What is an API and How to Access it?
WSU Python Working Group

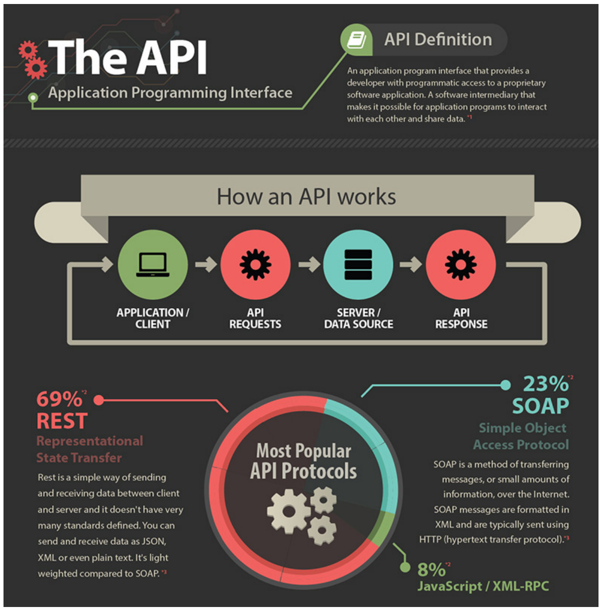
Joseph Navelski

WSU

November 3, 2021

What is an API and how does it work?

An API is an Application Programming Interface is a software intermediary that allows two applications to talk to each other. Each time you use an app like Twitter, send an instant message, or check the weather on your phone, you're using an API.



An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data.[1]

Rest is a simple way of sending and receiving data between client and server and it doesn't have very many standards defined. You can send and receive data as JSON, XML or even plain text. It's light weighted compared to SOAP.[3]

SOAP is a method of transferring messages, or small amounts of information, over the Internet. SOAP messages are formatted in XML and are typically sent using HTTP (hypertext transfer protocol).[3]

https://www.govtech.com/archive/whats-an-api-and-why-do-you-need-one.html

How do you access it?

1) Find the API you would like to access. Some may involve you registering with that API to get access (i.e. Twitter and US Census Bureau).

https://www.census.gov/data/developers/data-sets.html

2) Find either a package that people have written to access that specific API or find a package that will allow you to raw code an API request.

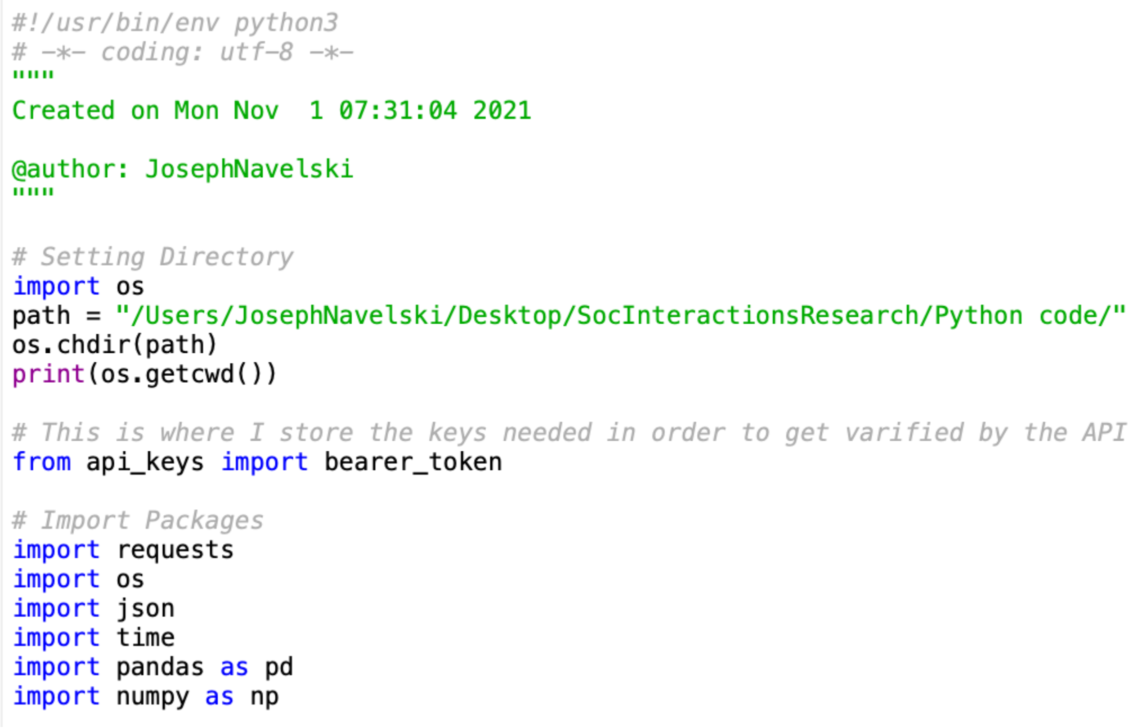
Examples include:

R: httr, rtweet, and tidycensus

Python: requests, tweepy (for V1), and censusdata

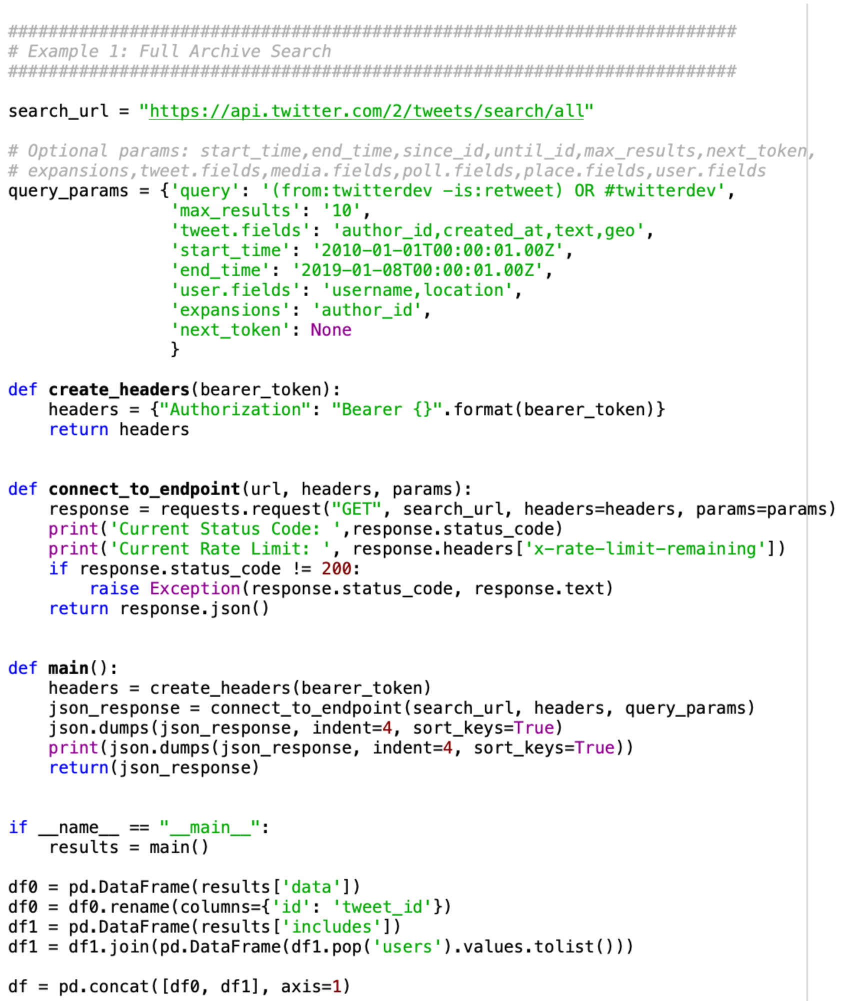
3) Develop a query using the API's documentation (usually written in HTML), send a request, enjoy the data!

Example Code: Packages Needed



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Nov  1 07:31:04 2021

@author: JosephNavelski
"""

# Setting Directory
import os
path = "/Users/JosephNavelski/Desktop/SocInteractionsResearch/Python code/"
os.chdir(path)
print(os.getcwd())

# This is where I store the keys needed in order to get varified by the API
from api_keys import bearer_token

# Import Packages
import requests
import os
import json
import time
import pandas as pd
import numpy as np
```

Example Code: Python Script for Twitter API



```
########################################################################
# Example 1: Full Archive Search
########################################################################

search_url = "https://api.twitter.com/2/tweets/search/all"

# Optional params: start_time,end_time,since_id,until_id,max_results,next_token,
# expansions,tweet.fields,media.fields,poll.fields,place.fields,user.fields
query_params = {'query': '(from:twitterdev -is:retweet) OR #twitterdev',
                'max_results': '10',
                'tweet.fields': 'author_id,created_at,text,geo',
                'start_time': '2010-01-01T00:00:01.00Z',
                'end_time': '2019-01-08T00:00:01.00Z',
                'user.fields': 'username,location',
                'expansions': 'author_id',
                'next_token': None
                }

def create_headers(bearer_token):
    headers = {"Authorization": "Bearer {}".format(bearer_token)}
    return headers


def connect_to_endpoint(url, headers, params):
    response = requests.request("GET", search_url, headers=headers, params=params)
    print('Current Status Code: ',response.status_code)
    print('Current Rate Limit: ', response.headers['x-rate-limit-remaining'])
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    return response.json()


def main():
    headers = create_headers(bearer_token)
    json_response = connect_to_endpoint(search_url, headers, query_params)
    json.dumps(json_response, indent=4, sort_keys=True)
    print(json.dumps(json_response, indent=4, sort_keys=True))
    return(json_response)


if __name__ == "__main__":
    results = main()

df0 = pd.DataFrame(results['data'])
df0 = df0.rename(columns={'id': 'tweet_id'})
df1 = pd.DataFrame(results['includes'])
df1 = df1.join(pd.DataFrame(df1.pop('users').values.tolist()))

df = pd.concat([df0, df1], axis=1)
```