```python
import os
import datetime
import json
import twython
import json
```

```python
import pandas as pd
import numpy as np
```

```python
from twython import Twython
from contextlib import suppress
from requests_oauthlib import OAuth1Session
from apscheduler.schedulers.background import BackgroundScheduler as Scheduler
```

```python
# Enter your keys/secrets as strings in the following fields
credentials = {}
credentials['CONSUMER_KEY'] =
credentials['CONSUMER_SECRET'] =
credentials['ACCESS_KEY'] =
credentials['ACCESS_SECRET'] =

# Save the credentials object to file
with open("data/twitter_credentials.json", "w") as file:
    json.dump(credentials, file)
```

```python
# Load credentials from json file
with open("data/twitter_credentials.json", "r") as file:
    creds = json.load(file)

# Instantiate an object
python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])
```

```python
# getting the header for our dataframe so we have something to append to
startdf = pd.read_csv('df')
# when our dataframe is created the index is unnamed, when you export
# it is renamed so we need to fix it so our df realizes it's the same
startdf = startdf.head(0).rename(columns={'Unnamed: 0':''})
#assign id as most recent tweet id to  so that we get everything after
id_ = python_tweets.search(**{'q': 'RT', 'result_type':
                              'recent', 'count': 2})['statuses'][0]['id']
```

```python
def gather_tweets(q):
    # import global variables we will be using this function to alter
    global id_
    global startdf
    # Create our query
    query = {'q': q,
            'result_type': 'recent',
            'count': 100,
            'lang': 'en',
            'max_id': id_,
            'tweet_mode' : 'extend',
            'entities': {
                "hashtags": [],
                "symbols": [],
                "user_mentions": []
            }
    }
    #build our query into a dictionary to easily turn into dataframe
    dict_ = {'user': [], 'user_id':[], 'post_id': [], 'text': [],
             'favorite_count': [], 'hashtags':[], 'symbols':[],
             'user_mentions':[], 'retweet_count':[]}
    for status in python_tweets.search(**query)['statuses']:
        dict_['user'].append(status['user']['screen_name'])
        dict_['user_id'].append(status['user']['id'])
        dict_['text'].append(status['text'])
        dict_['hashtags'].append(status['entities']['hashtags'])
        dict_['symbols'].append(status['entities']['symbols'])
        dict_['user_mentions'].append(status['entities']['user_mentions'])
        dict_['favorite_count'].append(status['favorite_count'])
        dict_['retweet_count'].append(status['retweet_count'])
        dict_['post_id'].append(status['id'])
    # put data in a DataFrame to work with it easier
    df = pd.DataFrame(dict_)
    #removing our vairables stuck in dictionaries within our
    #dataframe and give them their own columns
    df['mentions'] = ' '
    df['hashtag'] = ' '
    for i in range(len(df)):
        for j in range(len(df.at[i, 'user_mentions'])):
            try:
                df.at[i, 'mentions'] = str(df.at[i, 'mentions'] ,
                 ' ' , df.at[i, 'user_mentions'][j]['id'])
            except:
                df.at[i, 'mentions'] = str(df.at[i, 'mentions'])
        for k in range(len(df.at[i, 'hashtags'])):
            try:
                df.at[i, 'hashtag'] = str(df.at[i, 'hashtag']),
                ' ' , str(df.at[i, 'hashtags'][k]['text'])
            except:
                df.at[i, 'hashtag'] = str(df.at[i, 'hashtag'])
    # get rid of columns that we took everything we need from
    df = df.drop(['user_mentions', 'hashtags'], 1)
    startdf = startdf.append(df)
    id_ = None
    # get oldest id_ (lowest number) and subtract one so
    # that that you don't reinclude it in the next search reset
    #  id  to be oldest
    #  id_ to be oldest
    try:
        id_ = df['post_id'].sort_values(ascending=True)
        id_ = int(id_.to_frame().reset_index()['post_id'][0] - 1)
    except:
        id_ = python_tweets.search(**{'q': 'RT', 'result_type': 'recent',
                                      'count': 2})['statuses'][0]['id']
    return startdf, id_
```

```
# keeping a list of all of our used hashtags
#  'astronomy', 'Starship', 'mars', 'curiosityrover',
#  'oppertunityrover', 'starlink', 'falconheavy',
#  'sls','ESA', 'NASA', 'spacex', 'virgingalactic',
#  'virginorbit', 'JAXA', 'Roscosmos', 'artemis',
#  'starliner', 'blueorigin', 'spacetravel', 'marswebcam',
#  'falcon9', 'nasa_app', 'universe', 'cosmos',
#  'iss', 'climate', 'internationalspacestation', 'futurism',
#  'starliner', 'blueorigin', 'spacetravel','starlink', 'falconheavy',
#  'astronomy', 'mars', 'curiosityrover', 'oppertunityrover',
#  'NASA', 'spacex', 'virgingalactic', 'virginorbit', 'JAXA', 'areospace'
```

```
query_list = []
```

```
# eventual goal is to use those loop to do all of our searches
end_item = datetime.datetime.now()
id_ = python_tweets.search(**{'q': 'RT', 'result_type': 'recent',
                              'count': 2})['statuses'][0]['id']

for item in query_list:
    #reset id_ to be most recent tweet for each item
    id_ = python_tweets.search(**{'q': 'RT', 'result_type': 'recent',
                                  'count': 2})['statuses'][0]['id']
    # for each item, reset the start time to the end of the last item
    start_time = end_item + datetime.timedelta(seconds=1)
    # set the end time to be 90 min after you start, gathering 50
    #  tweets every 100 seconds for 2,700 tweets from each item in list
    end_item = end_item + datetime.timedelta(seconds=1500)
    #print start and end times so I know when the program
    # will be finished and where it should be at
    print(f'starting {item} time : {start_time.strftime("%H:%M:%S")},'+
          'ending time: {end_item.strftime("%H:%M:%S")}')
    # occassionally getting key/value errors of 0 that I cannot find the
    # cause of, but do not affect anything in how our dataframe
    # is made, I just don't want them printing
    with suppress(KeyError, ValueError):
        #initiate scheduler
        sch = Scheduler()
        #add our function and items, we're doing 50/10 seconds
        sch.add_job(myfn(q), 'interval', (item, startdf), seconds= 5,
                    start_date=start_time, end_date=end_item)
        #start scheduler
        sch.start()
```

```
starting marswebcam time : 22:21:02, ending time: 22:46:01
starting falcon9 time : 22:46:02, ending time: 23:11:01
starting nasa_app time : 23:11:02, ending time: 23:36:01
starting universe time : 23:36:02, ending time: 00:01:01
starting cosmos time : 00:01:02, ending time: 00:26:01
starting iss time : 00:26:02, ending time: 00:51:01
starting climate time : 00:51:02, ending time: 01:16:01
starting internationalspacestation time : 01:16:02, ending time: 01:41:01
starting futurism time : 01:41:02, ending time: 02:06:01
starting starliner time : 02:06:02, ending time: 02:31:01
starting blueorigin time : 02:31:02, ending time: 02:56:01
starting spacetravel time : 02:56:02, ending time: 03:21:01
starting astronomy time : 03:21:02, ending time: 03:46:01
starting mars time : 03:46:02, ending time: 04:11:01
starting curiosityrover time : 04:11:02, ending time: 04:36:01
starting oppertunityrover time : 04:36:02, ending time: 05:01:01
starting starlink time : 05:01:02, ending time: 05:26:01
starting falconheavy time : 05:26:02, ending time: 05:51:01
starting sls time : 05:51:02, ending time: 06:16:01
starting ESA time : 06:16:02, ending time: 06:41:01
starting NASA time : 06:41:02, ending time: 07:06:01
starting spacex time : 07:06:02, ending time: 07:31:01
starting virgingalactic time : 07:31:02, ending time: 07:56:01
starting virginorbit time : 07:56:02, ending time: 08:21:01
starting JAXA time : 08:21:02, ending time: 08:46:01
starting Roscosmos time : 08:46:02, ending time: 09:11:01
starting areospace time : 09:11:02, ending time: 09:36:01
```

```
startdf = startdf[['user', 'user_id', 'text',
                      'favorite_count', 'retweet_count',
                      'mentions', 'hashtag', 'post_id']]
```

```
# don't need these columns anymore
startdf = startdf.drop(['index', '', 'symbols'], 1)
```

```
# in order to add our strings for our hashtags and mentions together we
# had to make the entire column a string, sinlucding empty cells
# here we go through and replace empty cells with nan values so pandas
# will read them as being empty instead of a string
for i in range(len(startdf)):
    if startdf.at[i, 'mentions'] == ' ':
        startdf.at[i, 'mentions'] = np.nan
    else:
        pass
    if startdf.at[i, 'hashtag'] == ' ':
        startdf.at[i, 'hashtag'] = np.nan
    else:
        pass
```

```
startdf.to_csv('tweets.4.7')
```

✓ 0s completed at 12:35 AM