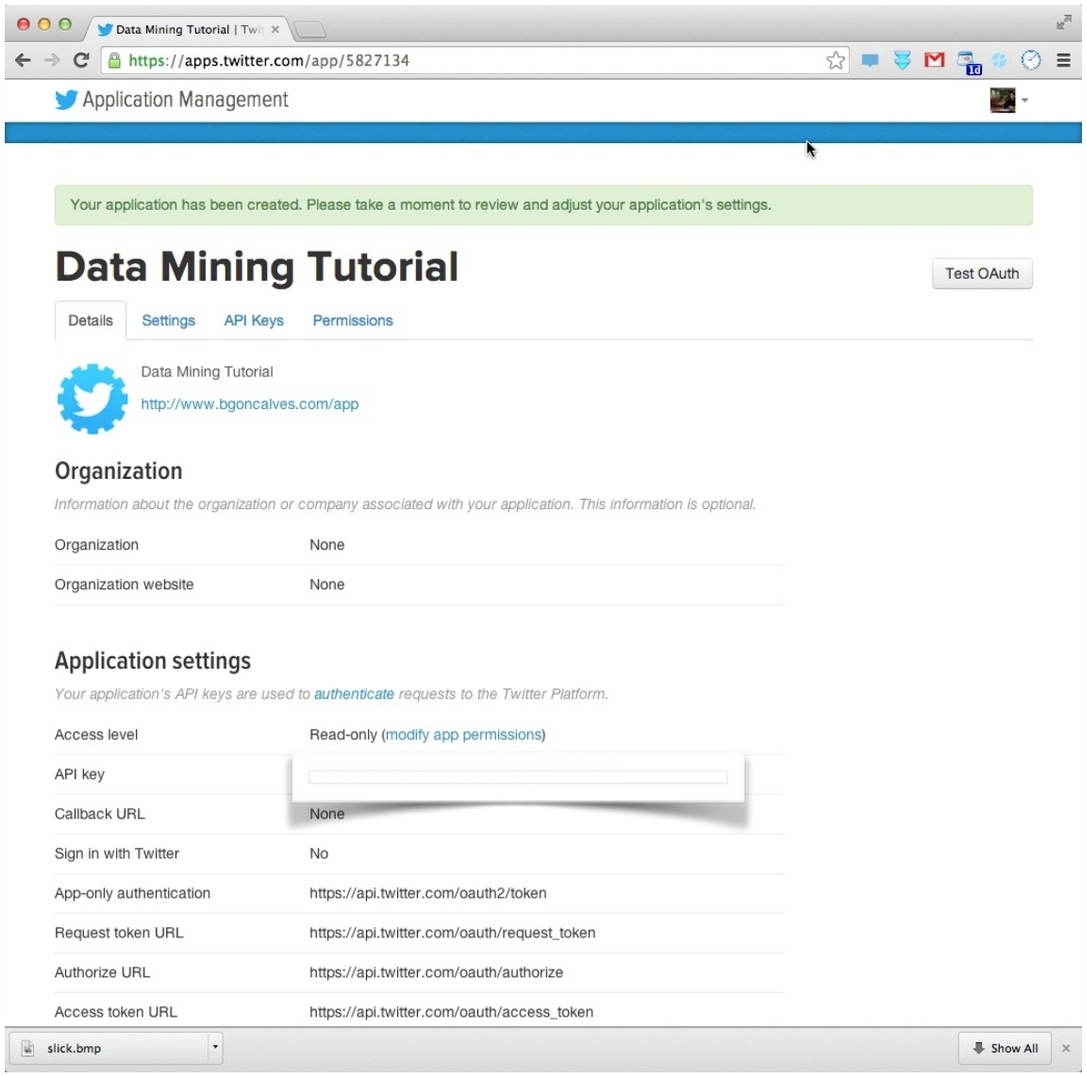
# Registering an Application



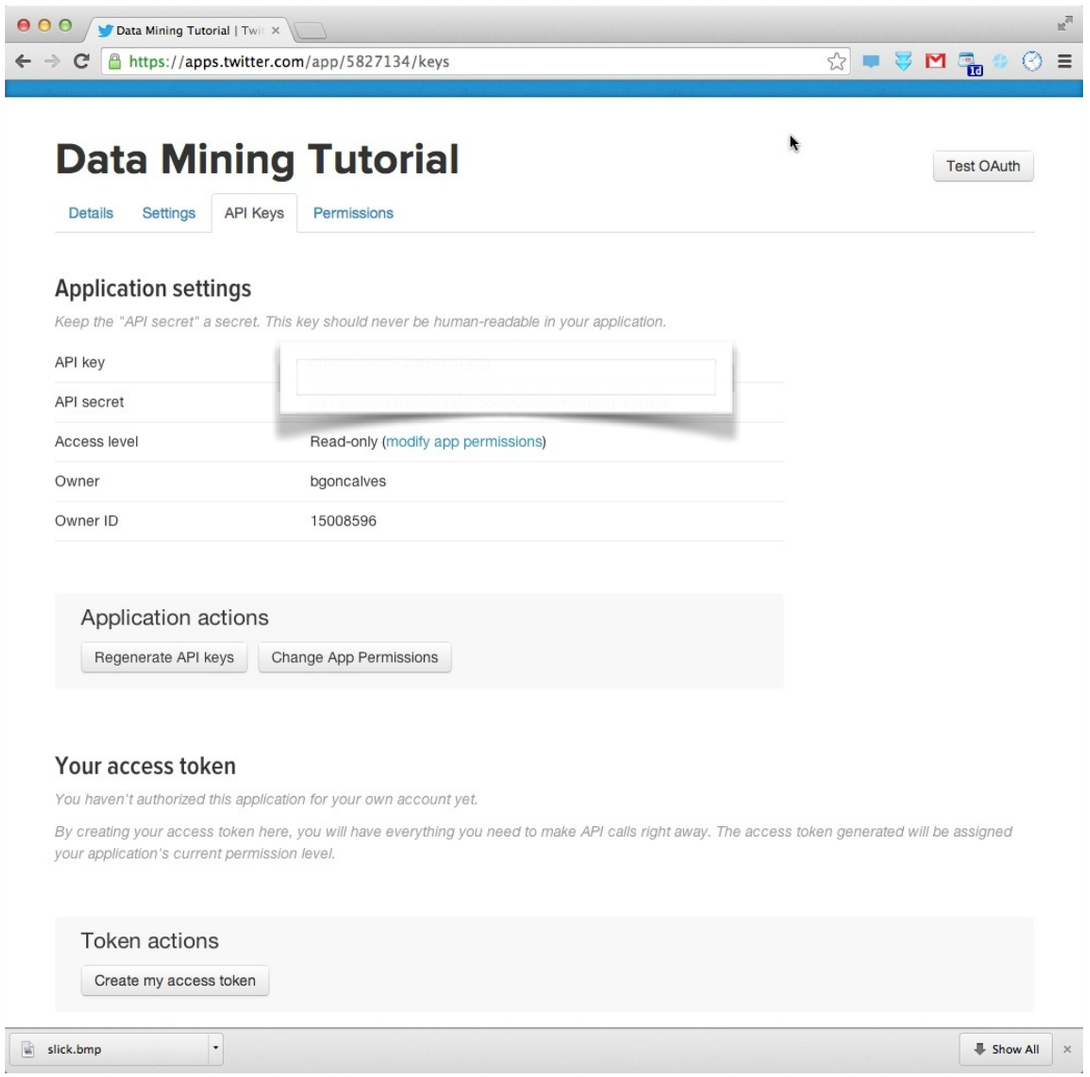
# Registering an Application



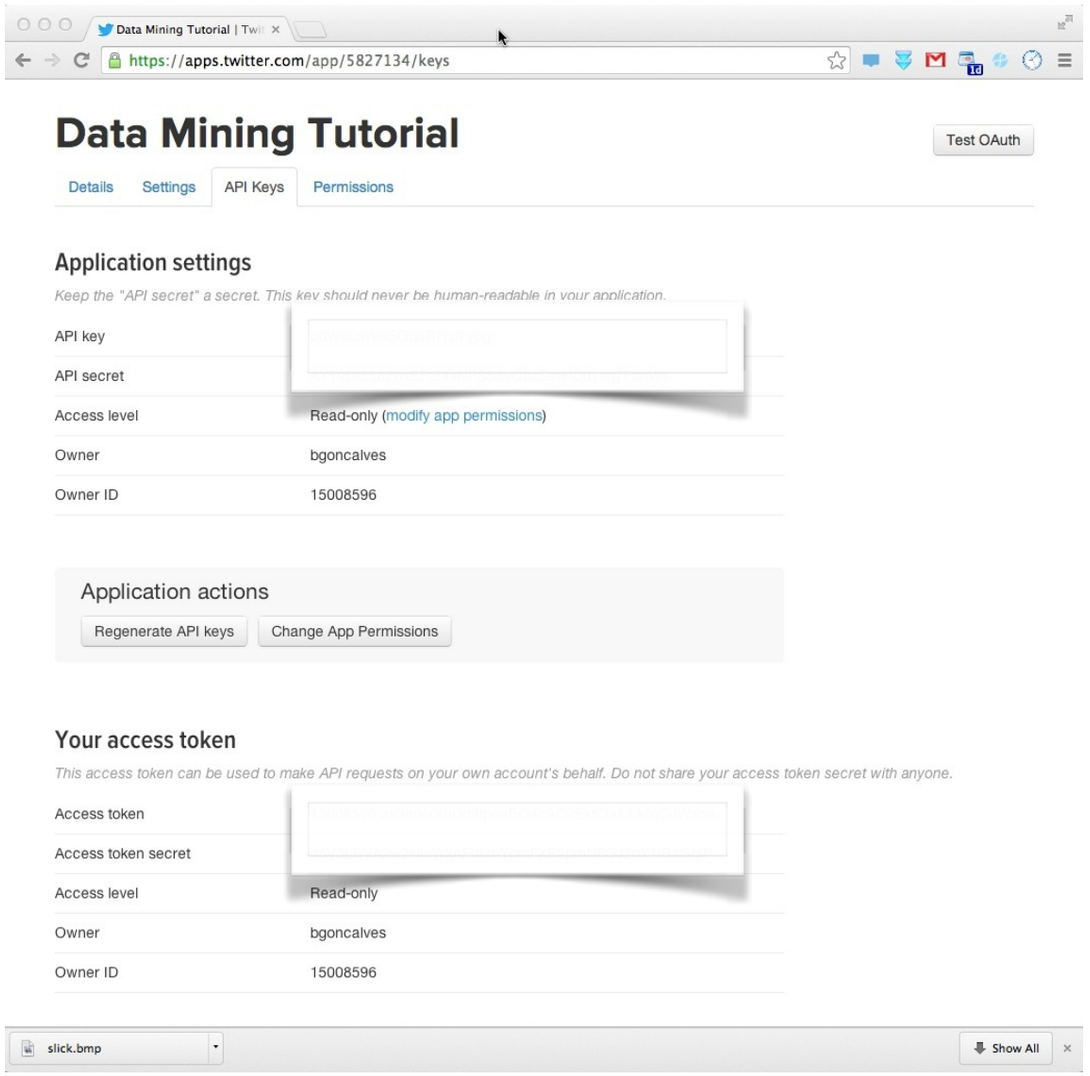
# Registering an Application



# 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

# Social Connections

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

```
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

```
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 = "bgoncalves"
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(user, "->", tweet["retweeted_status"]["user"]["id"])
    elif tweet["in_reply_to_user_id"]:
        print(tweet["in_reply_to_user_id"], "->", user)
```



twitter_interactions.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

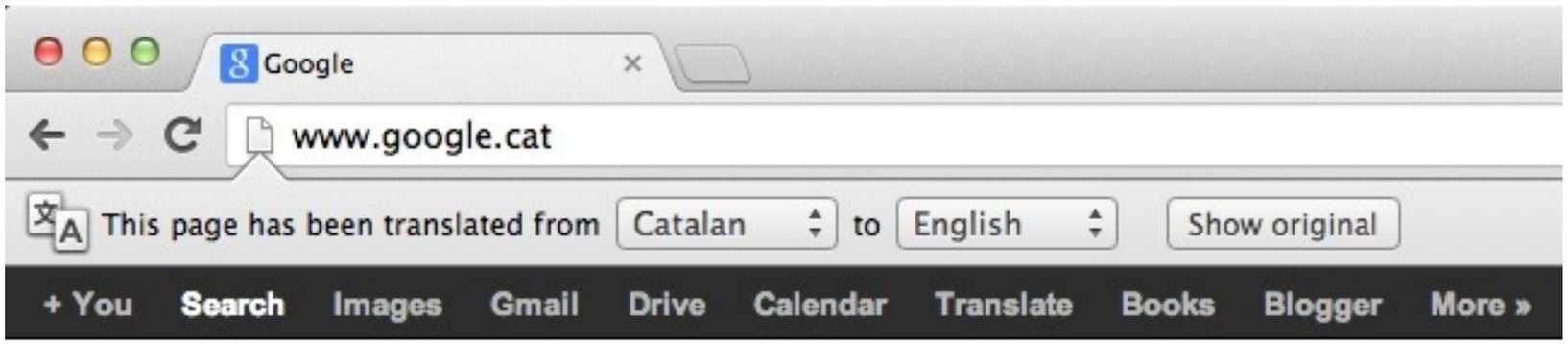
# Streaming Geocoded data

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

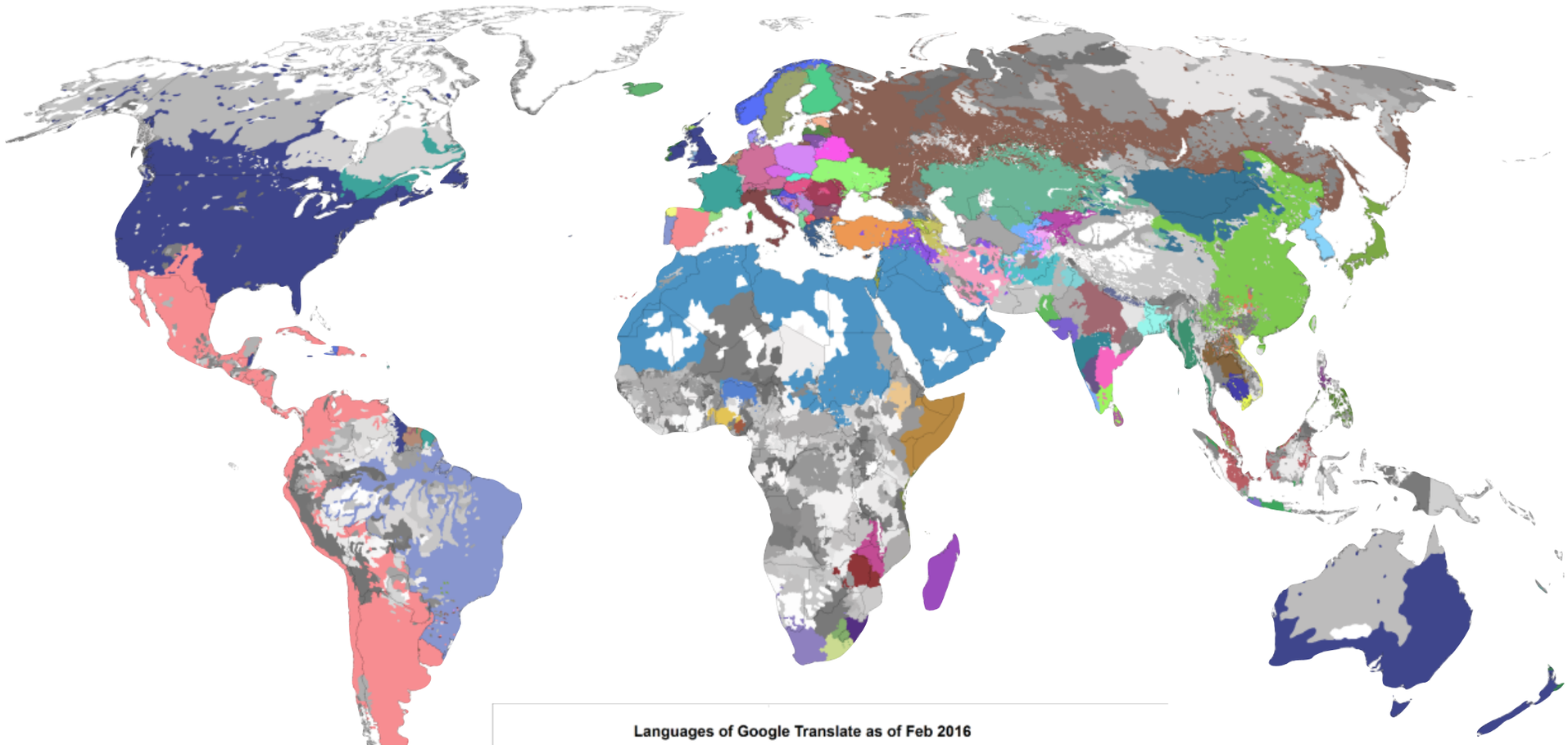
```
import twitter
from twitter_accounts import accounts
import sys
import gzip

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

# Chromium Compact Language Detector




# Chromium Compact Language Detector



# cld

https://github.com/mikemccand/chromium-compact-language-detector

- C package with C++ and Python bindings
- Particularly messy to install, but very fast and easy to use
- cld.detect(text) returns a list of possible languages and their likelihood
- Very conservative. If it thinks that it can't give a reliable answer, it will return "Unknown"
- Version 2 is better suited to longer pieces of text and it allows for the possibility of different sections being in different languages.

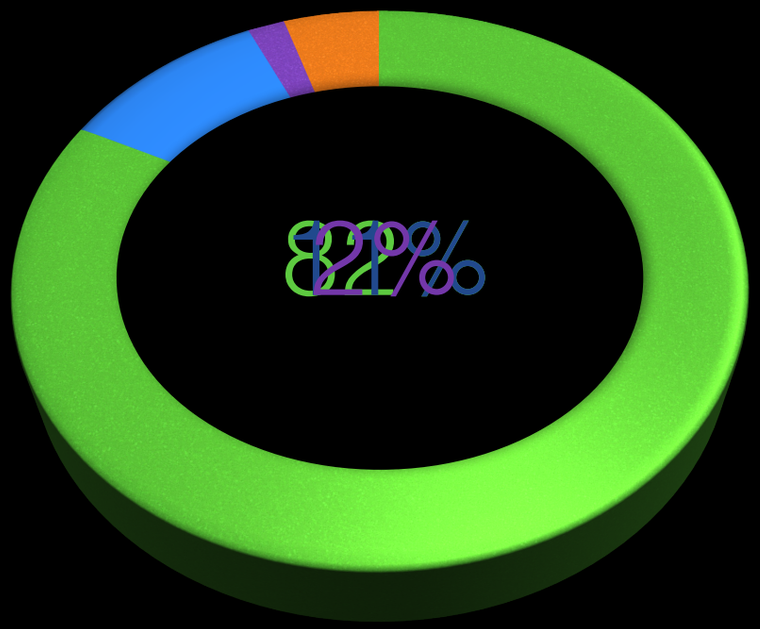
```
import cld

text_it = "Wales lancia la Wikipedia delle news. Contro il fake in campo anche Google"
text_en = "Cassini Spacecraft Re-Establishes Contact After 'Dive' Between Saturn And Its Rings"

lang_it = cld.detect(text_it)
lang_en = cld.detect(text_en)

print(text_it, "is in", lang_it)
print(text_en, "is in", lang_en)
```

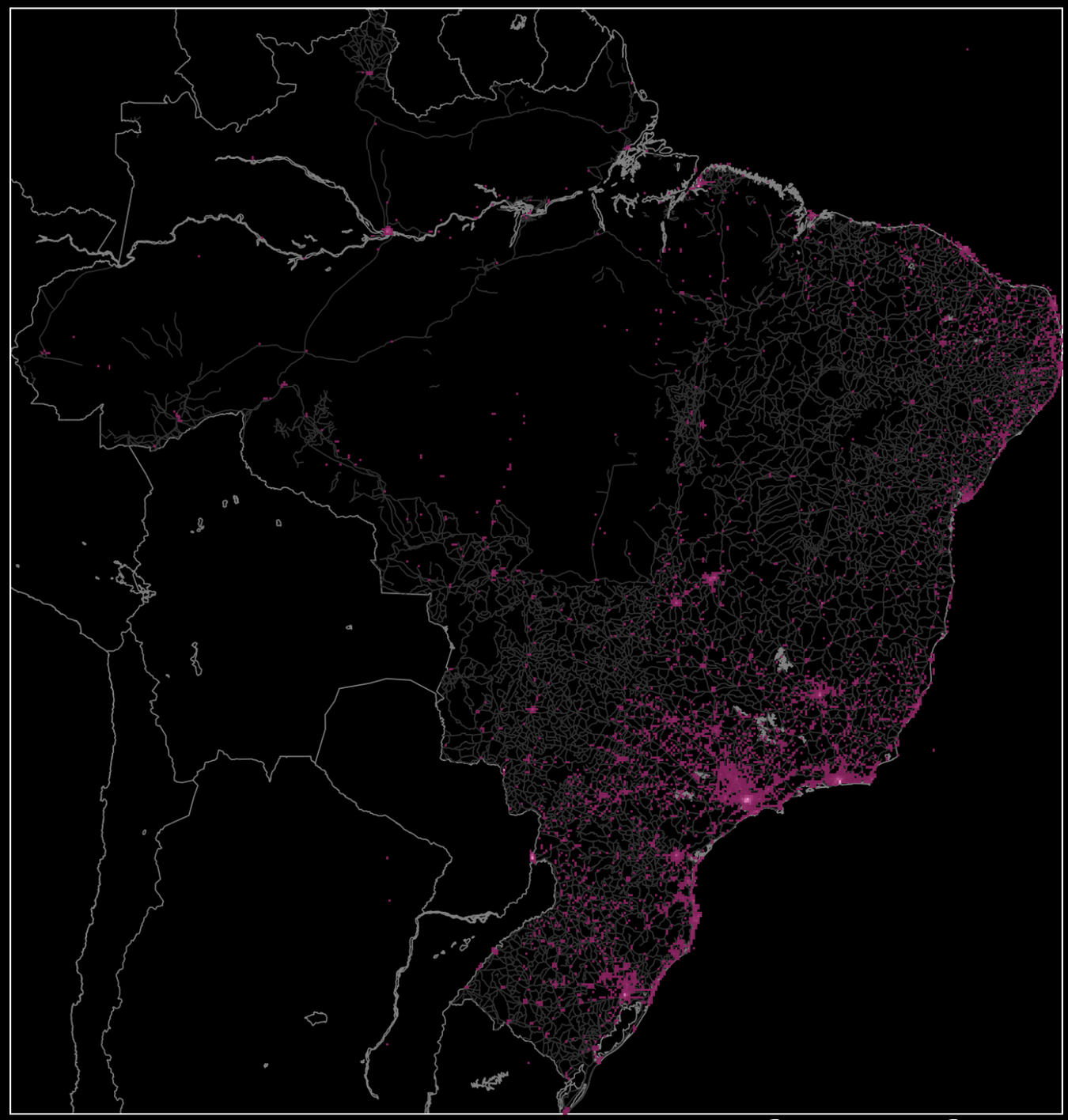


language_cld.py

# Signal By Language

- Portuguese
- English
- Spanish
- Other



# langid

https://github.com/saffsd/langid.py

- Entirely written in Python
- Supports 97 languages
- Has the option of retrieving the likelihood for every known language.
- .classify(text) - Returns the most likely language and it's likelihood
- .rank(text) - Returns the results for all known languages.

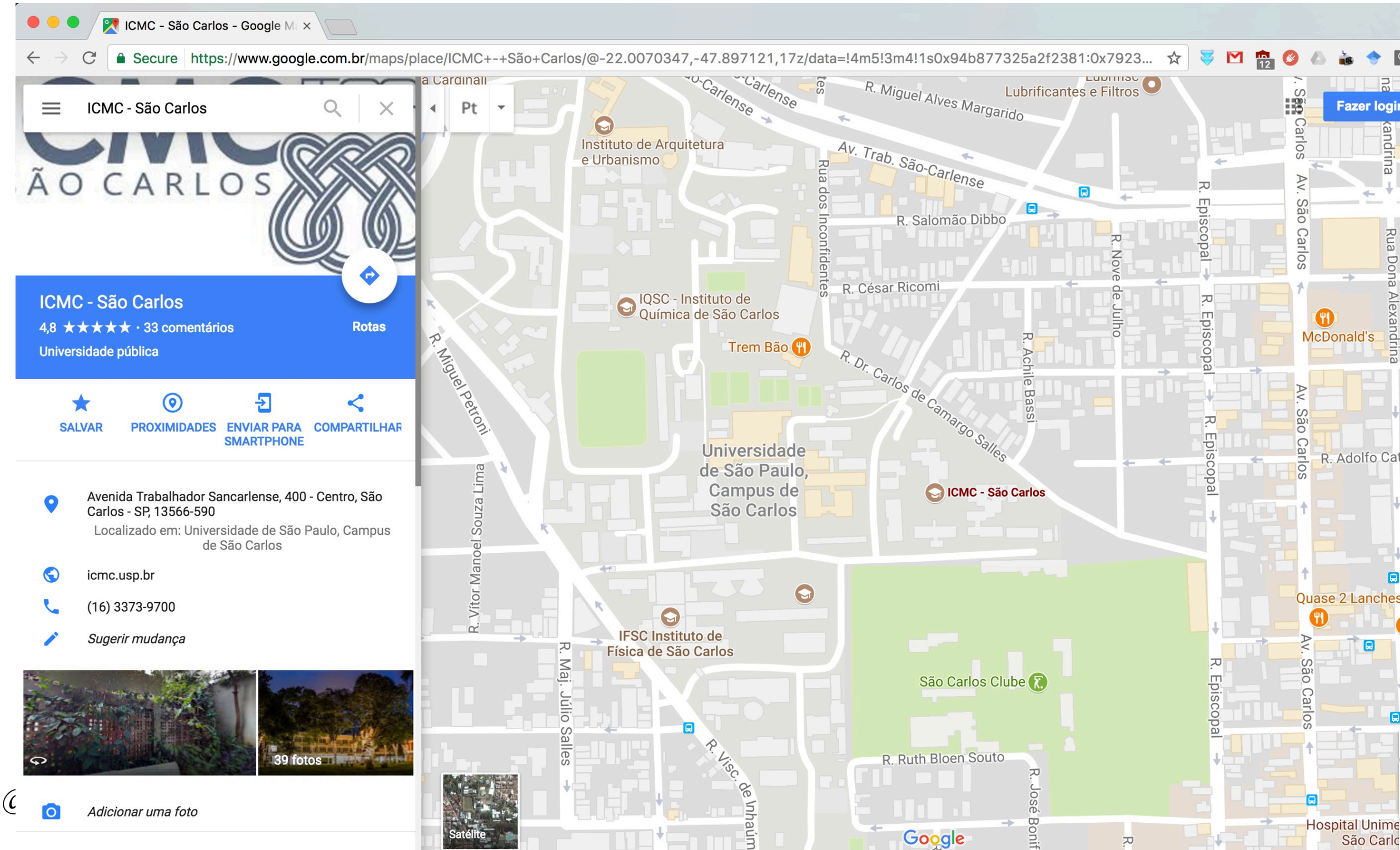
```
import langid

text_it = "Wales lancia la Wikipedia delle news. Contro il fake in campo anche Google"
text_en = "Cassini Spacecraft Re-Establishes Contact After 'Dive' Between Saturn And Its Rings"

lang_it = langid.classify(text_it)
lang_en = langid.classify(text_en)

print(text_it, "is in", lang_it)
print(text_en, "is in", lang_en)
```



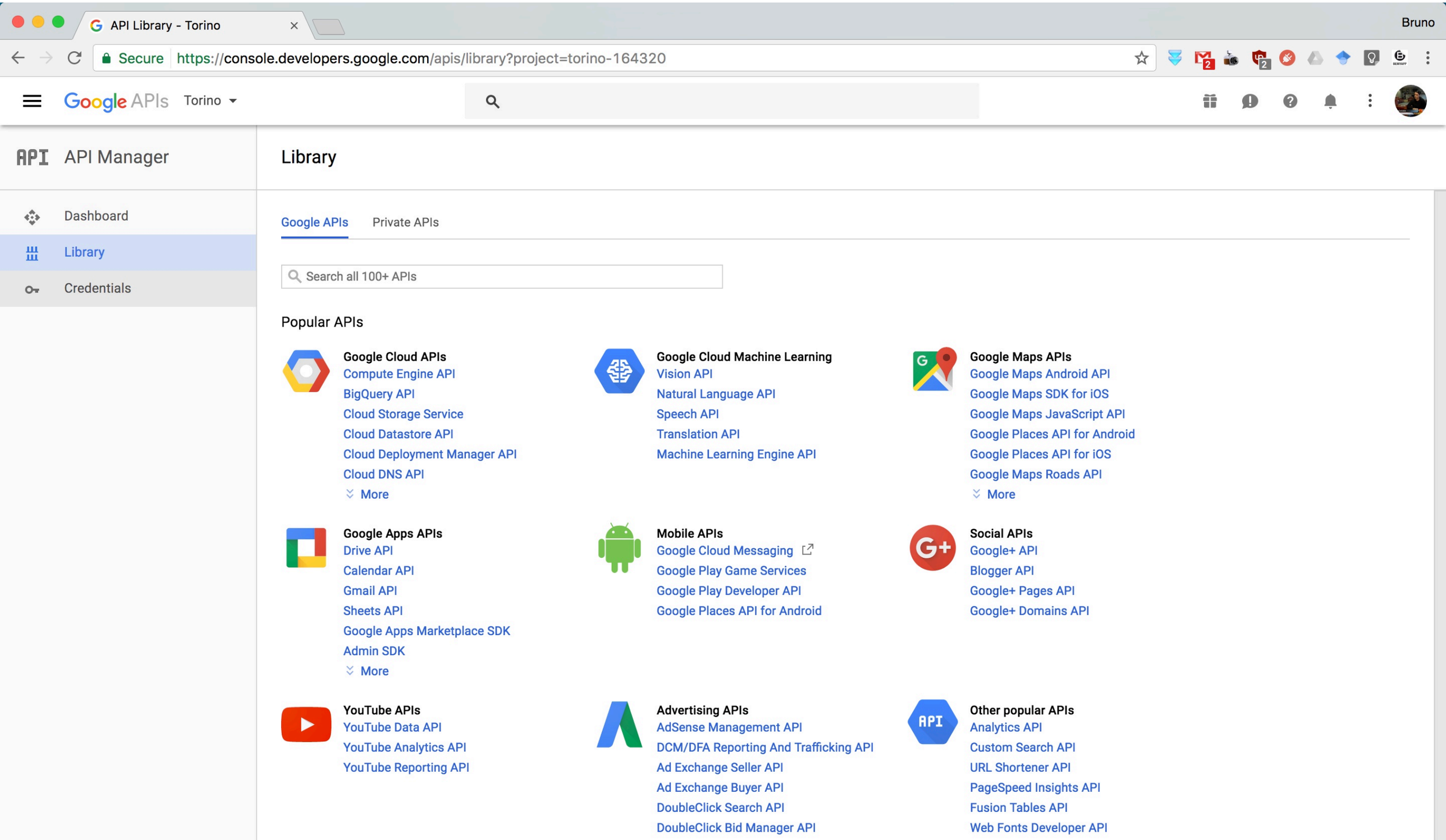
language_langid.py

# Google Maps APIs

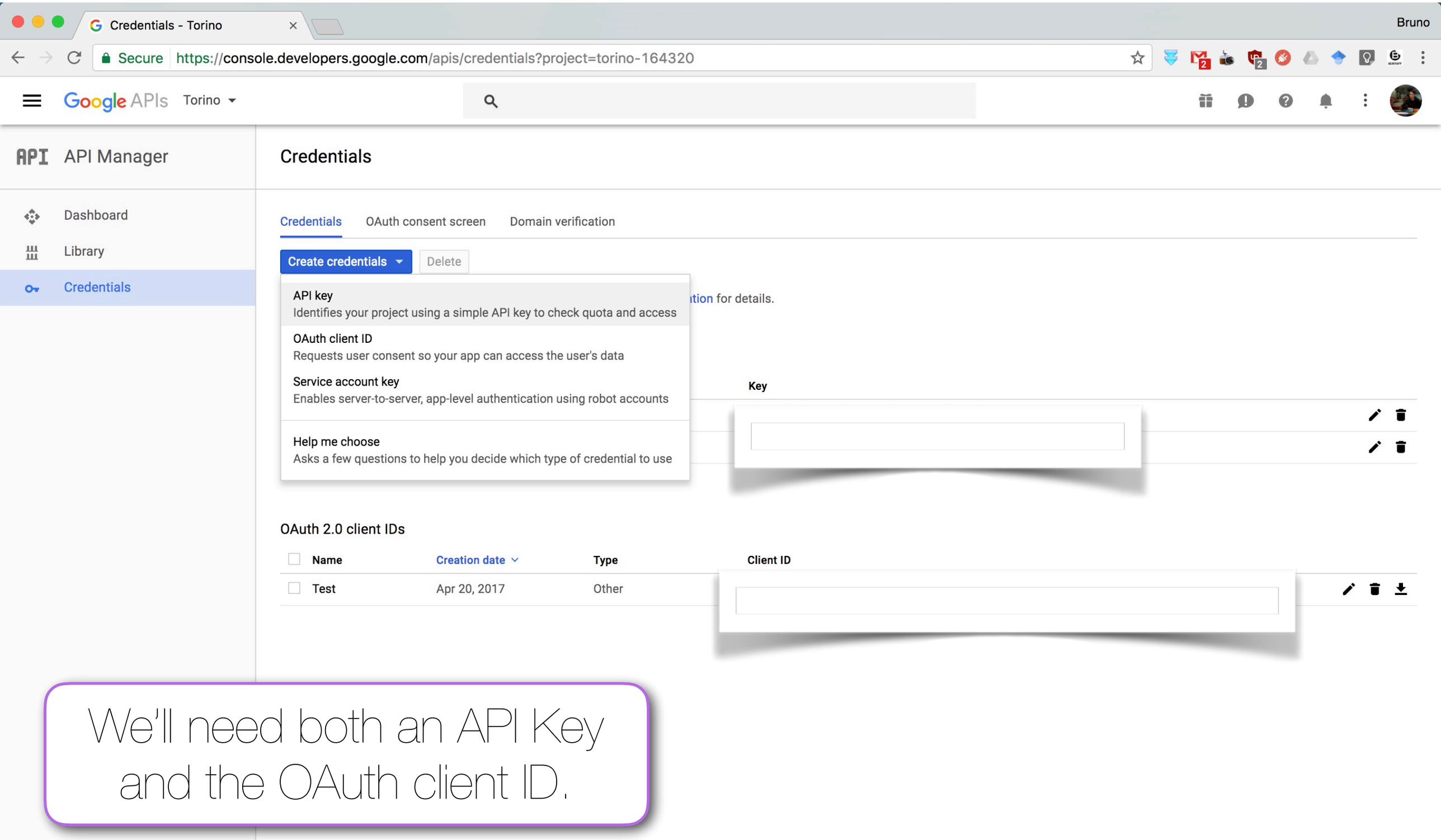
# Google Maps


ICMC - São Carlos - Google Ma
Secure
https://www.google.com.br/maps/place/ICMC+-+São+Carlos/@-22.0070347,-47.897121,17z/data=!4m5!3m4!1s0x94b877325a2f2381:0x7923...
ICMC - São Carlos
Pt
ICMC - São Carlos
4,8 ★★★★★ · 33 comentários
Universidade pública
Rotas
SALVAR
PROXIMIDADES
ENVIAR PARA SMARTPHONE
COMPARTILHAR
Avenida Trabalhador Sancarlense, 400 - Centro, São Carlos - SP, 13566-590
Localizado em: Universidade de São Paulo, Campus de São Carlos
icmc.usp.br
(16) 3373-9700
Sugerir mudança
39 fotos
Adicionar uma foto
Satélite
Instituto de Arquitetura e Urbanismo
IQSC - Instituto de Química de São Carlos
Trem Bão
Universidade de São Paulo, Campus de São Carlos
IFSC Instituto de Física de São Carlos
São Carlos Clube
R. Miguel Alves Margarido
Lubrificantes e Filtros
Av. Trab. São-Carlense
R. Salomão Dibbo
R. César Ricomi
R. Dr. Carlos de Camargo Salles
R. Achile Bassi
R. Episcopal
Av. São Carlos
McDonald's
R. Adolfo Cat
Quase 2 Lanches
R. Miguel Petroni
R. Vitor Manoel Souza Lima
R. Maj. Júlio Salles
R. Visc. de Inhaúm
R. Ruth Bloen Souto
R. José Bonif
Hospital Unime São Carlo
Google

# Google APIs

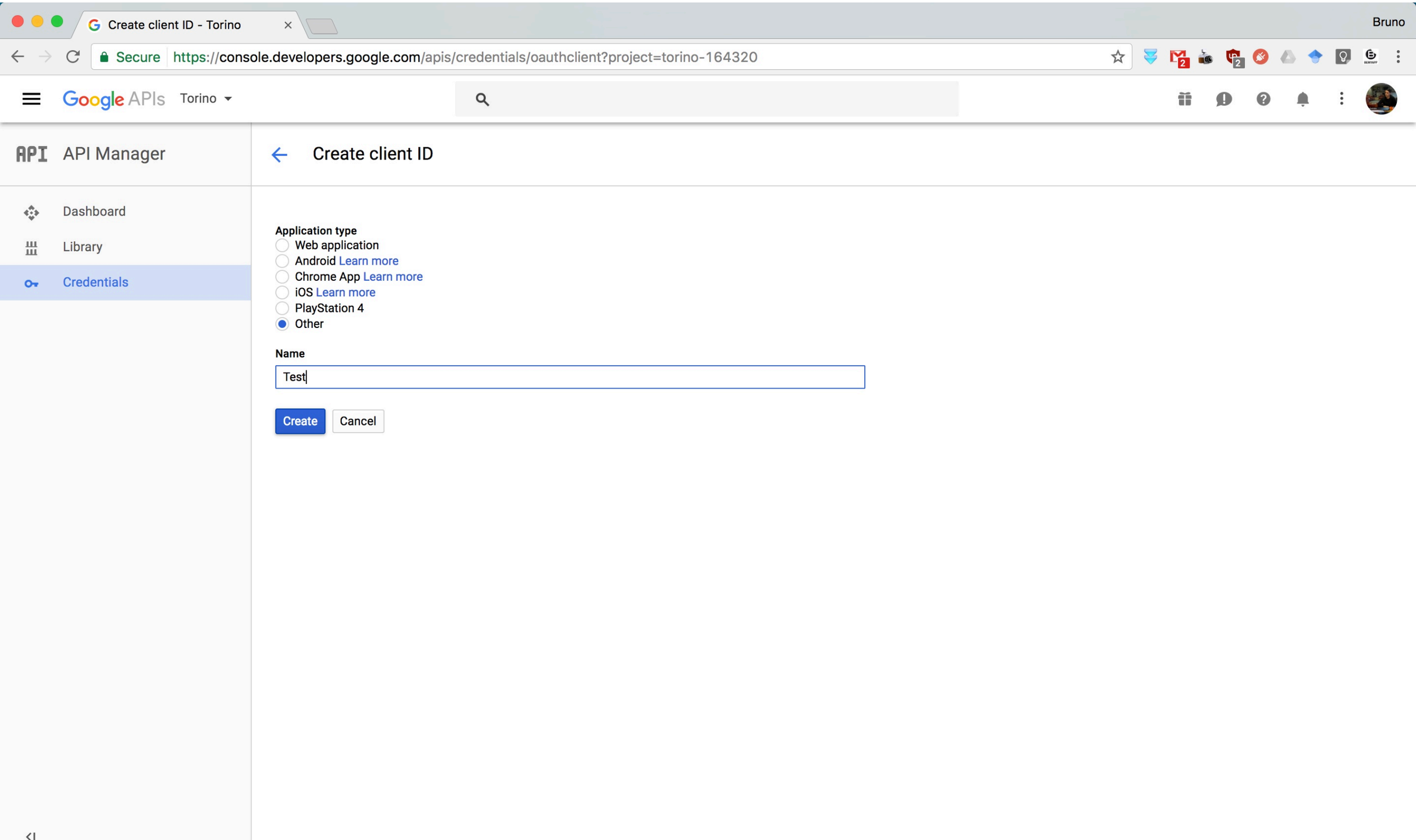

API Library - Torino
Bruno
Secure https://console.developers.google.com/apis/library?project=torino-164320
Google APIs Torino
API Manager
Library
Dashboard
Library
Credentials
Google APIs
Private APIs
Search all 100+ APIs
Popular APIs
Google Cloud APIs
Compute Engine API
BigQuery API
Cloud Storage Service
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

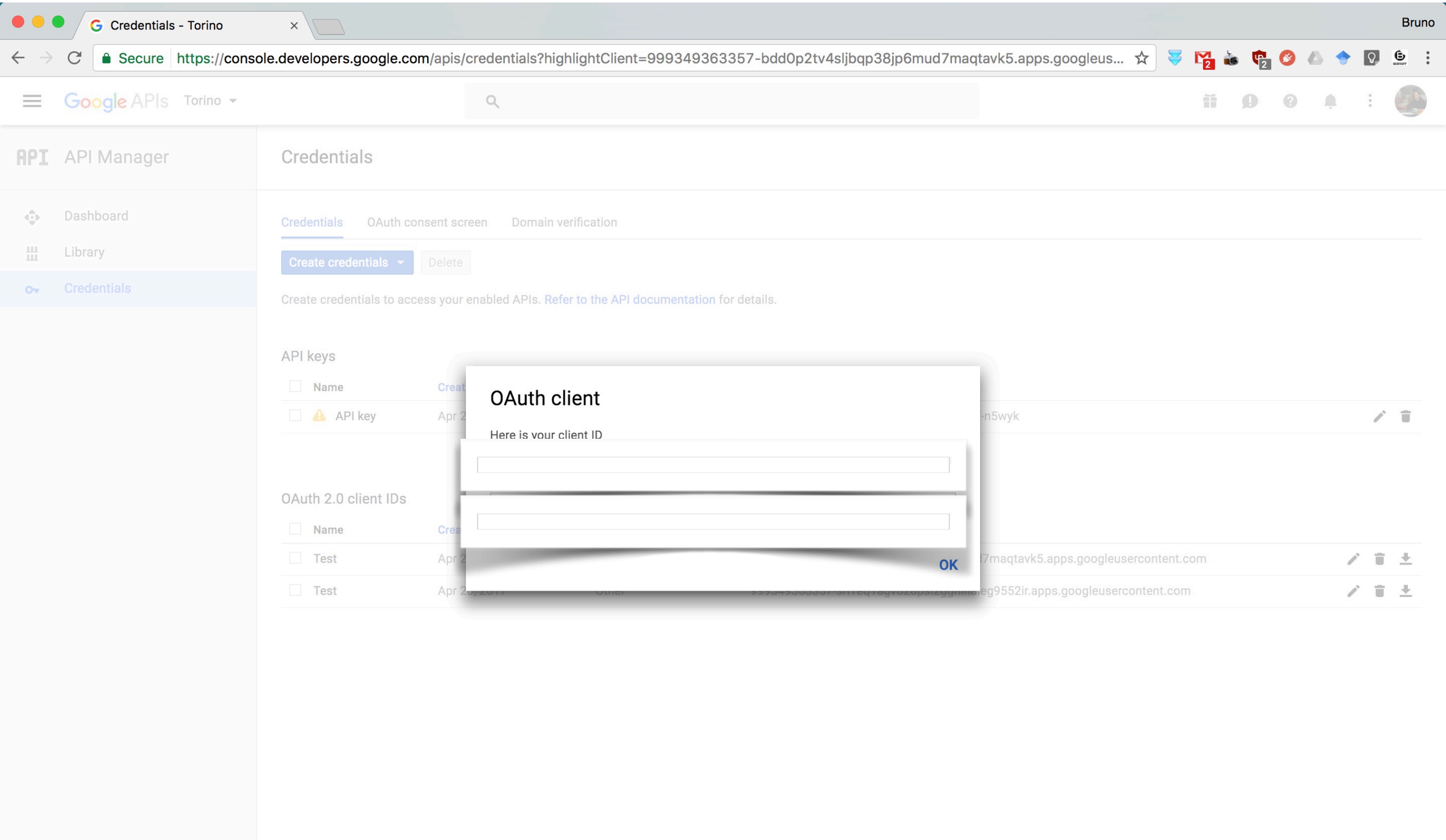



We'll need both an API Key and the OAuth client ID.

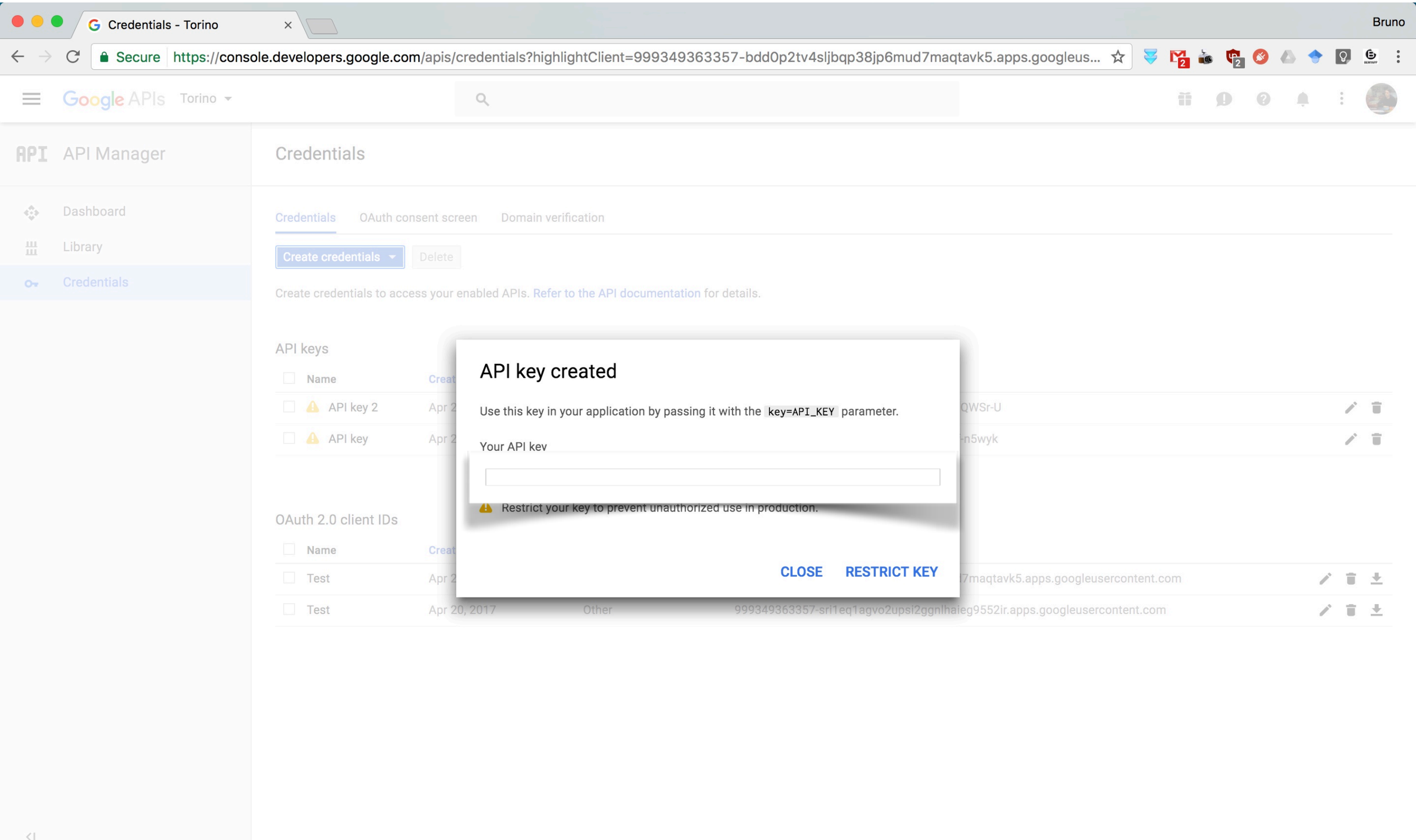
# Google Credentials



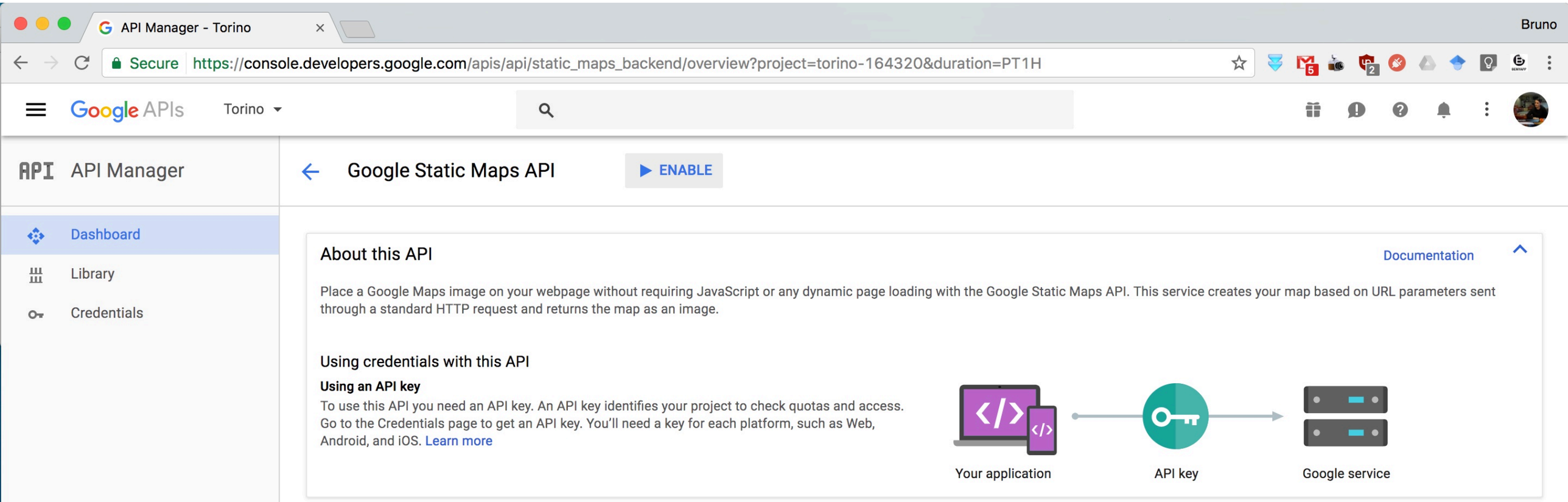
# Google Credentials



# Google Credentials

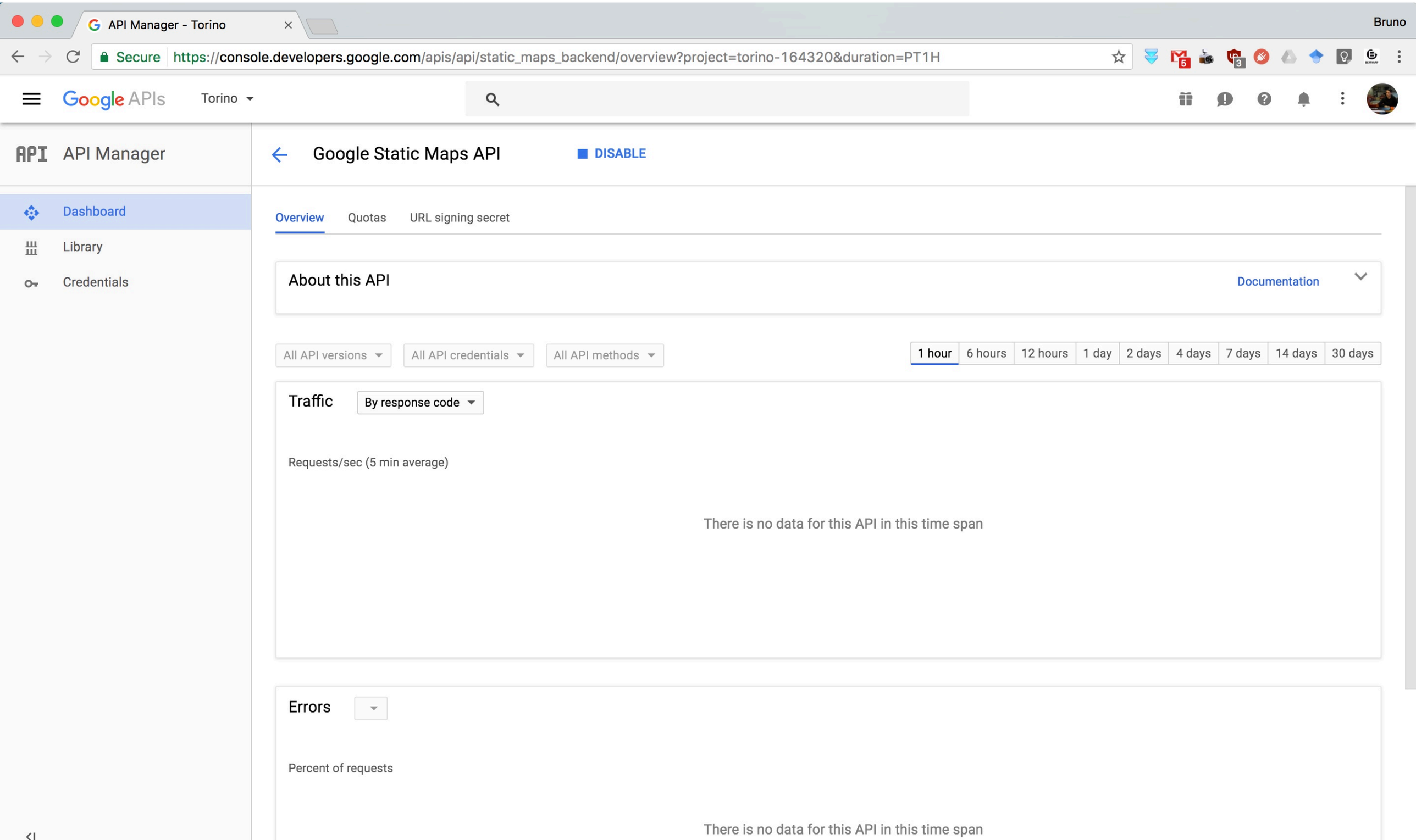


# API Authorization




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]["geography"]["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']}
```

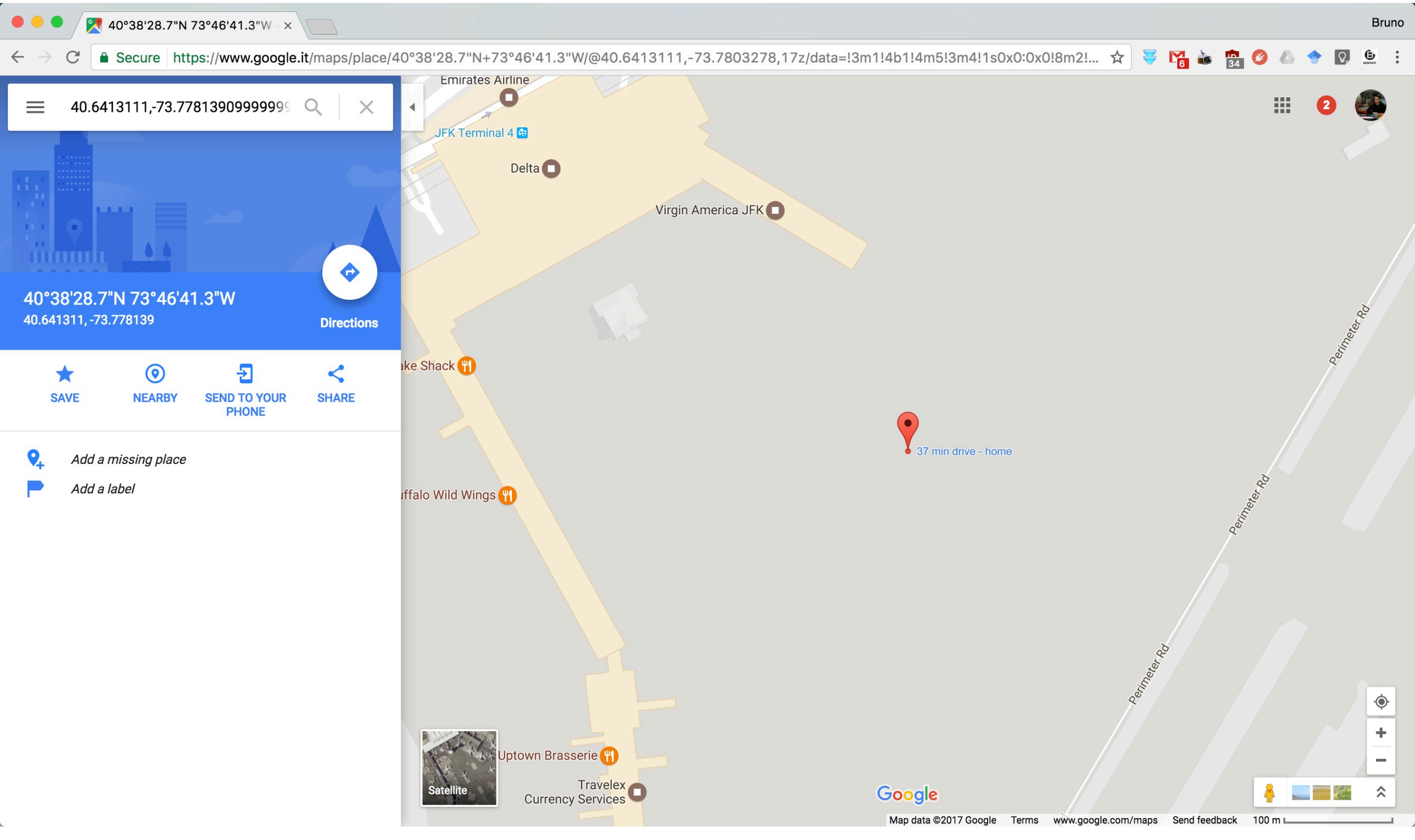
# Google Reverse Geocoding

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

```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"])
```

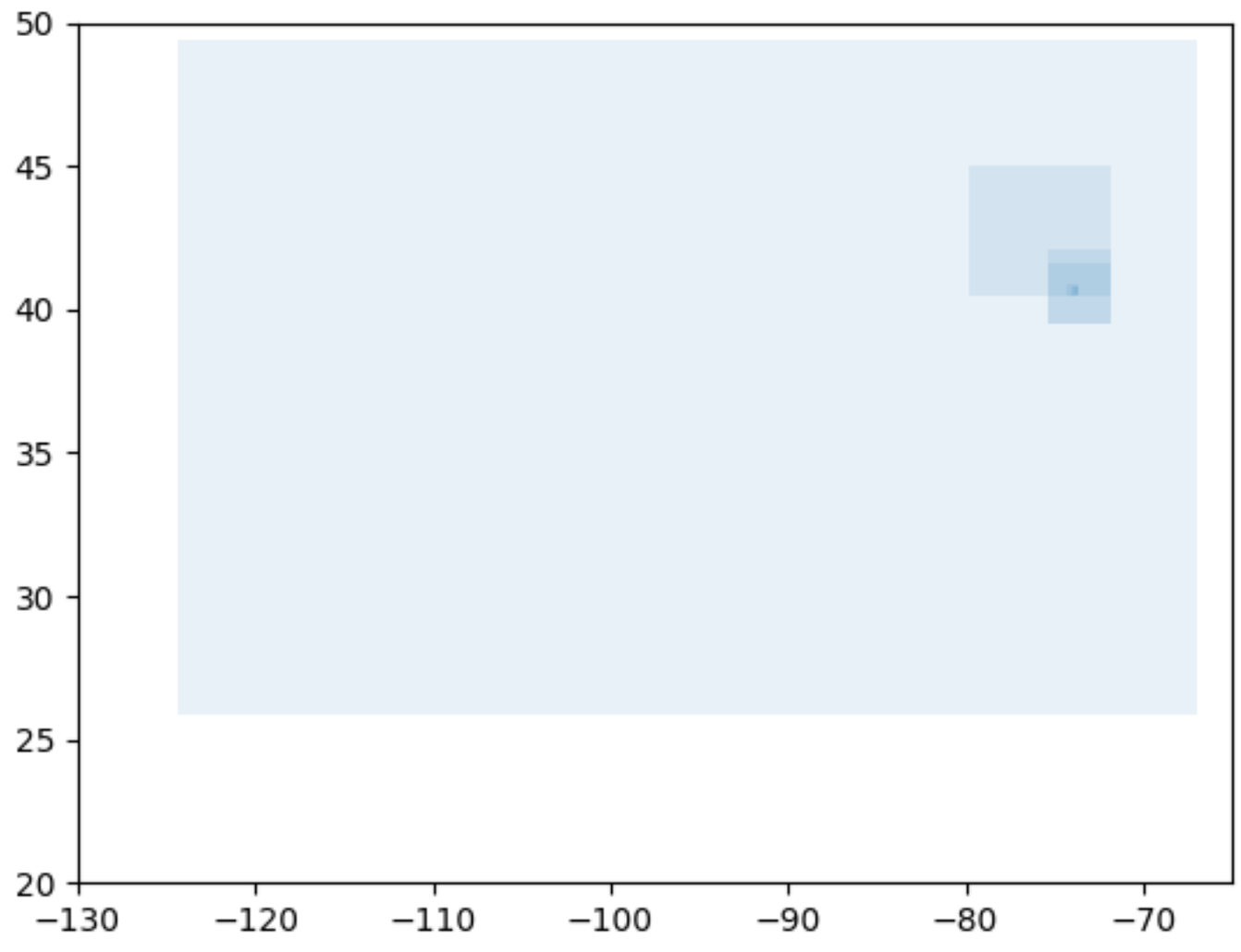
google_reverse_geocode.py

# Google Reverse Geocoding



# Google Reverse Geocoding

```
impo
impo
impo

reve

plt.
plt.
plt.
curr

for

plt.
```



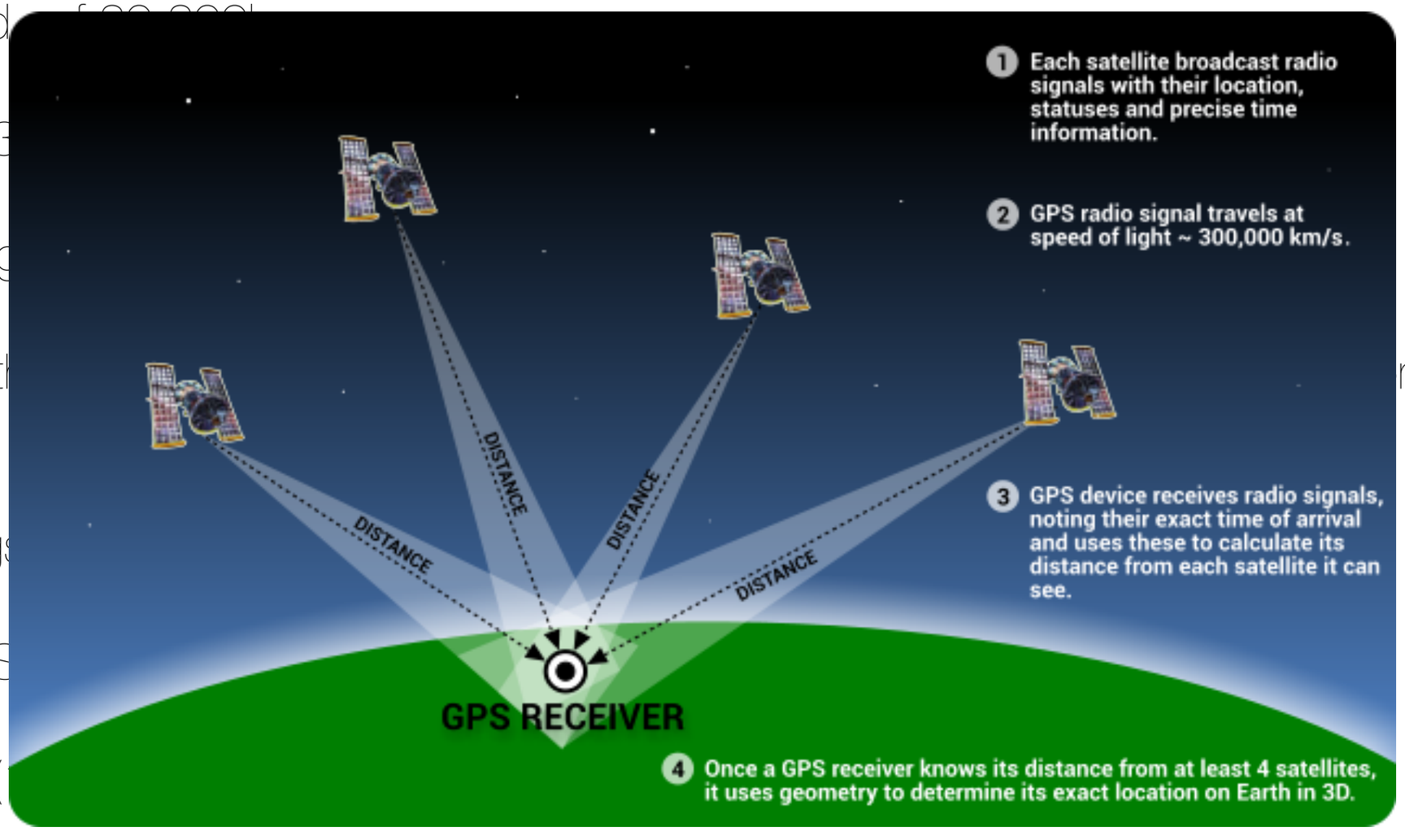

```
))
```

plot_rectangles.py

# Geographical Information Systems

# Global Positioning System (GPS)

- 32 satellites
- Orbital altitud
- Traveling at 3
- Broadcasting
- Position on t m 4 or more satellites.
- GPS belongs
  - GLONAS
  - BeiDou (
  - Galileo - Europe

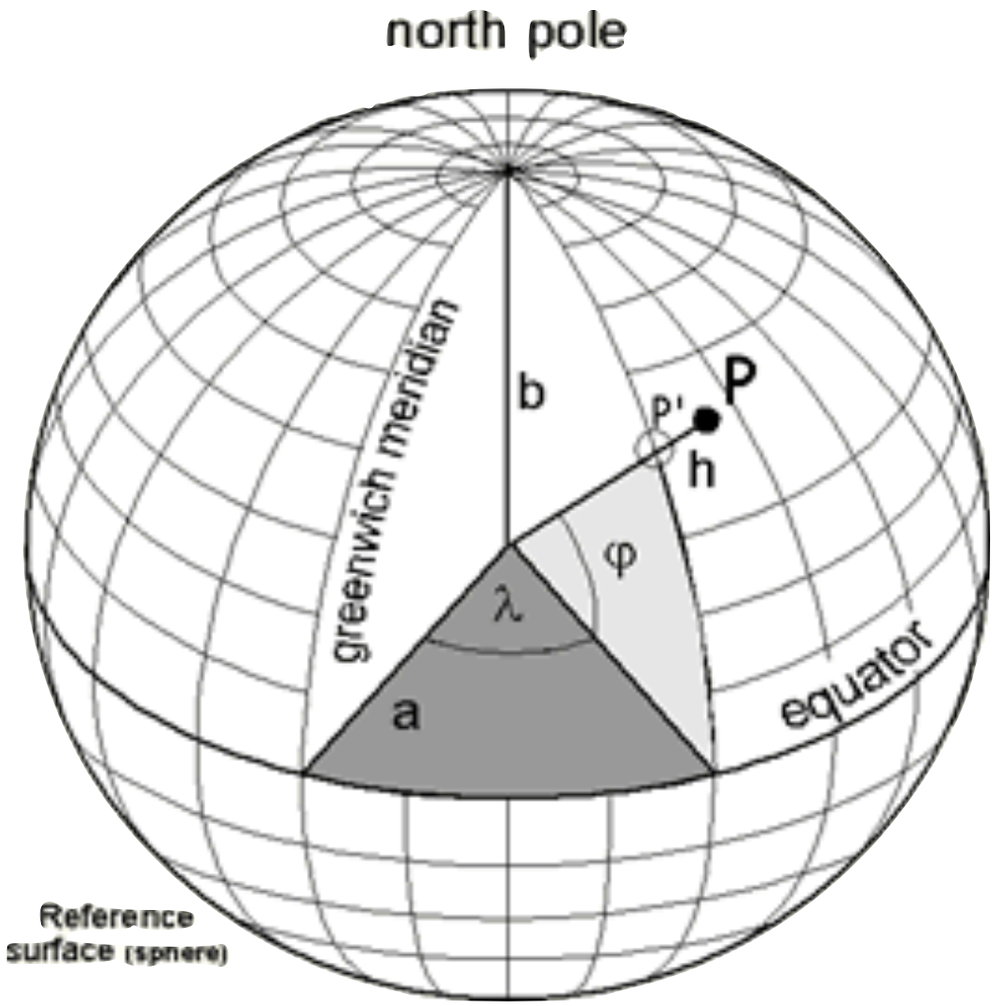


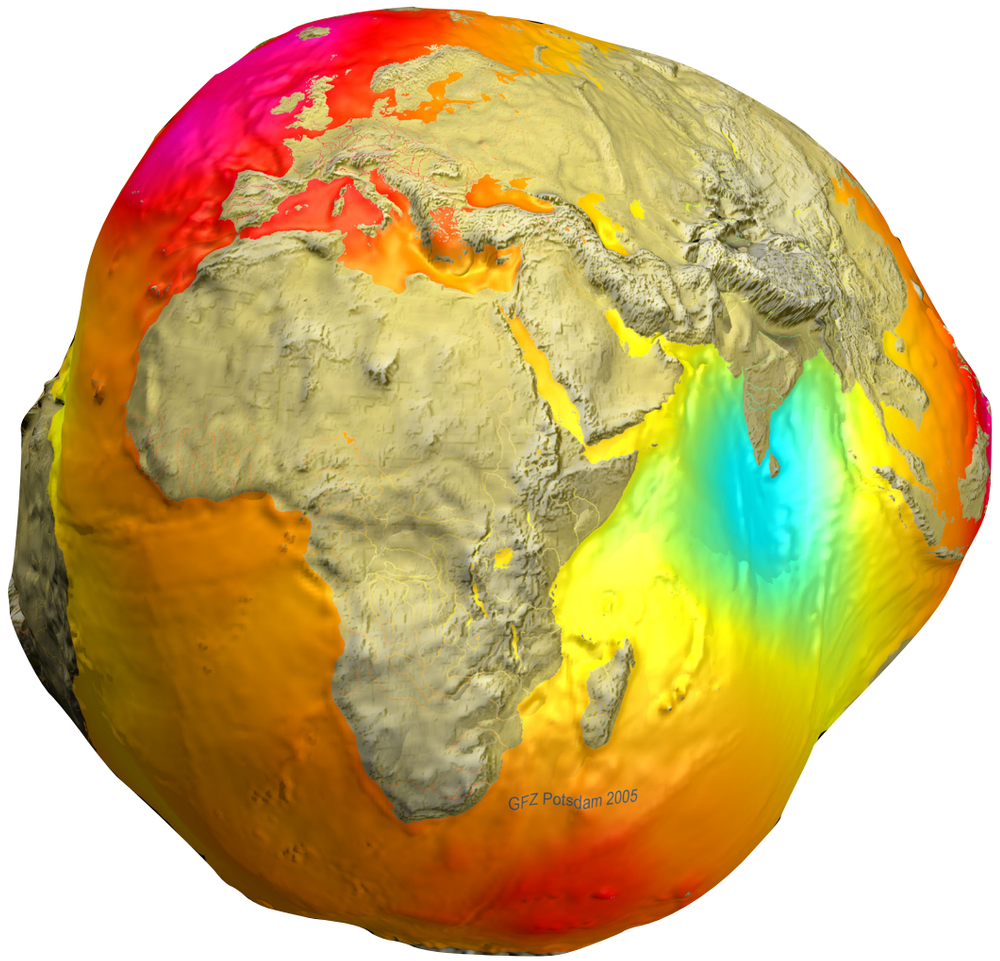
# Global Positioning System (GPS)



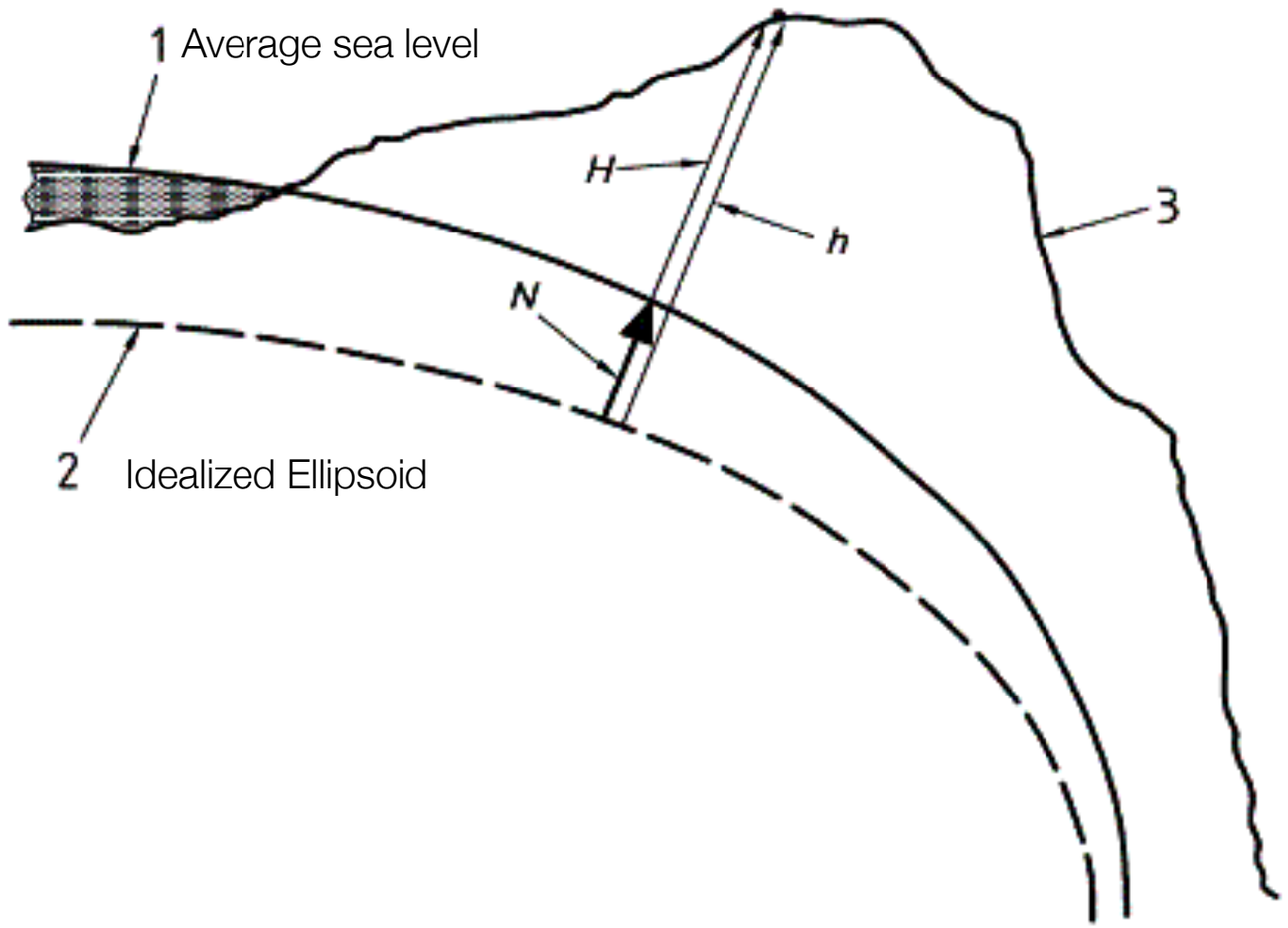


# Global Positioning System (GPS)



# Global Positioning System (GPS)




**Key**

1 geoid
2 ellipsoid
3 surface of the Earth

$h$ = ellipsoidal height, measured from ellipsoid along perpendicular passing through point; $h = H + N$
$H$ = gravity-related height, measured along direction of gravity from vertical datum plane at geoide
$N$ = geoid height, height of geoid above ellipsoid

# WGS84

http://www.unoosa.org/pdf/icg/2012/template/WGS_84.pdf

- "WGS 84 is an Earth-centered, Earth-fixed terrestrial reference system and geodetic datum"

**Coordinate System:** Cartesian Coordinates (X, Y, Z). WGS 84 (G1674) follows the criteria outlined in the International Earth Rotation Service (IERS) Technical Note 21. The WGS 84 Coordinate System origin also serves as the geometric center of the WGS 84 Ellipsoid and the Z-axis serves as the rotational axis of this ellipsoid of revolution. WGS 84 geodetic coordinates are generated by using its reference ellipsoid.

**Defining Parameters:** WGS 84 identifies four defining parameters. These are the semi-major axis of the WGS 84 ellipsoid, the flattening factor of the Earth, the nominal mean angular velocity of the Earth, and the geocentric gravitational constant as specified below.

| Parameter | Notation | Value |
|---|---|---|
| Semi-major Axis | a | 6378137.0 meters |
| Flattening Factor of the Earth | 1/f | 298.257223563 |
| Nominal Mean Angular Velocity of the Earth | $\omega$ | $7292115 \times 10^{-11}$ radians/second |
| Geocentric Gravitational Constant (Mass of Earth's Atmosphere Included) | GM** | $3.986004418 \times 10^{14}$ meter$^3$/second$^2$ |

**The value of GM for GPS users is $3.9860050 \times 10^{14}$ m$^3$/sec$^2$ as specified in the references below.

# GIS Data Systems

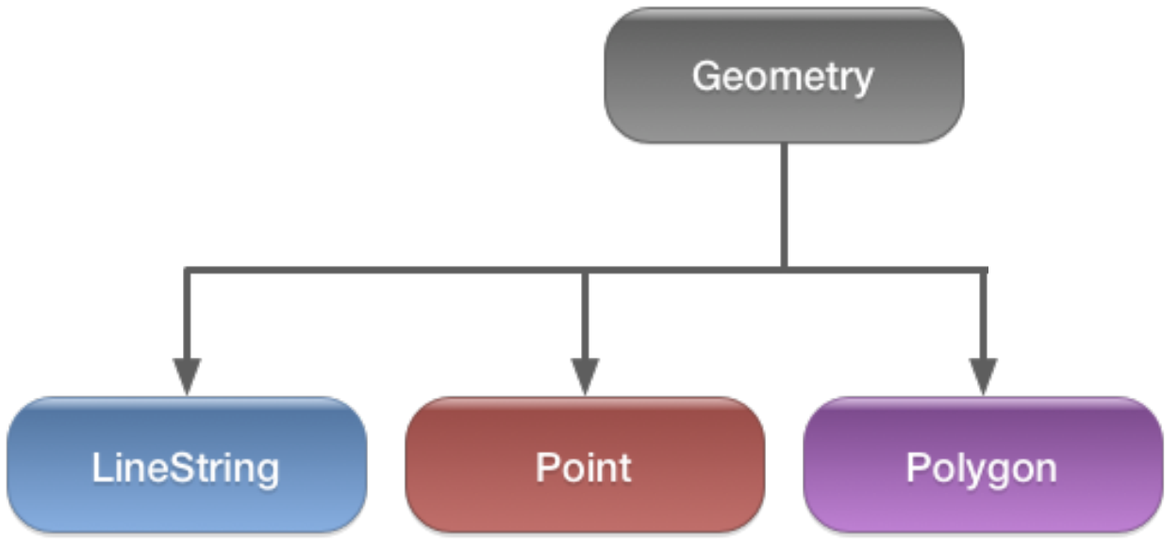
## Vector

- Data types:
  - Points
  - Lines
  - Polygons
- Discrete shapes and boundaries
- Spatial, Database and Network analysis
- Shapefile, GeoJSON, GML, etc…

## Raster

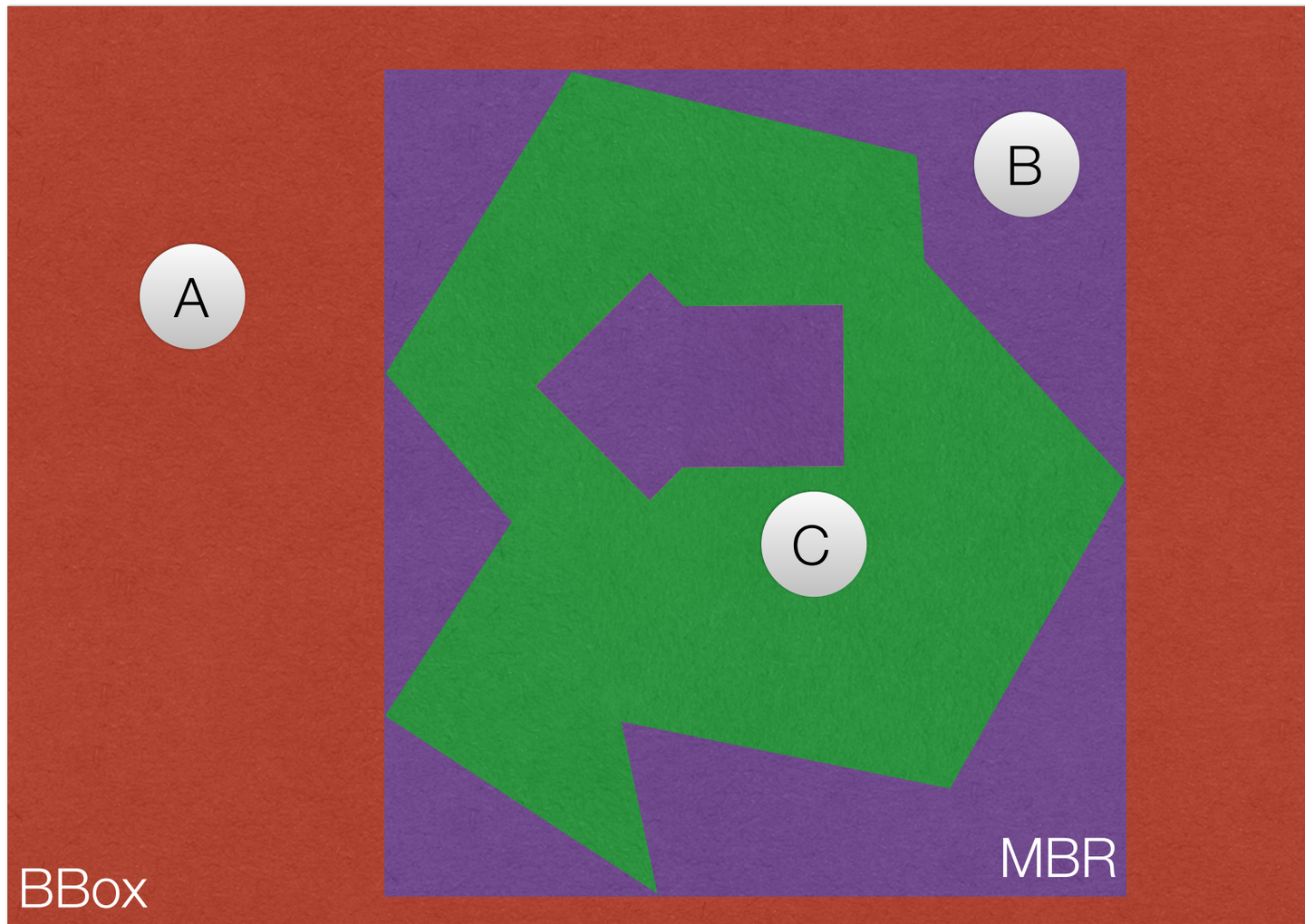
- Data types:
  - Cells
  - Pixels
  - Elements
- Dense data, Continuous surfaces
- Spatial Analysis and modeling
- GeoTIFF, ASC, JPEG2000, etc…

# GIS Vector Data types



# Fundamental Concepts

- **Bounding box** - A rectangular box containing the shape of interest
- **Minimum Bounding Rectangle** - The smallest possible Bounding Box that still contains the shape
- **Spatial Reference Identifier** - Maps "flat" lat/lon to a curved surface
  - Specially important to measure distances!
- Rectangles make it easy to quickly check relative positions, but lack precision
  - A within BBox but outside MBR
  - B within MBR but outside shape
  - C within shape
- If BBoxes don't overlap, neither do the shapes they contain

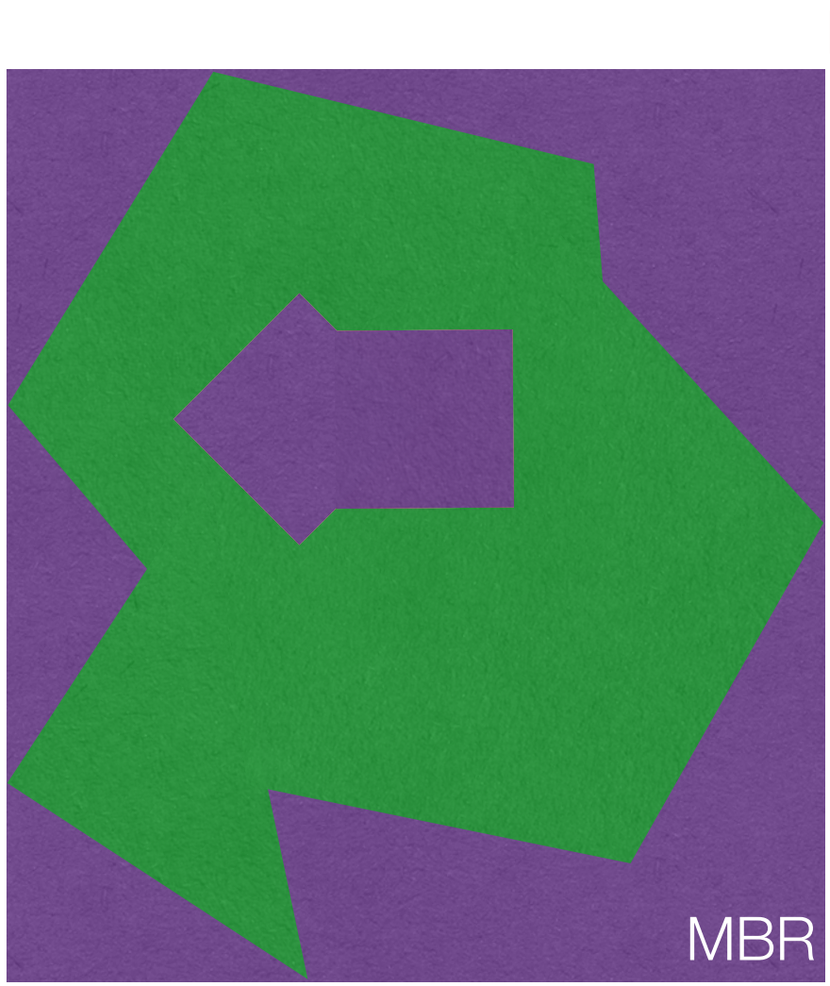


# Get MBR/BBox

- You'll often see the term **BBox** to mean the **MBR**

- The MBR can be uniquely defined by the coordinates of two corners.

- Coordinates are simply the extreme values in the four "cardinal" directions, North, South, East and West of all the points defining the shape.



# GeoJSON

https://tools.ietf.org/html/rfc7946

- A GeoJSON object is a JSON object specially tailored to spatial data

- It defines

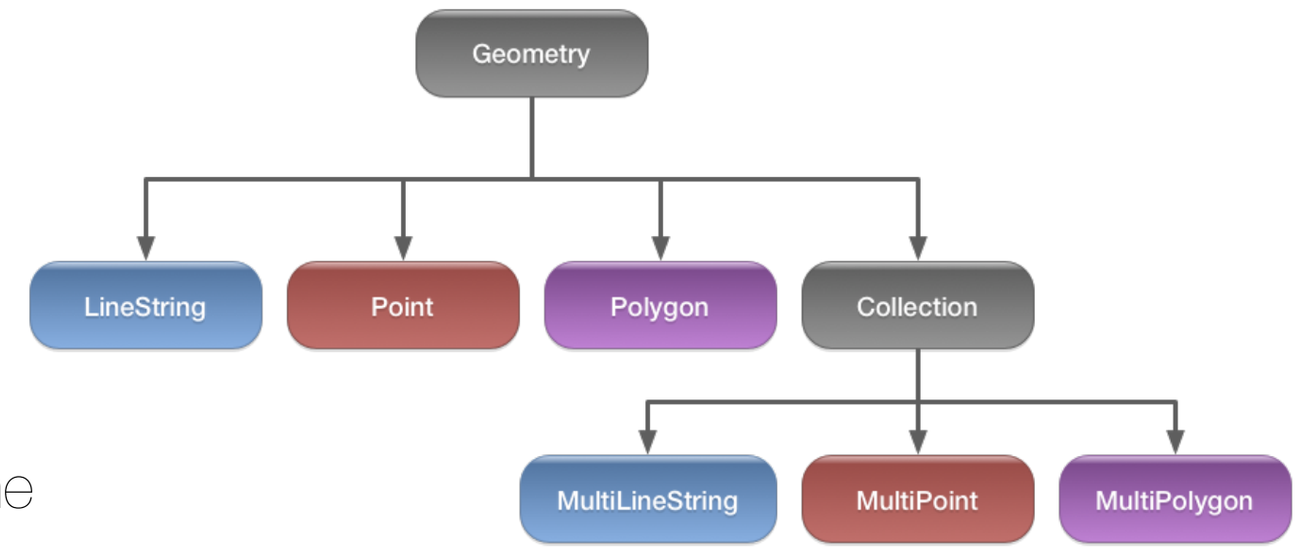
    - Geometry - a region of space

    - Feature - a spatially bounded entity that contains a contains a Geometry object

    - FeatureCollection - A list of Features

- Supports all types of GIS data types:

- Widely supported as an open standard online



# GeoJSON

```
['type',
 'crs',
 'features']
```

# GeoJSON

```
['type',          'FeatureCollection'
 'crs',           {"properties": {"name": "urn:ogc:def:crs:EPSG::4258"},
 'features']       "type": "name"}
```

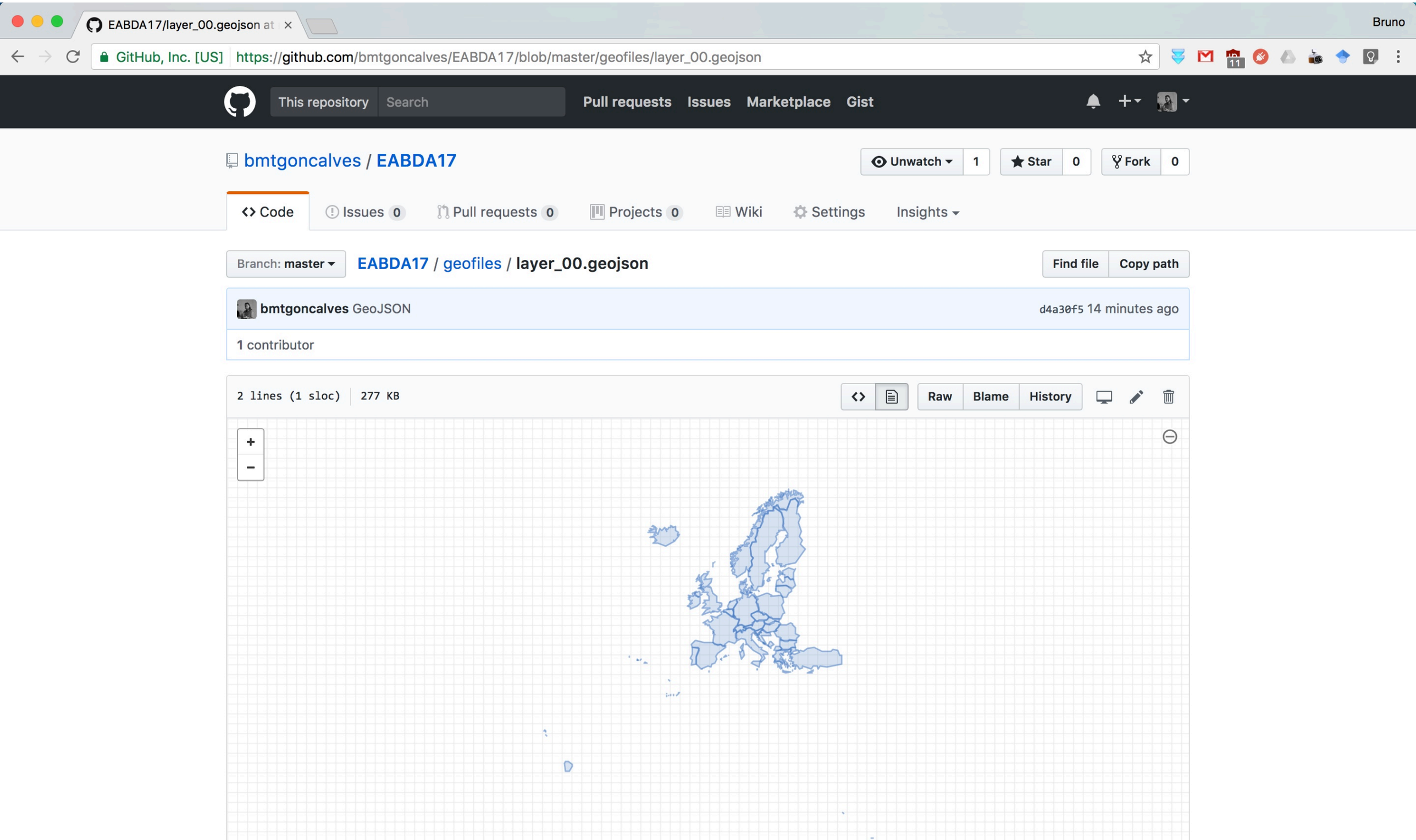
# GeoJSON

```
['type',
 'crs',
 'features']   {'geometry': {'coordinates': [[[[19.224069, 43.527541],
                                               [19.227058, 43.47903949800002],
                                               (…)
                                               [19.049223, 43.501789001000002],
                                               [19.224069, 43.527541]]]],
                             'type': 'MultiPolygon'},
                'properties': {'NUTS_ID': 'ME',
                               'SHAPE_AREA': 1.50797533788,
                               'SHAPE_LEN': 7.40877787465,
                               'STAT_LEVL_': 0},
               'type': 'Feature'}
```

# GeoJSON

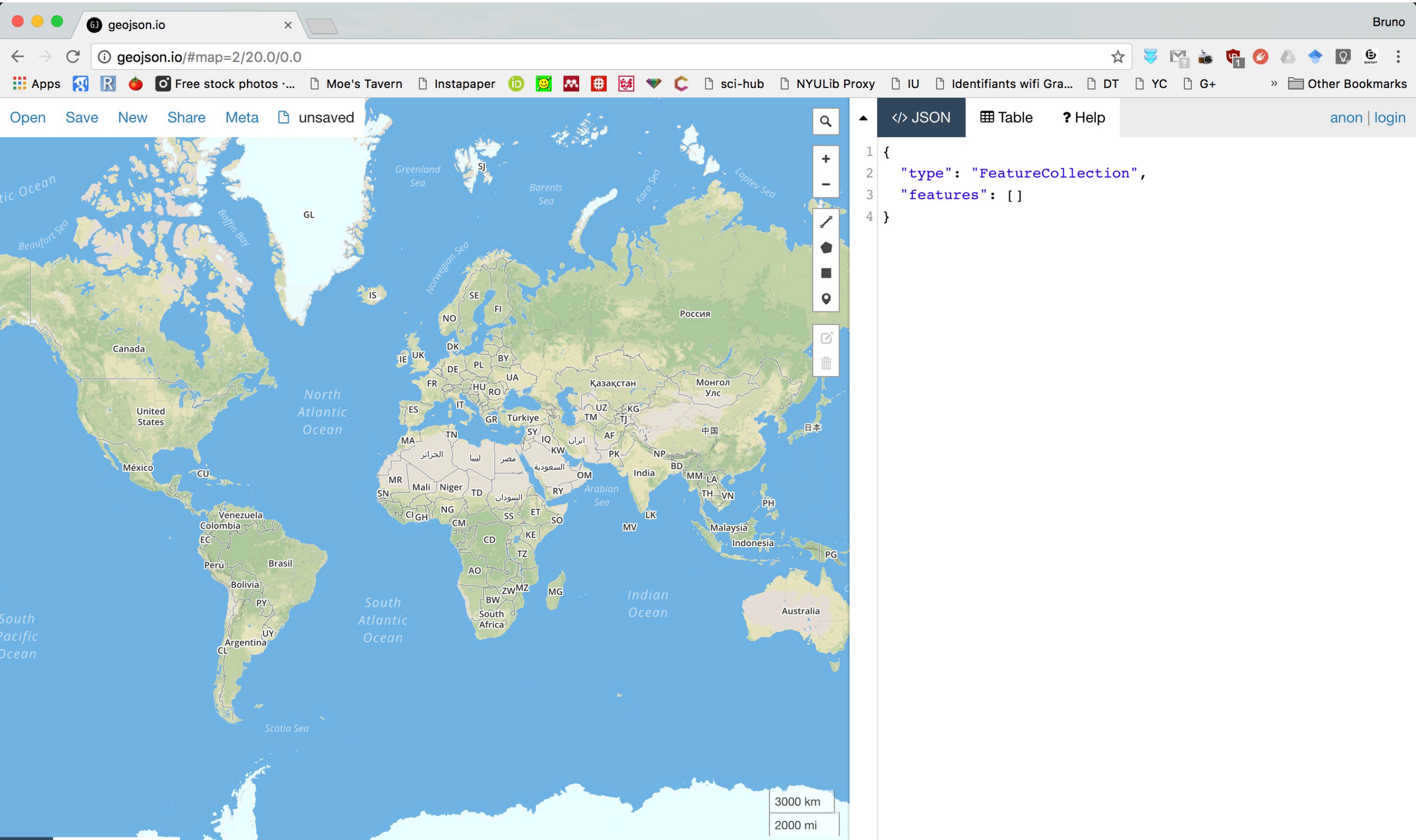
```
['type',
 'crs',
 'features']   {'geometry': {'coordinates': [[[[19.224069, 43.527541],
                                               [19.227058, 43.47903949800002],
                                               (…)
                                               [19.049223, 43.501789001000002],
                                               [19.224069, 43.527541]]]],
                             'type': 'MultiPolygon'},
                'properties': {'NUTS_ID': 'ME',
                               'SHAPE_AREA': 1.50797533788,
                               'SHAPE_LEN': 7.40877787465,
                               'STAT_LEVL_': 0},
               'type': 'Feature'}
```

# GitHub support



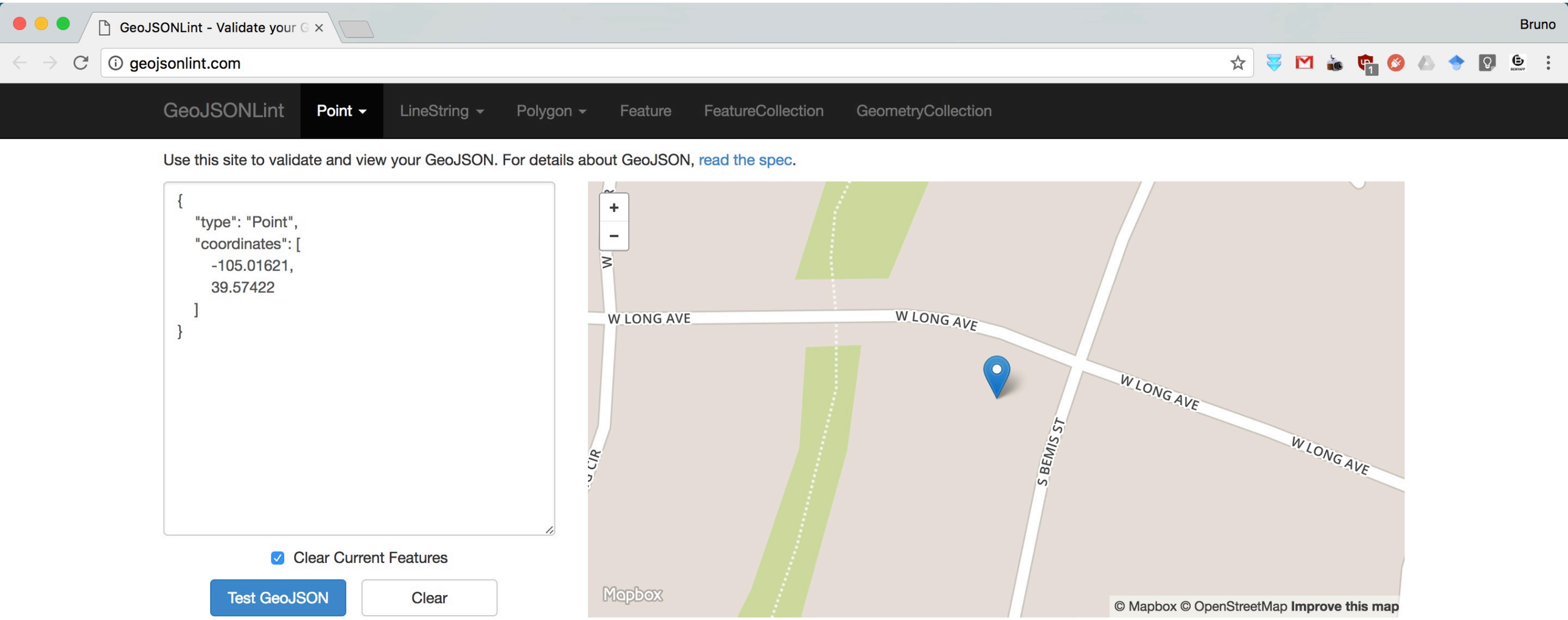
# GeoJSON

geojson.io



# GeoJSON

http://geojsonlint.com/



# GeoJSON

```python
def get_bbox(country):
    maxLat = None
    maxLon = None
    minLat = None
    minLon = None

    for polygon in country["geometry"]["coordinates"]:
        coords = np.array(polygon)[0]

        curMaxLat = np.max(coords.T[1])
        curMinLat = np.min(coords.T[1])

        curMaxLon = np.max(coords.T[0])
        curMinLon = np.min(coords.T[0])

        if maxLat is None or curMaxLat > maxLat:
            maxLat = curMaxLat

        if maxLon is None or curMaxLon > maxLon:
            maxLon = curMaxLon

        if minLat is None or curMinLat < minLat:
            minLat = curMinLat

        if  minLon is None or curMinLon < minLon:
            minLon = curMinLon

    return maxLat, maxLon, minLat, minLon
```

get_bbox.py

# GeoJSON

```
def plot_country(country):
    for polygon in country["geometry"]["coordinates"]:
        coords = np.array(polygon)

        plt.plot(coords.T[0], coords.T[1])

    maxLat, maxLon, minLat, minLon = get_bbox(country)

    plt.xlim(minLon, maxLon)
    plt.ylim(minLat, maxLat)
```

```
data = json.load(open('geofiles/NUTS_RG_20M_2013.geojson'))

countries = {}

countries["crs"] = data["crs"]
countries["type"] = data["type"]
countries = {}

for feat in data["features"]:
    if feat["properties"]["STAT_LEVL_"] == 0:
        countries[feat["properties"]["NUTS_ID"]] = feat

country = countries["EL"]

plot_country(country)
plt.savefig('Greece.png')
```



get_bbox.py

# Shapefiles

http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

- Open specification developed by ESRI, still the current leader in commercial GIS software
- shapefiles aren't actua (individual) files…
- but actually a set of files sharing the same name but with different extensions:

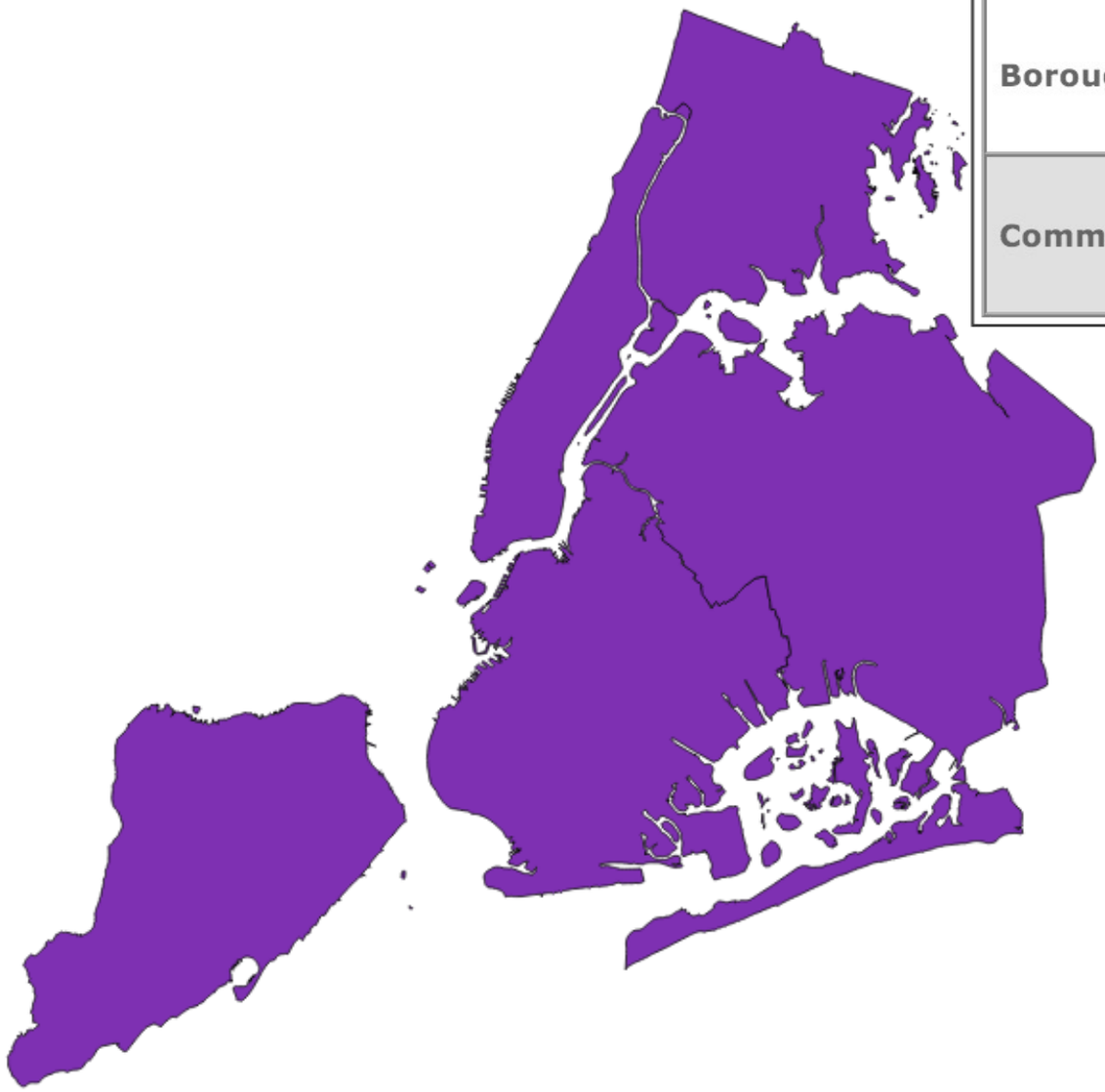
```
(py35) (master) bgoncalves@underdark:$ls -l
total 4856
-rw-r--r--  1 bgoncalves  staff      536 Apr 17 12:40 nybb_wgs84.dbf
-rw-r--r--  1 bgoncalves  staff      143 Apr 17 12:40 nybb_wgs84.prj
-rw-r--r--  1 bgoncalves  staff      257 Apr 17 12:40 nybb_wgs84.qpj
-rw-r--r--  1 bgoncalves  staff  1217376 Apr 17 12:40 nybb_wgs84.shp
-rw-r--r--  1 bgoncalves  staff      140 Apr 17 12:40 nybb_wgs84.shx
(py35) (master) bgoncalves@underdark:$
```

- the actual set of files changes depending on the contents, but three files are usually present:
  - .shp - also commonly referred to as "the" shapefile. Contains the geometric information
  - .dbf - a simple database containing the feature attribute table.
  - .shx - a spatial index, not strictly required

# Shapefiles

http://www.nyc.gov/html/dcp/html/bytes/districts_download_metadata.shtml#bcd

| Borough Boundaries & Community Districts | Download | Metadata |
|---|---|---|
| Borough Boundaries (Clipped to Shoreline) | (645k) | |
| Borough Boundaries (Water Areas Included) | (31k) | |
| Community Districts (Clipped to Shoreline) | (772k) | |



# QGIS Demo

# pyshp

https://github.com/GeospatialPython/pyshp

- pyshp defines utility functions to load and manipulate Shapefiles programmatically.
- The shapefile module handles the most common operations:
  - .Reader(filename) - Returns a Reader object
- Reader.records()/Reader.iterRecords() returns/iterates over the different records present in the shapefile
- Reader.shapes()/Reader.iterShapes() - returns/Iterates over the different shapes present in the shapefile
- Reader.shapeRecords()/Reader.iterShapeRecords() returns/Iterates over both shapes and records present in the shapefile
- Reader.record(index)/Reader.shape(index)/Reader.shapeRecord(index) - return the record/shape/shapeRecord at index position index
- Reader.numRecords - returns the number of records in the shapefile

# pyshp

https://github.com/GeospatialPython/pyshp

```
import sys
import shapefile

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')

print("Found", shp.numRecords, "records:")

recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))

for record, id in recordDict.items():
    print(id, record)
```



shapefile_load.py

# pyshp

https://github.com/GeospatialPython/pyshp
http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

- shape objects contain several fields:
  - bbox - lower left and upper right x,y coordinates (long/lat) - optional
  - parts - list of indexes for the first point of each of the parts making up the shape.
  - points - x,y coordinates for each point in the shape.
  - shapeType - integer representing the shape type - all shapes in a shapefile are required to be of the same shapeType or null.

| Value | Shape Type |
|---|---|
| 0 | Null Shape |
| 1 | Point |
| 3 | PolyLine |
| 5 | Polygon |
| 8 | MultiPoint |
| 11 | PointZ |
| 13 | PolyLineZ |
| 15 | PolygonZ |
| 18 | MultiPointZ |
| 21 | PointM |
| 23 | PolyLineM |
| 25 | PolygonM |
| 28 | MultiPointM |
| 31 | MultiPatch |

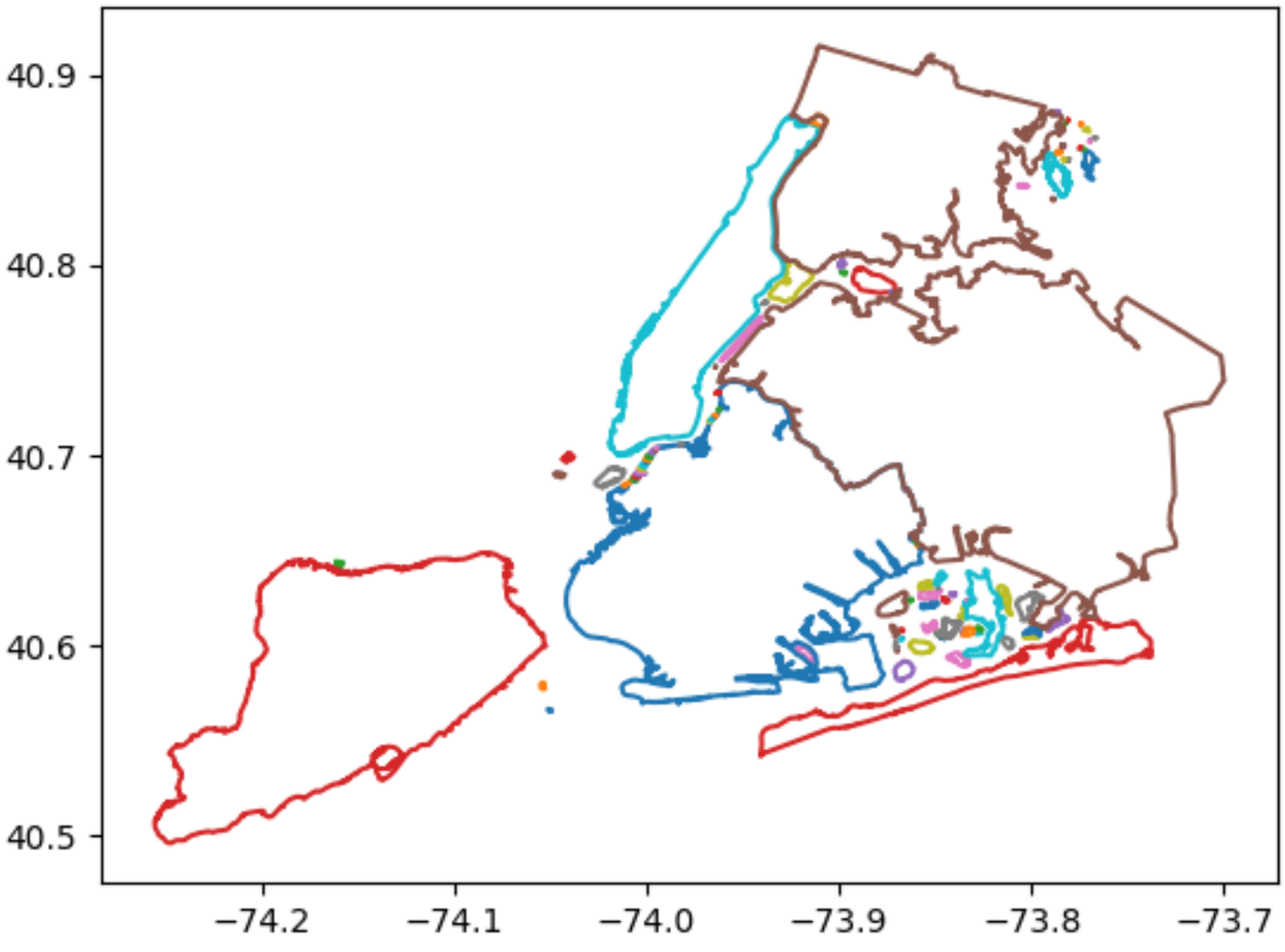
# pyshp

```
import shapefile
import matplotlib.pyplot as plt
import numpy as np

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')

pos = None
count = 0
for shape in shp.iterShapes():
    points = np.array(shape.points)
    parts = shape.parts
    parts.append(len(shape.points))

    for i in range(len(parts)-1):
        plt.plot(points.T[0][parts[i]:parts[i+1]], points.T[1][parts[i]:parts[i+1]])

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



plot_shapefile.py

# pyshp




*@bgoncalves*

# shapely

http://toblerity.org/shapely/manual.html

- Shapely defines geometric objects under shapely.geometry:
  - Point
  - Polygon
  - MultiPolygon
  - shape() Convenience function that creates the appropriate geometric object
- and common operations
  - .crosses(shape) - if it partially overlaps shape
  - .contains(shape) - wether it contains or not the object shape
  - .within(shape)- wether it is contained by object shape
  - .touches(shape) - if the boudaries of this object touch shape

# shapely

http://toblerity.org/shapely/manual.html

- shape objects provide useful fields to query a shapes properties:
  - .centroid - The centroid ("center of mass") of the object
  - .area - returns the area of the object
  - .bounds - the MBR of the shape in (minx, miny, maxx, maxy) format
  - .length - the length of the shape
  - .geom_type - the Geometry Type of the object
- shapely.shape is also able to easily load pyshp's shape objects to allow for further manipulations.

# shapely

http://toblerity.org/shapely/manual.html

```
import sys
import shapefile
from shapely.geometry import shape

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')

recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))

manhattan = shape(shp.shape(recordDict["Manhattan"]))

print("Centroid:", manhattan.centroid)
print("Bounding box:", manhattan.bounds)
print("Geometry type:", manhattan.geom_type)
print("Length:", manhattan.length)
```



shapefile_shape_properties.py

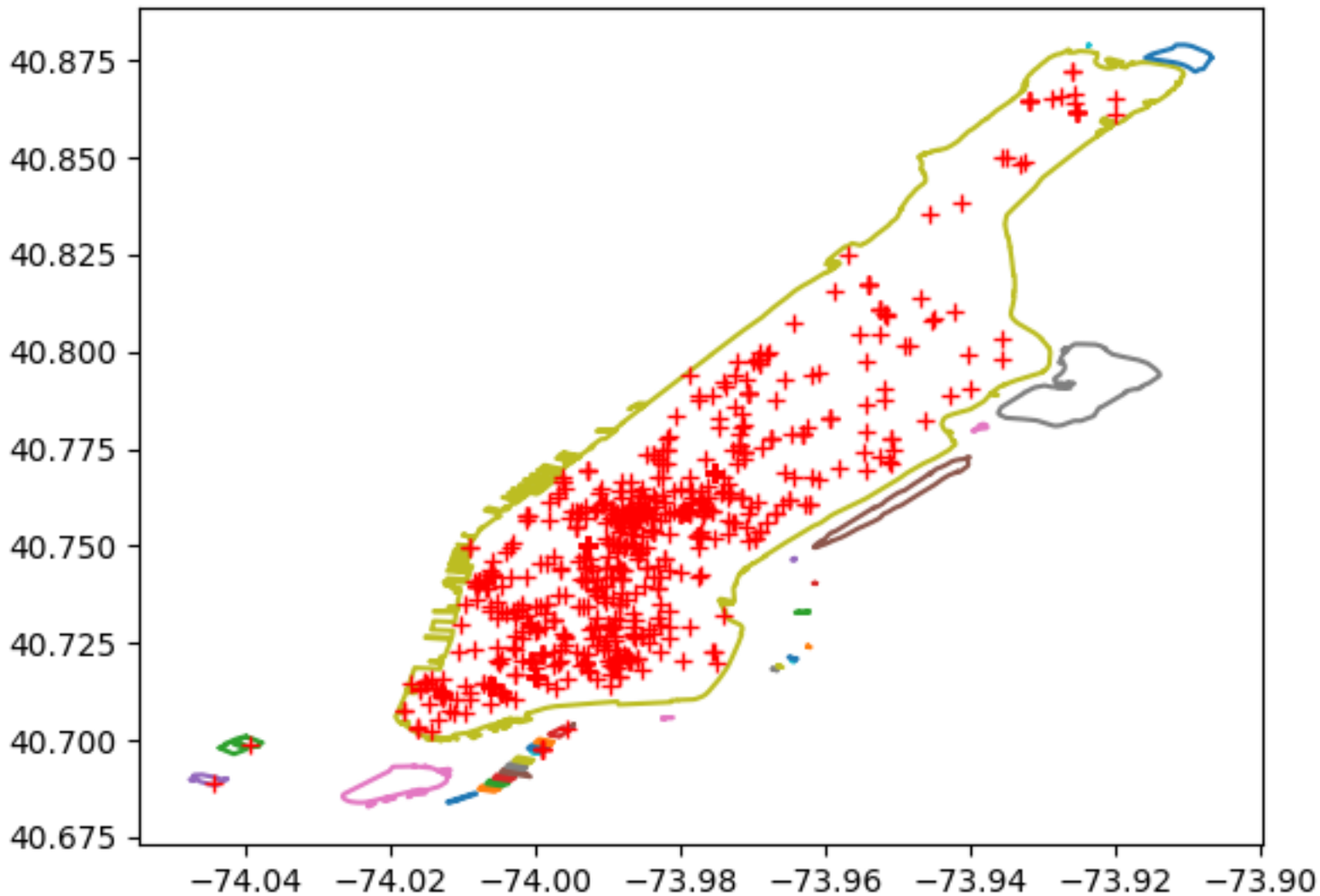
# Filter points within a Shapefile

```
import sys
import shapefile
from shapely.geometry import shape, Point
import gzip

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')

recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))

manhattan = shape(shp.shape(recordDict["Manhattan"]))
fp = gzip.open("Manhattan.json.gz", "w")

for line in gzip.open("NYC.json.gz"):
    try:
        tweet = eval(line.strip())

        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])

            if manhattan.contains(point):
                fp.write(line)
    except:
        pass

fp.close()
```

@bgoncalves

shapefile_filter.py

# Filter points within a Shapefile




*@bgoncalves*

# Twitter places

- As we saw last week, Twitter defines a "coordinates" field in tweets
- There is also a "place" field that we glossed over.
- The **place** object contains also geographical information, but at a courser resolution than the **coordinates** field.
- Each place has a unique place_id, a bounding_box and some geographical information, such as **country** and **full_name**:

```
{'attributes': {},
 'bounding_box': {'coordinates': [[[-74.041878, 40.570842],
    [-74.041878, 40.739434],
    [-73.855673, 40.739434],
    [-73.855673, 40.570842]]],
  'type': 'Polygon'},
 'country': 'United States',
 'country_code': 'US',
 'full_name': 'Brooklyn, NY',
 'id': '011add077f4d2da3',
 'name': 'Brooklyn',
 'place_type': 'city',
 'url': 'https://api.twitter.com/1.1/geo/id/011add077f4d2da3.json'}
```

- places can be of several different types: 'admin', 'city', 'neighborhood', 'poi'

# Twitter places

- As we saw earlier, Twitter defines a "coordinates" field in tweets
- There is also a "place" field that we glossed over.
- The **place** object contains also geographical information, but at a courser resolution than the **coordinates** field.
- Each place has a unique place_id, a bounding_box and some geographical information, such as **country** and **full_name**:

```
{'attributes': {},
 'bounding_box': {'coordinates': [[[-74.041878, 40.570842],
    [-74.041878, 40.739434],
    [-73.855673, 40.739434],
    [-73.855673, 40.570842]]],
  'type': 'Polygon'},
 'country': 'United States',
 'country_code': 'US',
 'full_name': 'Brooklyn, NY',
 'id': '011add077f4d2da3',
 'name': 'Brooklyn',
 'place_type': 'city',
 'url': 'https://api.twitter.com/1.1/geo/id/011add077f4d2da3.json'}
```

The bounding_box field is GeoJSON formatted and compatible with pyshp.shape

- places can be of several different types: 'admin', 'city', 'neighborhood', 'poi'

# Twitter places

https://dev.twitter.com/overview/api/places

## Place Attributes

Place Attributes are metadata about places. An attribute is a key-value pair of arbitrary strings, but with some conventions.

Below are a number of well-known place attributes which may, or may not exist in the returned data. These attributes are provided when the place was created in the Twitter places database.

| Key | Description |
|---|---|
| street_address | |
| locality | the city the place is in |
| region | the administrative region the place is in |
| iso3 | the country code |
| postal_code | in the preferred local format for the place |
| phone | in the preferred local format for the place, include long distance code |
| twitter | twitter screen-name, without @ |
| url | official/canonical URL for place |
| app:id | An ID or comma separated list of IDs representing the place in the applications place database. |

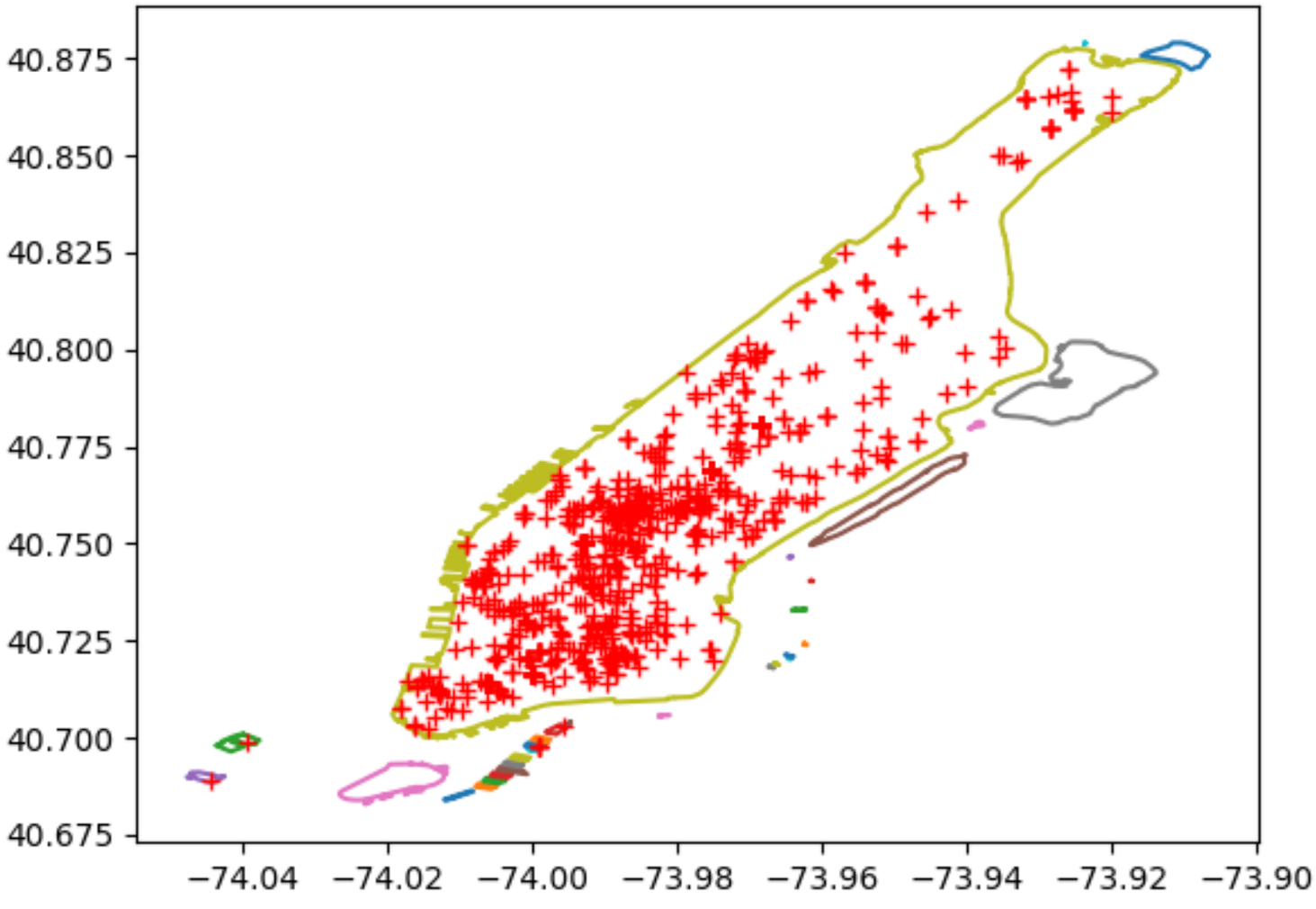
Keys can be no longer than 140 characters in length. Values are unicode strings and are restricted to 2000 characters.

# Filter points and places

```
import sys
import shapefile
from shapely.geometry import shape, Point
import gzip

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')

recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))

manhattan = shape(shp.shape(recordDict["Manhattan"]))
fp = gzip.open("Manhattan_places.json.gz", "w")

for line in gzip.open("NYC.json.gz"):
    try:
        tweet = eval(line.strip())
        point = None

        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])
        else:
            if "place" in tweet and tweet["place"]["bounding_box"] is not None:
                bbox = shape(tweet["place"]["bounding_box"])
                point = bbox.centroid

        if point is not None and manhattan.contains(point):
            fp.write(line)
    except:
        pass

fp.close()
```

shapefile_filter_places.py

# Filter points and places



# Filter points and places

```
import sys
import gzip
import numpy as np
import shapefile
from shapely.geometry import shape, Point
import matplotlib.pyplot as plt

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')
recordDict = dict(zip([record[1] for record in shp.iterRecords()],
range(shp.numRecords)))

manhattan = shp.shape(recordDict["Manhattan"])

points = np.array(manhattan.points)
parts = manhattan.parts
parts.append(len(manhattan.points))

for i in range(len(parts)-1):
      plt.plot(points.T[0][parts[i]:parts[i+1]], points.T[1]
[parts[i]:parts[i+1]])

points_X = []
points_Y = []

for line in gzip.open(sys.argv[1]):
    try:
        tweet = eval(line.strip())
        point = None

        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])
        else:
            if "place" in tweet and tweet["place"]["bounding_box"] is not
None:
                  bbox = shape(tweet["place"]["bounding_box"])
                  point = bbox.centroid

        if point is not None:
            points_X.append(point.x)
            points_Y.append(point.y)
    except:
        pass

plt.plot(points_X, points_Y, 'r+')

plt.savefig(sys.argv[1] + '.png')
```

@bgoncalves

plot_shapefile_points.py

# Calculating distances

https://en.wikipedia.org/wiki/Vincenty%27s_formulae
https://en.wikipedia.org/wiki/Great-circle_distance
https://en.wikipedia.org/wiki/Haversine_formula

- Earlier we saw how to obtain the distance between two points using the Google Maps API.

- But what is the shortest distance between two **arbitrary** points on the surface of the Earth?

- This depends strongly on our model of the Earth:

  - Great Circle - Assumes that the Earth is a perfect sphere of a given radius

    - Usually uses the Haversine formula $\Delta\sigma = 2\arcsin\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}$

  - Vincenty - Uses a (more) accurate ellipsoid model of the Earth

# geopy

https://geopy.readthedocs.io/en/1.10.0/

- geopy provides two different types of functionality
    - geopy.geocoders - a unified interface to several geocoding services (Google Maps, Nominatim, Yahoo, Bing, etc…)
    - geopy.distance - state of the art distance calculations
- We will focus just on the distance module:
    - distance.vincenty(p1, p2) - Calculate the vincenty distance between p1 and p2
    - distance.great_circle(p1, p2) - Calculate the great circle distance between p1 and p2
    - distance.distance(p1, p2) - an alias to distance.vincenty to be used as a default.

# geopy

https://geopy.readthedocs.io/en/1.10.0/

- all distance functions return a Distance object.
- the Distance object provides properties that represent the result in different units:
  - .km/.kilometers
  - .m/.meters
  - .mi/.miles
  - .ft/.feet
  - .nm/.nautical
- it also allows us to recalculate the result using different ellipsoids:
  - .set_ellipsoid('elliipsoid')
  - by default WGS-84 is used.

```
                 model             major (km)   minor (km)     flattening
ELLIPSOIDS = {'WGS-84':        (6378.137,    6356.7523142,  1 /
              'GRS-80':        (6378.137,    6356.7523141,  1 /
              'Airy (1830)':   (6377.563396, 6356.256909,   1 /
              'Intl 1924':     (6378.388,    6356.911946,   1 / 297.0),
              'Clarke (1880)': (6378.249145, 6356.51486955, 1 / 293.465),
              'GRS-67':        (6378.1600,   6356.774719,   1 / 298.25),
              }
```

# geopy

https://geopy.readthedocs.io/en/1.10.0/

```
from geopy import distance

p1 = (41.49008, -71.312796)
p2 = (41.499498, -81.695391)

dist_vincenty = distance.vincenty(p1,
p2).meters
dist_great = distance.great_circle(p1,
p2).meters

print("Vincenty:", dist_vincenty)
print("Great Circles:", dist_great)
```

geopy_distance.py

# GIS Data Systems

## Vector

- Data types:
  - Points
  - Lines
  - Polygons
- Discrete shapes and boundaries
- Spatial, Database and Network analysis
- Shapefile, GeoJSON, GML, etc…

## Raster

- Data types:
  - Cells
  - Pixels
  - Elements
- Dense data, Continuous surfaces
- Spatial Analysis and modeling
- GeoTIFF, ASC, JPEG2000, etc…

# ASCII Grid

- Perhaps the simples raster file
- ASCII text based
- A small header

```
ncols         246
nrows         119
xllcorner     -126.500000000000
yllcorner     22.750000000000
cellsize      0.250000000000
NODATA_value  -9999
```

- Followed by rows of numbers
- Very convenient to Read and Write

geofiles/US_pop.asc

# ASCII Gri

```python
import numpy as np
import matplotlib.pyplot as plt

def map_points(xllcorner, yllcorner, cellsize, nrows, x, y):
    x = int((x-xllcorner)/cellsize)
    y = (nrows-1)-int((y-yllcorner)/cellsize)

    return x, y

fp = open("geofiles/US_pop.asc")
ncols, count = fp.readline().split()
ncols = int(count)
nrows, count = fp.readline().split()
nrows = int(count)
xllcorner, value = fp.readline().split()
xllcorner = float(value)
yllcorner, value = fp.readline().split()
yllcorner = float(value)
cellsize, value = fp.readline().split()
cellsize = float(value)

NODATA_value, value = fp.readline().split()
NODATA_value = float(value)

data = []
for line in fp:
    fields = line.strip().split()
    data.append([float(field) for field in fields])

data = np.array(data)
data[data==NODATA_value] = 0

x = -74.243251
y = 40.730503

coord_x, coord_y = map_points(xllcorner, yllcorner, cellsize, nrows, x, y)
print(data[coord_y, coord_x])
```

asc_load.py

# ASCII Gri

```python
import numpy as np
import matplotlib.pyplot as plt

def map_points(xllcorner, yllcorner, cellsize, nrows, x, y):
    x = int((x-xllcorner)/cellsize)
    y = (nrows-1)-int((y-yllcorner)/cellsize)

    return x, y

fp = open("geofiles/US_pop.asc")
ncols, count = fp.readline().split()
ncols = int(count)
nrows, count = fp.readline().split()
nrows = int(count)
xllcorner, value = fp.readline().split()
xllcorner = float(value)
yllcorner, value = fp.readline().split()
yllcorner = float(value)
cellsize, value = fp.readline().split()
cellsize = float(value)

NODATA_value, value = fp.readline().split()
NODATA_value = float(value)

data = []
for line in fp:
    fields = line.strip().split()
    data.append([float(field) for field in fields])

data = np.array(data)
data[data==NODATA_value] = 0

x = -74.243251
y = 40.730503

coord_x, coord_y = map_points(xllcorner, yllcorner, cellsize, nrows, x, y)
print(data[coord_y, coord_x])
```

asc_load.py

# ASCII Grid


asc_load.py

# ASCII Grid

- This type of grid is a very convenient way to aggregate spatial data.
- Simply map **lat, lon** pairs to matrix entries and then increment the values
- All we need is to define the **bbox** we are interested in, and the size of each cell and create a matrix with that shape.

```
import numpy as np
from shapely.geometry import shape, Point
import shapefile

shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')
recordDict = dict(zip([record[1] for record in shp.iterRecords()],
range(shp.numRecords)))

manhattan = shp.shape(recordDict["Manhattan"])

xllcorner, yllcorner, xurcorner, yurcorner = manhattan.bbox

cellsize = 0.01

ncols = int((xurcorner-xllcorner)/cellsize)
nrows = int((yurcorner-yllcorner)/cellsize)

data = np.zeros((nrows, ncols), dtype='int')

print(data.shape)
```

asc_generate_grid.py

# Aggregate

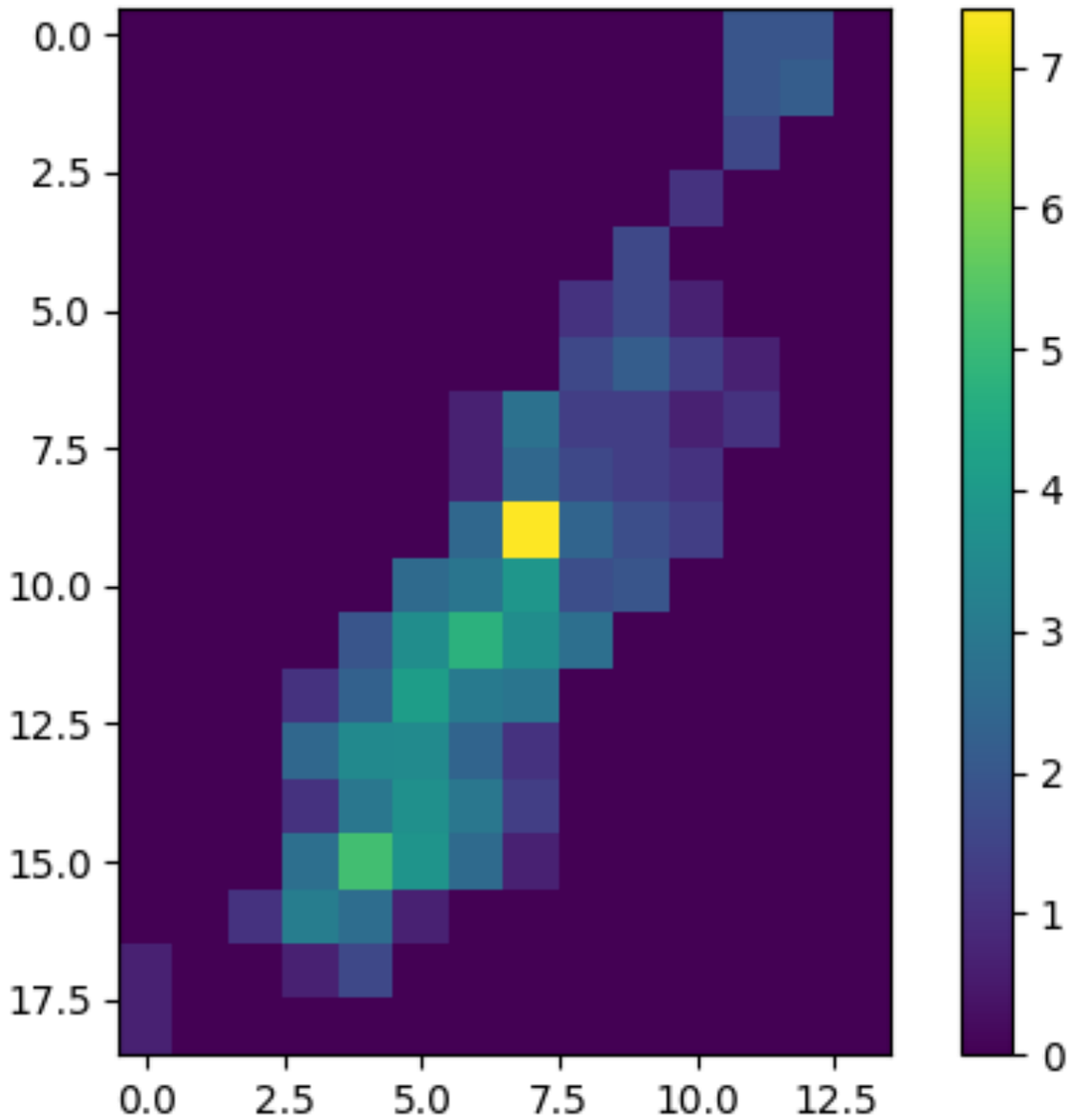
```
import sys
import numpy as np
import shapefile
from shapely.geometry import shape, Point
import matplotlib.pyplot as plt
import gzip

def map_points(xllcorner, yllcorner, cellsize, nrows, x, y):
    x = int((x-xllcorner)/cellsize)
    y = (nrows-1)-int((y-yllcorner)/cellsize)

    return x, y

def save_asc(data, xllcorner, yllcorner, cellsize, filename):
    fp = open(filename, "w")

    nrows, ncols = data.shape

    print("ncols", ncols, file=fp)
    print("nrows", nrows, file=fp)
    print("xllcorner", xllcorner, file=fp)
    print("yllcorner", yllcorner, file=fp)
    print("cellsize", cellsize, file=fp)
    print("NODATA_value", "-9999", file=fp)

    for i in range(nrows):
        for j in range(ncols):
            print(("%u " % data[i, j]), end="", file=fp)

        print("\n", end="", file=fp)

    fp.close()
```

shapefile_filter_aggregate.py

```
shp = shapefile.Reader('geofiles/nybb_15c/nybb_wgs84.shp')
recordDict = dict(zip([record[1] for record in shp.iterRecords()], range(shp.numRecords)))
manhattan = shape(shp.shape(recordDict["Manhattan"]))

xllcorner, yllcorner, xurcorner, yurcorner = manhattan.bounds
cellsize = 0.01

ncols = int((xurcorner-xllcorner)/cellsize)
nrows = int((yurcorner-yllcorner)/cellsize)

data = np.zeros((nrows, ncols), dtype='int')

for line in gzip.open("NYC.json.gz"):
    try:
        tweet = eval(line.strip())
        point = None

        if "coordinates" in tweet and tweet["coordinates"] is not None:
            point = Point(tweet["coordinates"]["coordinates"])
        else:
            if "place" in tweet and tweet["place"]["bounding_box"] is not None:
                bbox = shape(tweet["place"]["bounding_box"])
                point = bbox.centroid

        if point is not None and manhattan.contains(point):
            coord_x, coord_y = map_points(xllcorner, yllcorner, cellsize, nrows, point.x, point.y)
            data[coord_y, coord_x] += 1

    except:
        pass


save_asc(data, xllcorner, yllcorner, cellsize, "Manhattan.asc")

plt.imshow(np.log(data+1))
plt.colorbar()
plt.savefig('Manhattan_cells.png')
```

shapefile_filter_aggregate.py

# Aggregate

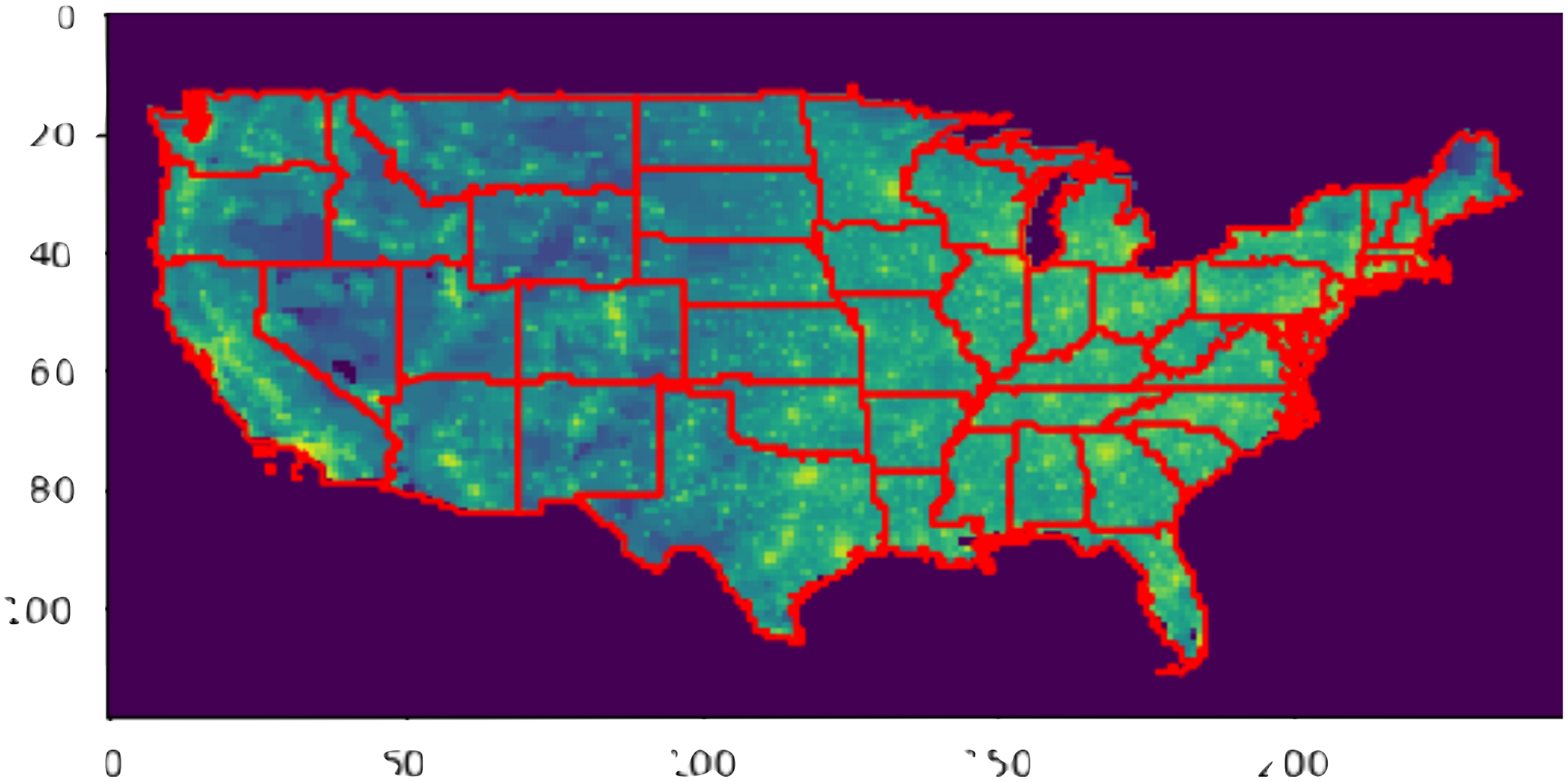


# Overlap a raster and a shapefile

```
import numpy as np
import matplotlib.pyplot as plt
import shapefile

(…)

fig, ax = plt.subplots(1,1)
data, xllcorner, yllcorner, cellsize  = load_asc('geofiles/US_pop.asc')
ax.imshow(np.log(data+1))

shp = shapefile.Reader('geofiles/48States/48States.shp')

pos = None
count = 0
for shape in shp.iterShapes():
        points = np.array(shape.points)
        parts = shape.parts
        parts.append(len(shape.points))

        for i in range(len(parts)-1):
            positions = []

            for j in range(parts[i+1]-parts[i]):
                x_orig = points.T[0][parts[i]+j]
                y_orig = points.T[1][parts[i]+j]
                x, y = map_points(xllcorner, yllcorner, cellsize, data.shape[0], x_orig, y_orig)
                positions.append([x, y])

            positions = np.array(positions)

            ax.plot(positions.T[0], positions.T[1], 'r-')


fig.savefig('US overlap.png')
```

matplotlib_overlap.py

# Overlap a raster and a shapefile




*@bgoncalves*

# Matplotlib Basemap

# Basemap

- The Basemap module is the workhorse and returns a Basemap object when instantiated.
- The Basemap object has many useful methods to assist in drawing a map:
  - .drawcoastlines() - To draw the coastlines of continents
  - .drawmapboundary() - To draw the boundary of the map
  - .fillcontinents() - add color to the continents
  - etc…
- The constructor for Basemap can take many different arguments to be able to handle different projections, but it defaults to the Plate Carrée projection centered at (0, 0)
- The minimal map is simply:

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

map = Basemap()
map.drawcoastlines()
plt.savefig('basemap_demo.png')
```

- Please note that with the .drawcoastlines() call nothing is plotted as our map has no content.

# Basemap

- We can also visualize just specific regions by setting the bbox and center coordinates by setting

llcrnrlon, llcrnrlat, urcrnrlon, urcrnrlat

- And

lat_0, lon_0

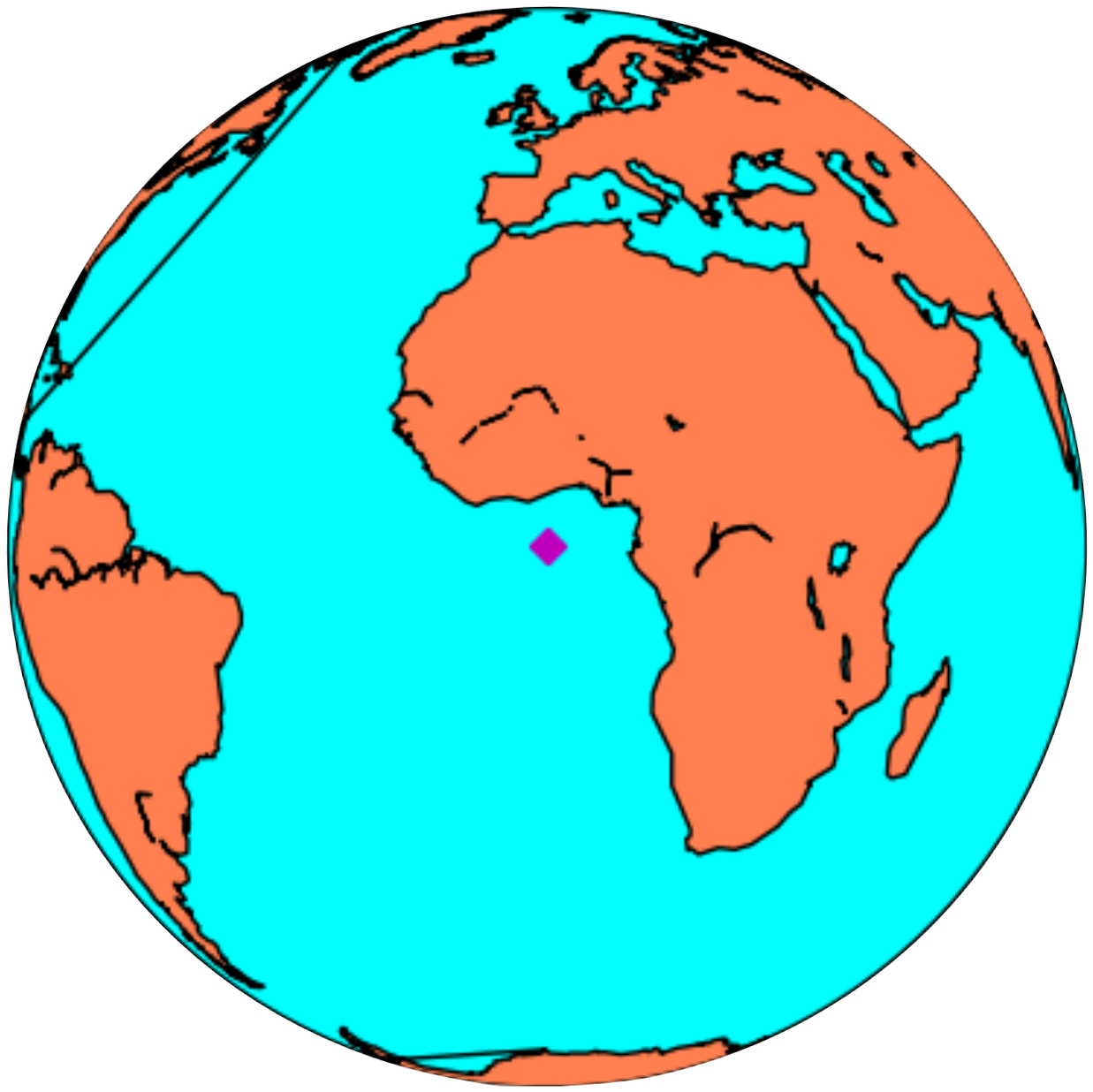
- Respectively.
- We can convert arbitrary lat, lon values to map coordinates by calling the map() object directly.
- After we obtain the map coordinates we can add them to the map by calling the .plot(x, y) method of the map object.

# Basemap Example

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

map = Basemap(projection='ortho', lat_0=0, lon_0=0)

map.drawmapboundary(fill_color='aqua')
map.fillcontinents(color='coral',lake_color='aqua')
map.drawcoastlines()

x, y = map(0, 0)

map.plot(x, y, marker='D',color='m')

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

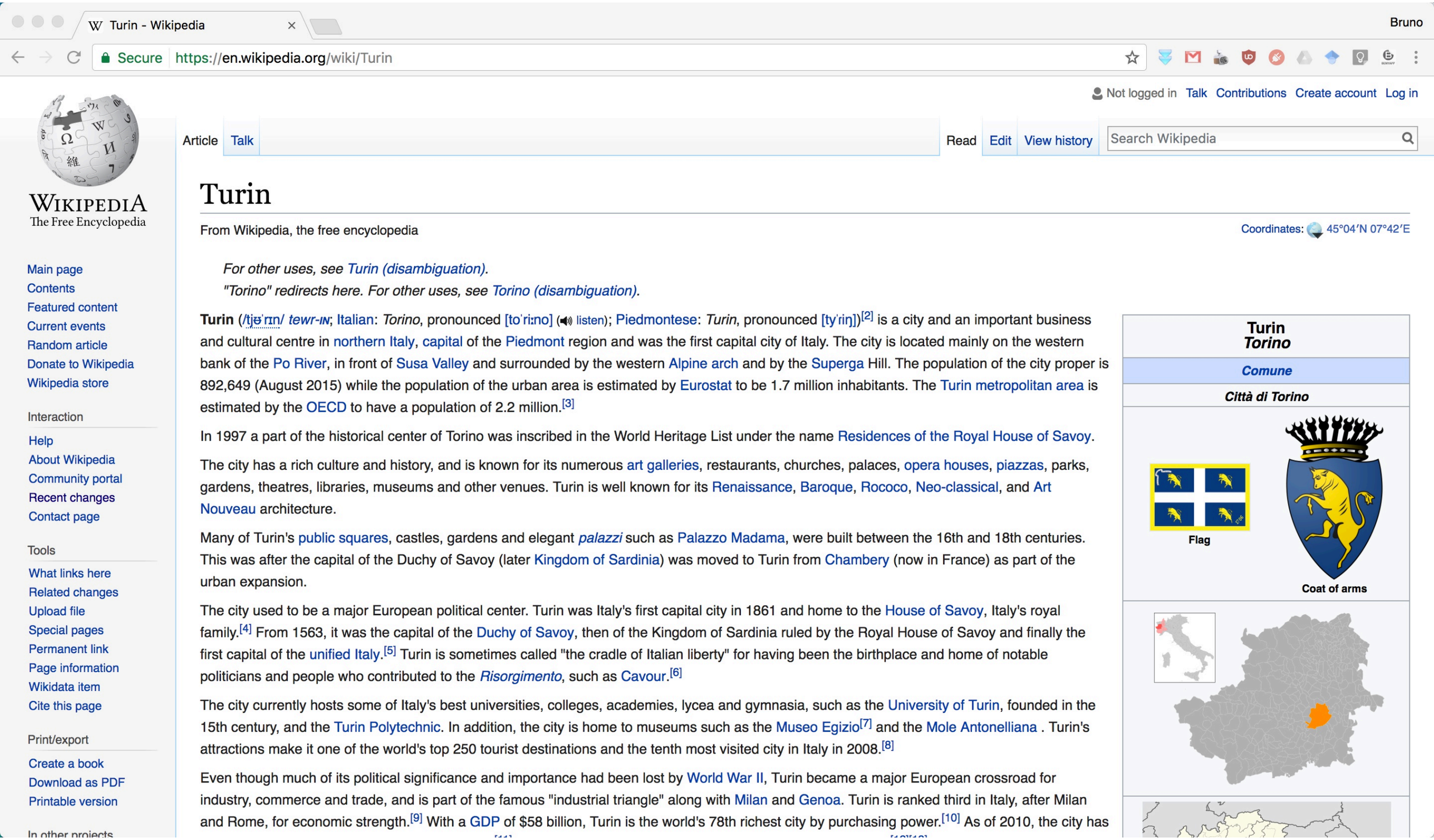
basemap_example.py

# Basemap Example

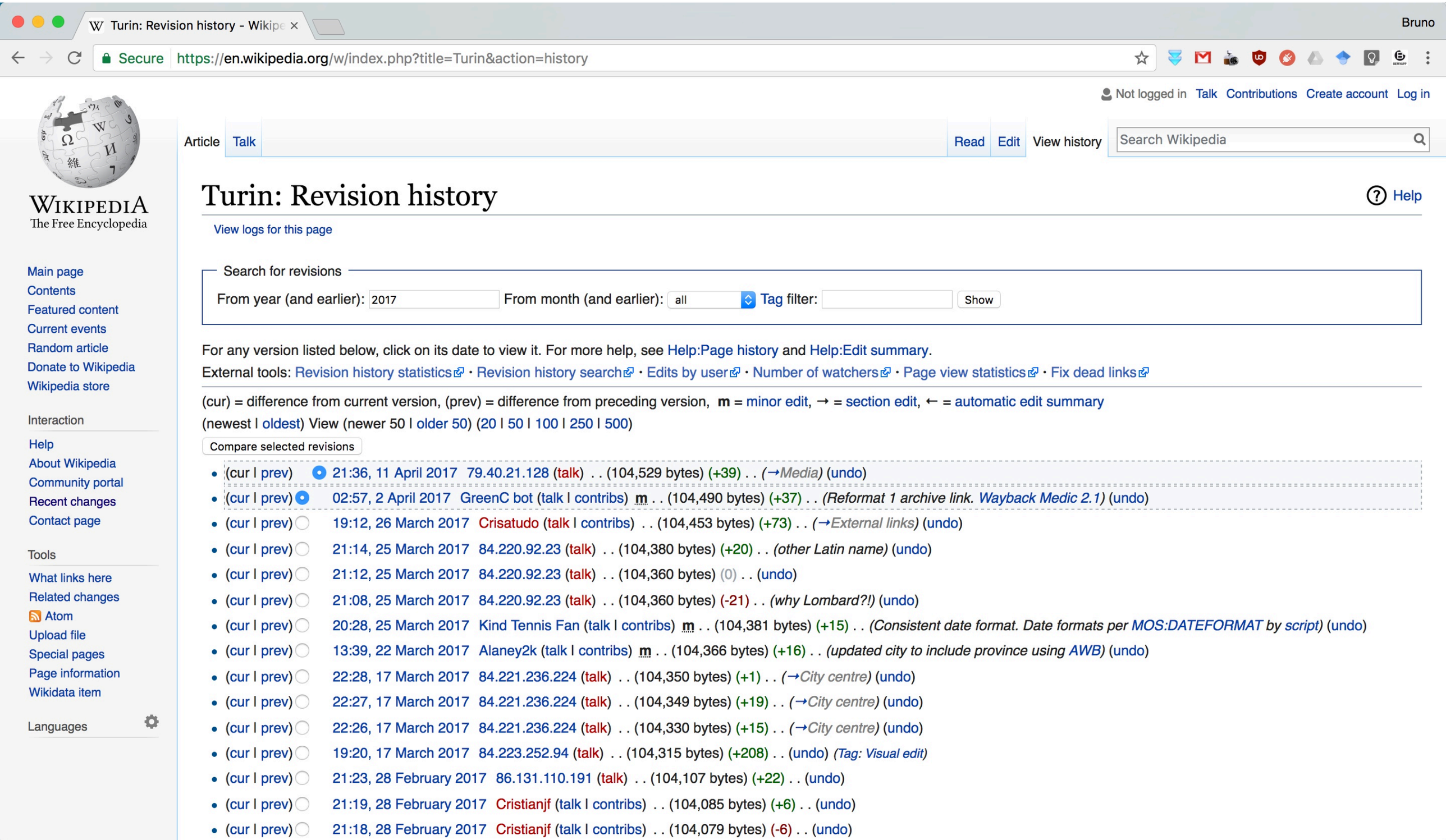


Wikipedia

# Wikipedia


Turin - Wikipedia
Bruno
Secure https://en.wikipedia.org/wiki/Turin
Not logged in Talk Contributions Create account Log in
Article Talk Read Edit 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
Wikidata item
Cite this page
Print/export
Create a book
Download as PDF
Printable version
In other projects
Turin
From Wikipedia, the free encyclopedia
Coordinates: 45°04′N 07°42′E
For other uses, see Turin (disambiguation).
"Torino" redirects here. For other uses, see Torino (disambiguation).
Turin (/tjʊˈrɪn/ tewr-in; Italian: Torino, pronounced [toˈriːno] (listen); Piedmontese: Turin, pronounced [tyˈrin])[2] is a city and an important business and cultural centre in northern Italy, capital of the Piedmont region and was the first capital city of Italy. The city is located mainly on the western bank of the Po River, in front of Susa Valley and surrounded by the western Alpine arch and by the Superga Hill. The population of the city proper is 892,649 (August 2015) while the population of the urban area is estimated by Eurostat to be 1.7 million inhabitants. The Turin metropolitan area is estimated by the OECD to have a population of 2.2 million.[3]
In 1997 a part of the historical center of Torino was inscribed in the World Heritage List under the name Residences of the Royal House of Savoy.
The city has a rich culture and history, and is known for its numerous art galleries, restaurants, churches, palaces, opera houses, piazzas, parks, gardens, theatres, libraries, museums and other venues. Turin is well known for its Renaissance, Baroque, Rococo, Neo-classical, and Art Nouveau architecture.
Many of Turin's public squares, castles, gardens and elegant palazzi such as Palazzo Madama, were built between the 16th and 18th centuries. This was after the capital of the Duchy of Savoy (later Kingdom of Sardinia) was moved to Turin from Chambery (now in France) as part of the urban expansion.
The city used to be a major European political center. Turin was Italy's first capital city in 1861 and home to the House of Savoy, Italy's royal family.[4] From 1563, it was the capital of the Duchy of Savoy, then of the Kingdom of Sardinia ruled by the Royal House of Savoy and finally the first capital of the unified Italy.[5] Turin is sometimes called "the cradle of Italian liberty" for having been the birthplace and home of notable politicians and people who contributed to the Risorgimento, such as Cavour.[6]
The city currently hosts some of Italy's best universities, colleges, academies, lycea and gymnasia, such as the University of Turin, founded in the 15th century, and the Turin Polytechnic. In addition, the city is home to museums such as the Museo Egizio[7] and the Mole Antonelliana . Turin's attractions make it one of the world's top 250 tourist destinations and the tenth most visited city in Italy in 2008.[8]
Even though much of its political significance and importance had been lost by World War II, Turin became a major European crossroad for industry, commerce and trade, and is part of the famous "industrial triangle" along with Milan and Genoa. Turin is ranked third in Italy, after Milan and Rome, for economic strength.[9] With a GDP of $58 billion, Turin is the world's 78th richest city by purchasing power.[10] As of 2010, the city has
Turin
Torino
Comune
Città di Torino
Flag
Coat of arms

# Wikipedia


Turin: Revision history - Wikipe
Bruno
Secure https://en.wikipedia.org/w/index.php?title=Turin&action=history
Not logged in Talk Contributions Create account Log in
Article Talk Read Edit View history Search Wikipedia
WIKIPEDIA
The Free Encyclopedia
Turin: Revision history
Help
View logs for this page
Search for revisions
From year (and earlier): 2017 From month (and earlier): all Tag filter: Show
For any version listed below, click on its date to view it. For more help, see Help:Page history and Help:Edit summary.
External tools: Revision history statistics · Revision history search · Edits by user · Number of watchers · Page view statistics · Fix dead links
(cur) = difference from current version, (prev) = difference from preceding version, m = minor edit, → = section edit, ← = automatic edit summary
(newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)
Compare selected revisions
(cur | prev) 21:36, 11 April 2017 79.40.21.128 (talk) . . (104,529 bytes) (+39) . . (→Media) (undo)
(cur | prev) 02:57, 2 April 2017 GreenC bot (talk | contribs) m . . (104,490 bytes) (+37) . . (Reformat 1 archive link. Wayback Medic 2.1) (undo)
(cur | prev) 19:12, 26 March 2017 Crisatudo (talk | contribs) . . (104,453 bytes) (+73) . . (→External links) (undo)
(cur | prev) 21:14, 25 March 2017 84.220.92.23 (talk) . . (104,380 bytes) (+20) . . (other Latin name) (undo)
(cur | prev) 21:12, 25 March 2017 84.220.92.23 (talk) . . (104,360 bytes) (0) . . (undo)
(cur | prev) 21:08, 25 March 2017 84.220.92.23 (talk) . . (104,360 bytes) (-21) . . (why Lombard?!) (undo)
(cur | prev) 20:28, 25 March 2017 Kind Tennis Fan (talk | contribs) m . . (104,381 bytes) (+15) . . (Consistent date format. Date formats per MOS:DATEFORMAT by script) (undo)
(cur | prev) 13:39, 22 March 2017 Alaney2k (talk | contribs) m . . (104,366 bytes) (+16) . . (updated city to include province using AWB) (undo)
(cur | prev) 22:28, 17 March 2017 84.221.236.224 (talk) . . (104,350 bytes) (+1) . . (→City centre) (undo)
(cur | prev) 22:27, 17 March 2017 84.221.236.224 (talk) . . (104,349 bytes) (+19) . . (→City centre) (undo)
(cur | prev) 22:26, 17 March 2017 84.221.236.224 (talk) . . (104,330 bytes) (+15) . . (→City centre) (undo)
(cur | prev) 19:20, 17 March 2017 84.223.252.94 (talk) . . (104,315 bytes) (+208) . . (undo) (Tag: Visual edit)
(cur | prev) 21:23, 28 February 2017 86.131.110.191 (talk) . . (104,107 bytes) (+22) . . (undo)
(cur | prev) 21:19, 28 February 2017 Cristianjf (talk | contribs) . . (104,085 bytes) (+6) . . (undo)
(cur | prev) 21:18, 28 February 2017 Cristianjf (talk | contribs) . . (104,079 bytes) (-6) . . (undo)
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
Atom
Upload file
Special pages
Page information
Wikidata item
Languages

# 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 ]

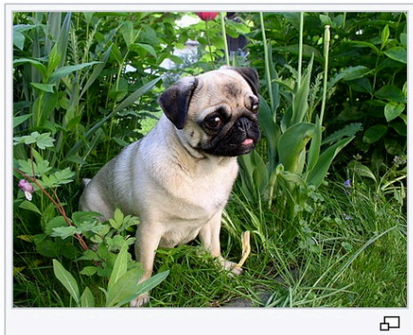
Politics

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.)

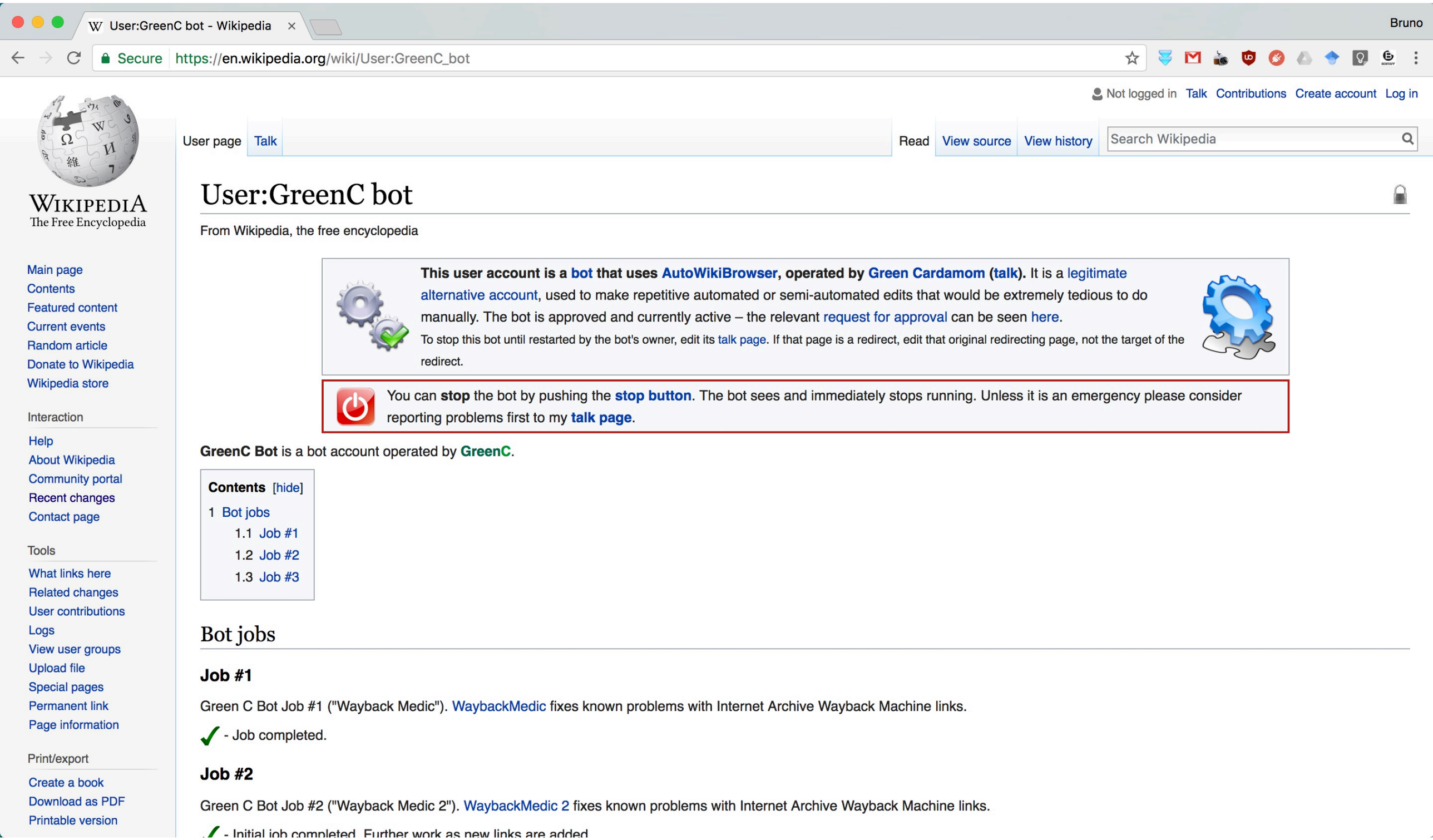
Restaurants and Cuisine

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

Talk:Turin - Wikipedia

Bruno

Secure https://en.wikipedia.org/wiki/Talk:Turin

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Article Talk Read Edit New section View history Search Wikipedia

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:Turin

From Wikipedia, the free encyclopedia

**Turin** has been listed as a level-4 ***vital article*** in Geography. If you can improve it, please do. This article has been rated as C-Class.

**This article is of interest to the following WikiProjects:** [hide]

| | |
|---|---|
| **WikiProject Italy** | **(Rated C-class, Top-importance)** [show] |
| **WikiProject Cities** | **(Rated C-class, High-importance)** [show] |
| **WikiProject Olympics / Paralympics** | **(Rated C-class, Mid-importance)** [show] |
| **Wikipedia Version 1.0 Editorial Team / v0.5 / Vital** | **(Rated C-class)** [show] |

This article is/was the subject of a Wiki Education Foundation-supported course assignment. Further details are available on the course page. Assigned peer reviews: NicholasKZalewski.

**Contents** [hide]



This article contains a translation of Torino from it.wikipedia.

# 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?

# 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…

# 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

```
<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

# expat

- To 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

# 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

# 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

# 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

# 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 is done in two parts:

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

    - Use the 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