

# **USER MANUAL**

*Getwit*



Disusun oleh:

Ahmad Afiq Fitrah

Niken Dwi Wahyu Cahyani

Erwid Mustofa Jadied

## Daftar Isi

Daftar Isi .....	i
Daftar Gambar .....	i
1. Deskripsi .....	1
2. Cara Penggunaan .....	1
2.1 Persiapan .....	1
2.2 Generate Key dan Token .....	2
1.3 Running window.py .....	3
1.4 Pilih Endpoint .....	3
1.5 Masukkan Parameter .....	7
1.5 Generate hash value .....	9
1.6 Read documentation .....	10
3. Source Code .....	10
3.1 api_secret.py .....	10
3.2 model.py .....	10
3.3 function.py .....	20
3.4 window.py .....	52

## Daftar Gambar

Gambar 1 Perintah install getwit python package .....	1
Gambar 2 perintah clone git .....	1
Gambar 3 Token dan key disalin ke file api_secret.py .....	2
Gambar 4 Tampilan utama dari Getwit .....	2
Gambar 5 Tampilan menu endpoint .....	7
Gambar 6 tampilan frame fitur get data dari endpoint .....	8
Gambar 7 tampilan webpage authorize app .....	8
Gambar 8 tampilan setelah melakukan authorized app .....	9
Gambar 9 salin bagian oauth_varifier dari page sebelumnya ke terminal .....	9
Gambar 10 tampilan frame fitur get hash value .....	10

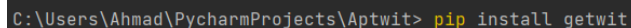
## 1. Deskripsi

Getwit adalah program komputer yang terhubung pada endpoint dengan metode 'GET' pada Twitter API v2. Aplikasi ini bertujuan untuk mengakuisisi data user pada aplikasi Twitter untuk keperluan penyidikan forensik digital. Aplikasi ini dilengkapi dengan fitur perhitungan nilai hash MD5 dan SHA256 untuk data yang diakuisisi. Library aplikasi dibangun dengan menggunakan bahasa pemrograman python. Getwit memiliki 3 modul utama yaitu function.py, model.py dan api\_secret.py. Selain itu, terdapat pula module untuk menampilkan UI dari aplikasi dengan nama window.py. Getwit terhubung pada 32 endpoint yang ada di Twitter API v2. Sebelum menggunakan library tersebut, diperlukan proses generate key dan token, serta pengaturan user authentication di laman developer portal dari Twitter Developer. Seluruh endpoint yang ada, telah diuji coba menggunakan dua tipe autentikasi, yaitu OAuth1.0 User-context dan OAuth2.0 App-Only dengan level akses ialah 'elevated'.

## 2. Cara Penggunaan

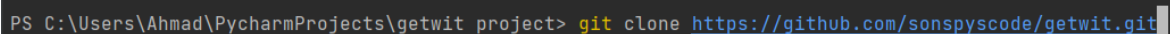
### 2.1 Persiapan

Untuk mengunduh library, cukup mengakses di repository 'getwit' di akun Github 'sonspyscode' dan melakukan pengunduhan. Selain itu, untuk mengunduh library tersebut dapat dilakukan melalui terminal dari text editor dengan perintah 'pip install getwit'. Pastikan telah mengunduh setiap requirement yang dibutuhkan pada file dengan nama requirement.txt. Selain itu, melakukan clone pada repository 'getwit app' dengan command 'git clone <https://github.com/sonspyscode/getwit.git>'. Berikut tampilan pada terminal untuk install package dan clone repository pada gambar 1 dan gambar 2.



```
C:\Users\Ahmad\PycharmProjects\Aptwit> pip install getwit
```

Gambar 1 Perintah install getwit python package

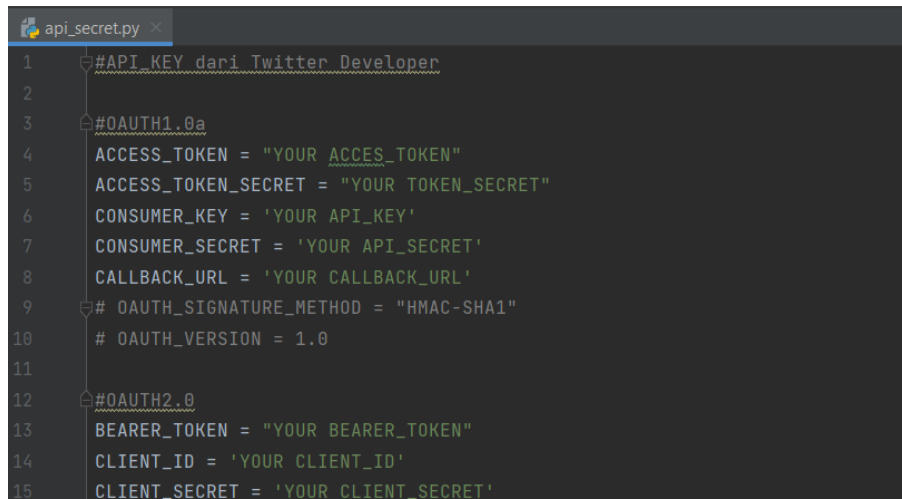


```
PS C:\Users\Ahmad\PycharmProjects\getwit project> git clone https://github.com/sonspyscode/getwit.git
```

Gambar 2 perintah clone git

## 2.2 Generate Key dan Token

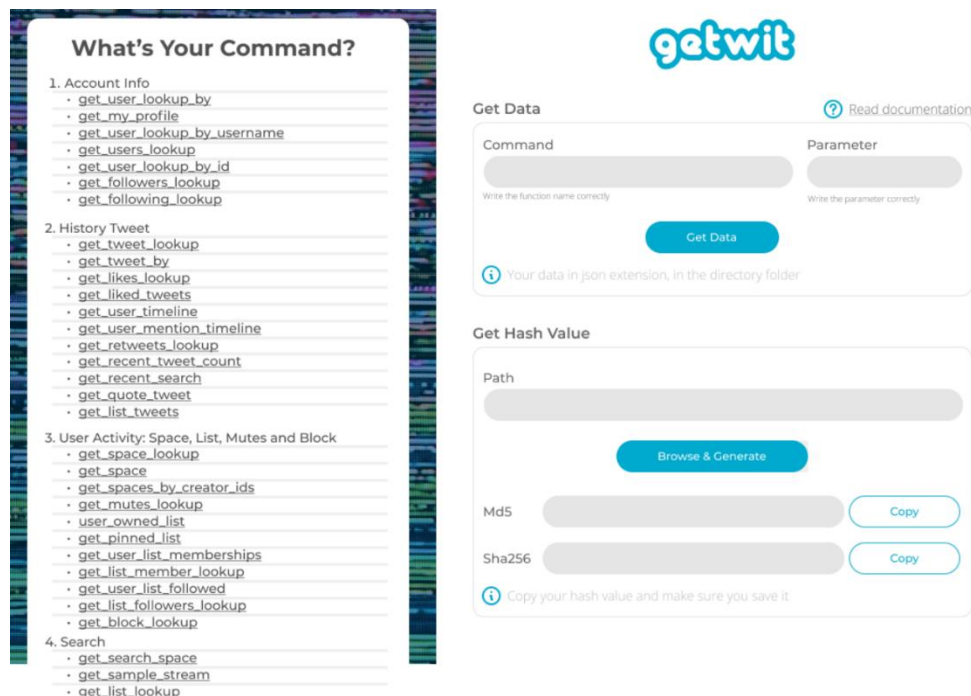
Sebelum mengimplementasikan getwit, pastikan telah memiliki akun developer pada website Twitter Developer. Setelah itu, generate key dan token yang ada pada



```
1 #API_KEY dari Twitter Developer
2
3 #OAUTH1.0a
4 ACCESS_TOKEN = "YOUR ACCES_TOKEN"
5 ACCESS_TOKEN_SECRET = "YOUR TOKEN_SECRET"
6 CONSUMER_KEY = 'YOUR API_KEY'
7 CONSUMER_SECRET = 'YOUR API_SECRET'
8 CALLBACK_URL = 'YOUR CALLBACK_URL'
9 # OAUTH_SIGNATURE_METHOD = "HMAC-SHA1"
10 # OAUTH_VERSION = 1.0
11
12 #OAUTH2.0
13 BEARER_TOKEN = "YOUR BEARER_TOKEN"
14 CLIENT_ID = 'YOUR CLIENT_ID'
15 CLIENT_SECRET = 'YOUR CLIENT_SECRET'
```

Gambar 3 Token dan key disalin ke file api\_secret.py

projects & apps. Key dan token tersebut disalin dan simpan pada file dengan nama api\_secret.api, seperti pada Gambar 3. Perhatikan bahwa status dari aplikasi dan User Autentication Setting mempengaruhi jenis key dan token yang dapat di generate. Untuk detailnya silahkan akses <https://developer.twitter.com/en/docs/apps/overview>.



Gambar 4 Tampilan utama dari Getwit

### 1.3 Running window.py

Temukan window.py pada repository yang telah diclone, running window.py untuk memulai menggunakan aplikasi. Berikut tampilan dashboard dari running window.py pada Gambar 4 .

### 1.4 Pilih Endpoint

Function.py berisi fungsi yang terhubung dengan endpoint yang ada pada Twitter API v2 dengan jenis autentikasi beragam. Ada 32 endpoint yang dapat digunakan untuk mendapatkan data atau informasi seperti bukti digital potensial. Berikut endpoint yang dimaksud,

#### 1.4.1 Tweet Lookup

##### a. get\_tweet\_lookup

Mengembalikan beragam informasi tentang spesifik tweet berdasarkan id atau daftar id.

##### b. get\_tweet\_by

Mengembalikan beragam informasi tentang spesifik tweet berdasarkan sebuah id.

#### 1.4.2 Likes

##### a. get\_liked\_tweets

Mendapatkan informasi mengenai tweet yang disukai pengguna.

##### b. get\_likes\_lookup

Mendapatkan informasi mengenai pengguna yang menyukai sebuah tweet tertentu.

#### 1.4.3 Timeline

##### a. get\_user\_timeline

Mengembalikan Tweet yang dibuat oleh satu pengguna, ditentukan oleh ID pengguna yang diminta. Secara default, sepuluh Tweet terbaru dikembalikan per permintaan. Menggunakan pagination, 3.200 Tweet terbaru dapat diambil.

##### b. get\_user\_mention\_timeline

Mengembalikan Tweet yang menyebutkan satu pengguna yang ditentukan oleh ID pengguna yang diminta. Secara default, sepuluh Tweet terbaru dikembalikan per permintaan. Menggunakan pagination, hingga 800 Tweet terbaru dapat diambil.

#### 1.4.4 Users

a. `get_user_lookup_by`

Mengembalikan berbagai informasi tentang satu atau lebih pengguna yang ditentukan oleh username mereka.

b. `get_my_profile`

Mengembalikan informasi tentang pengguna yang telah ter authorization.

c. `get_user_lookup_by_username`

Mengembalikan berbagai informasi tentang satu atau lebih pengguna yang ditentukan oleh nama pengguna mereka.

d. `get_users_lookup`

Mengembalikan berbagai informasi tentang satu atau lebih pengguna yang ditentukan oleh ID yang diminta.

e. `get_user_lookup_by_id`

Mengembalikan berbagai informasi tentang satu pengguna yang ditentukan oleh ID yang diminta.

#### 1.4.5 Spaces

a. `get_space_lookup`

Mengembalikan berbagai informasi tentang satu pengguna yang ditentukan oleh ID yang diminta.

b. `get_search_space`

Kembalikan Spaces langsung atau terjadwal yang cocok dengan istilah pencarian yang Anda tentukan. Titik akhir ini melakukan pencarian kata kunci, yang berarti akan mengembalikan Spasi yang sama persis dengan huruf besar-kecil yang cocok dengan istilah pencarian yang ditentukan. Istilah pencarian akan cocok dengan judul asli Space.

c. `get_space`

Mengembalikan detail tentang beberapa Spasi. Hingga 100 ID Spasi yang dipisahkan koma dapat dicari menggunakan titik akhir ini.

d. `get_spaces_by_creator_ids`

Mengembalikan Spaces langsung atau terjadwal yang dibuat oleh ID pengguna yang ditentukan. Hingga 100 ID yang dipisahkan koma dapat dicari menggunakan titik akhir ini.

#### 1.4.6 Volume Streams

a. `get_sample_stream`

Mendapatkan sekitar 1% dari semua Tweet secara real-time. Jika Anda memiliki akses '*Academic Research*', Anda dapat menghubungkan hingga dua koneksi redundan untuk memaksimalkan waktu streaming Anda.

1.4.7 Retweets

a. `get_retweets_lookup`

Memungkinkan Anda mendapatkan informasi tentang siapa yang telah me-Retweet Tweet.

1.4.8 Search Tweets

a. `get_recent_tweet`

End point ini mengembalikan Tweet dari tujuh hari terakhir yang cocok dengan permintaan pencarian.

1.4.9 Tweet Count

a. `get_recent_tweet_count`

End point ini mengembalikan jumlah Tweet dari tujuh hari terakhir yang cocok dengan `str_query` yang dimasukkan.

1.4.10 Quote Tweets

a. `get_quote_tweet`

Mengembalikan Tweet Kutipan untuk Tweet yang ditentukan oleh ID Tweet yang diminta

1.4.11 Mutes

a. `get_mutes_lookup`

Mengembalikan daftar pengguna yang dibisukan oleh ID pengguna yang ditentukan.

1.4.12 Lists Lookup

a. `get_user_owned_list`

Mengembalikan semua Daftar yang dimiliki oleh pengguna yang ditentukan.

b. `get_list_lookup`

Mengembalikan detail dari Daftar yang ditentukan.

1.4.13 List Tweets Lookup

a. `get_list_tweets`

Mengembalikan daftar Tweet dari Daftar yang ditentukan.

#### 1.4.14 Pinned Lists

##### a. `get_pinned_list`

Mengembalikan Daftar yang disematkan oleh pengguna tertentu.

#### 1.4.15 List Member

##### a. `get_user_list_membership`

Mengembalikan semua List yang menjadi anggota pengguna tertentu.

##### b. `get_user_list_member`

Mengembalikan daftar pengguna yang menjadi anggota List yang ditentukan.

#### 1.4.16 List Follows

##### a. `get_user_list_followed`

Mengembalikan daftar pengguna yang diikuti dari List yang ditentukan.

##### b. `get_list_followers`

Mengembalikan daftar pengguna yang merupakan pengikut Daftar yang ditentukan.

#### 1.4.17 Follows

##### a. `get_following_lookup`

Mengembalikan daftar pengguna yang diikuti oleh ID pengguna tertentu.

##### b. `get_followers_lookup`

Mengembalikan daftar pengguna yang merupakan pengikut ID pengguna yang ditentukan.

#### 1.4.18 Blocks

##### a. `get_block_lookup`

Mengembalikan daftar pengguna yang diblokir oleh ID pengguna yang ditentukan.





Gambar 3 Tampilan menu endpoint

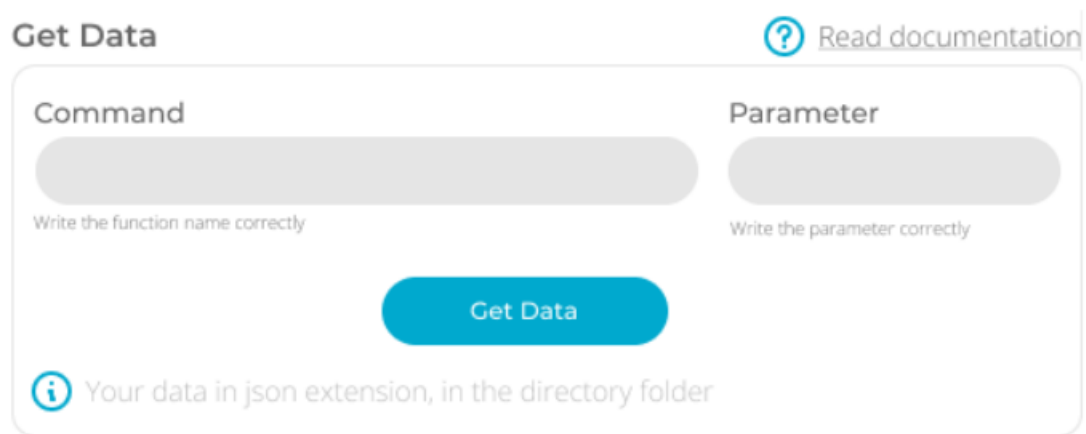
Pada user interface, telah diatur setiap endpoint berdasarkan data yang dapat diakusisi, mulai dari account info hingga user activity. Berikut tampilan user interface yang dimaksud pada Gambar 5.

### 1.5 Masukkan Parameter

Setelah memilih endpoint atau memasukkan nama fungsinya pada entry command, masukkan parameter sesuai parameter pada fungsi tersebut, tampilan fitur GET DATA dapat dilihat pada Gambar 6. Pada parameter dengan tipe data 'integer' dapat ditulis tanpa petik, contohnya 123456. Tipe data 'string' ditulis dengan tanda petik diantaranya contohnya 'xyx'. Selain itu, ada beberapa kasus yang harus diperhatikan penulisannya. Jika fungsinya tidak memiliki parameter, maka tuliskan 'None' (dengan petik) dan jika fungsi memiliki parameter yang pengguna tidak ingin gunakan pada saat pengambilan data, maka tulis nilai parameter dengan 'None' (tanpa petik).

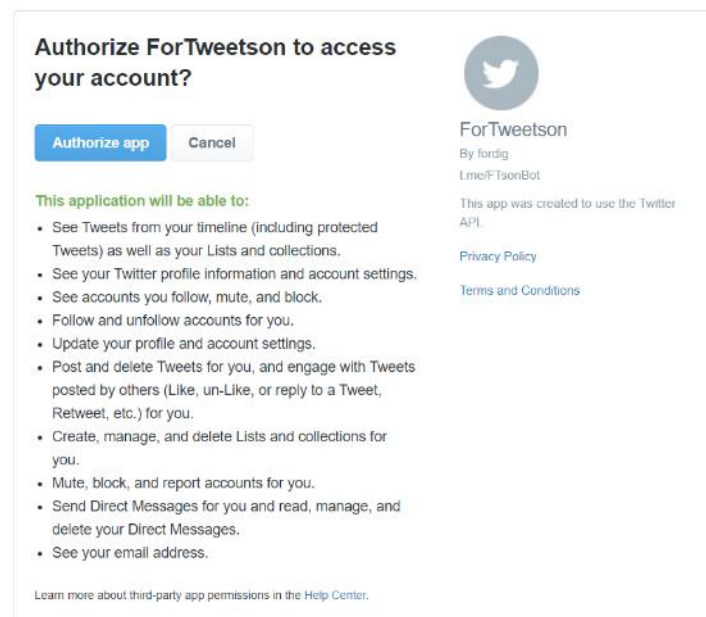
Tekan tombol 'get data' untuk mendapatkan data yang diinginkan. Hasil download disimpan dalam file dengan ekstensi json didalam folder 'databank'. Setiap pemanggilan dengan fungsi yang sama akan menimpa data yang telah diakuisisi

sebelumnya, maka untuk menghindari hal tersebut, pindahkan file tersebut ke folder lain atau mengganti nama file tersebut.



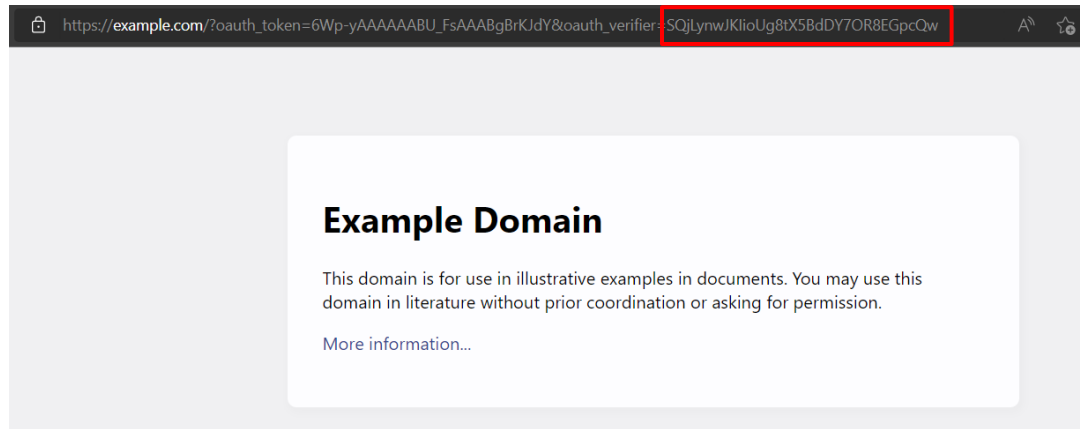
Gambar 4 tampilan frame fitur get data dari endpoint

Ada dua jenis autentikasi yang dilakukan, pertama autentikasi OAuth1.0a yang memerlukan 'consumer\_key' dan 'consumer\_secret'. Disaat fungsi (endpoint) yang memerlukan OAuth1.0a dipanggil, maka akan otomatis menjalankan proses autentikasi tersebut. Pada OAuth1.0a setelah program dieksekusi, akan otomatis diarahkan ke sebuah link autentikasi untuk meminta izin pada aplikasi yang terhubung dengan akun developer, seperti pada Gambar 7.

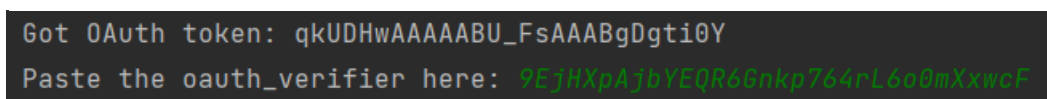


Gambar 5 tampilan webpage authorize app

Setelah itu, akan dialihkan ke halaman sesuai 'CALLBACK\_URL' yang telah diisi pada User Authentication Setting, seperti pada Gambar 8. Pada link tersebut, salin token yang memiliki nama variable oauth\_verifier, bagian yang diberi tanda kotak merah pada gambar 8. Tempel hasil salin tersebut ke dalam terminal dan tekan 'enter'. Untuk tampilannya dapat dilihat pada Gambar 9.



Gambar 6 tampilan setelah melakukan authorized app



Gambar 7 salin bagian oauth\_verifier dari page sebelumnya ke terminal

Jika program telah dieksekusi dan response codenya 200, maka program berhasil dieksekusi dan data telah didownload. Selanjutnya pada fungsi (endpoint) dengan OAuth2.0 App-Only yang memerlukan bearer\_token. Dalam proses akuisisi data hanya perlu menekan tombol 'get data' dan data akan didownload kedalam folder 'databank'. Setiap objek yang berhasil diakuisisi, semuanya diatur didalam file model.py.

File dengan nama model.py berisi query parameter setiap endpointnya. Maka jika ingin mendapatkan lebih banyak objek pada suatu endpoint maka dapat mempelajari setiap objek yang dibutuhkan pada <https://developer.twitter.com/en/docs>. Jenis autentikasi dengan key dan token yang berbeda dapat mempengaruhi objek yang dapat diakuisisi. Model.py juga berpengaruh pada parameter yang akan digunakan dalam memanggil endpoint pada file main.py. Isi dari model.py dapat dimodifikasi dan disesuaikan dengan kebutuhan dan variasi objek yang diinginkan.

### 1.5 Generate hash value

Fitur ini dapat digunakan pada berbagai jenis file untuk mendapatkan nilai hash. Nilai hash yang dimaksud adalah MD5 dan SHA256. Pengguna dapat menyalin hash value tersebut dengan menekan tombol 'copy' di masing-masing entry. Tampilan fitur tersebut dapat dilihat pada Gambar 10.

## 1.6 Read documentation

Fitur ini akan mengarahkan pengguna ke dokumentasi Getwit pada repository Github. Pengguna cukup menekan tombol 'Read documentation' yang berada diatas frame 'Get Data'. Repository tersebut berisi source code dan user manual dari Getwit.


**Get Hash Value**

Path

**Browse & Generate**

Md5  **Copy**

Sha256  **Copy**

 Copy your hash value and make sure you save it

Gambar 8 tampilan frame fitur get hash value

## 3. Source Code

### 3.1 api\_secret.py

```
#API_KEY dari Twitter Developer

#OAUTH1.0a
ACCESS_TOKEN = "YOUR_ACCESS_TOKEN"
ACCESS_TOKEN_SECRET = "YOUR_ACCESS_TOKEN_SECRET"
CONSUMER_KEY = 'YOUR_CONSUMER_KEY'
CONSUMER_SECRET = 'YOUR_CONSUMER_SECRET'

#OAUTH2.0
BEARER_TOKEN = "YOUR_BEARER_TOKEN"
```

### 3.2 model.py

```
#QUERY PARAMETER
#TWEETS
def get_tweet_par(tweet_ids):
    return {
        'ids': tweet_ids,
        'tweet.fields':
        'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,possibly_sensitive,non_public_metrics,refer
enced_tweets,reply_settings,source,text,withheld',
        'user.fields':
        'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',
        'expansions':
        'author_id,referenced_tweets.id,referenced_tweets.id.author_id,entities.menti
```

```

ons.username, attachments.poll_ids, attachments.media_keys, in_reply_to_user_id,
geo.place_id',
    'poll.fields':
'duration_minutes, end_datetime, id, options, voting_status',
    'place.fields':
'contained_within, country, country_code, full_name, geo, id, name, place_type',
    'media.fields':
'alt_text, duration_ms, height, media_key, non_public_metrics, organic_metrics, pre
view_image_url, promoted_metrics, public_metrics, type, url, width'
    }

def get_tweet_by_par():
    return {
        'tweet.fields':
'attachments, author_id, context_annotations, conversation_id, created_at, entitie
s, geo, id, in_reply_to_user_id, lang, non_public_metrics, public_metrics, organic_m
etrics, possibly_sensitive, referenced_tweets, reply_settings, source, text, withhe
ld',
        'user.fields':
'created_at, description, entities, id, location, name, pinned_tweet_id, profile_ima
ge_url, protected, public_metrics, url, username, verified, withheld',
        'expansions':
'attachments.poll_ids, attachments.media_keys, author_id, entities.mentions.user
name, geo.place_id, in_reply_to_user_id, referenced_tweets.id, referenced_tweets.
id.author_id',
        'poll.fields':
'duration_minutes, end_datetime, id, options, voting_status',
        'place.fields':
'contained_within, country, country_code, full_name, geo, id, name, place_type',
        'media.fields':
'duration_ms, height, media_key, preview_image_url, type, url, width, public_metrics
, non_public_metrics, organic_metrics, promoted_metrics, alt_text'
    }

#LIKES
def get_likes_lookup_par(max_results, pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        'max_results': max_results,
        # 'media.fields': 'duration_ms, height, media_key, preview_image_url,
type, url, width, public_metrics, non_public_metrics, organic_metrics,
promoted_metrics, alt_text',
        'pagination_token': pagination_token,
        # 'place.fields': 'contained_within, country, country_code,
full_name, geo, id, name, place_type',
        # 'poll.fields': 'duration_minutes, end_datetime, id, options,
voting_status',
        'tweet.fields':
'attachments, author_id, context_annotations, conversation_id, created_at, entitie
s, geo, id, in_reply_to_user_id, lang, non_public_metrics, organic_metrics, possibly
_sensitive, promoted_metrics, public_metrics, referenced_tweets, reply_settings, s
ource, text, withheld',
        'user.fields':
'created_at, description, entities, id, location, name, pinned_tweet_id, profile_ima
ge_url, protected, public_metrics, url, username, verified, withheld',
    }

```

```

def get_liked_tweets_par(max_results,pagination_token):
    return {
        'expansions':
'author_id,referenced_tweets.id,referenced_tweets.id.author_id,entities.mentions.username,attachments.poll_ids,attachments.media_keys,in_reply_to_user_id,geo.place_id',
        'max_results': max_results,
        'media.fields':
'alt_text,duration_ms,height,media_key,preview_image_url,public_metrics,type,url,width',
        'pagination_token': pagination_token,
        'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
        'poll.fields':
'duration_minutes,end_datetime,id,options,voting_status',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

#TIMELINE
def get_user_timeline_par():
    return {
        'exclude': 'replies,retweets',
        'expansions':
'author_id,referenced_tweets.id,referenced_tweets.id.author_id,entities.mentions.username,attachments.poll_ids,attachments.media_keys,in_reply_to_user_id,geo.place_id',
        'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
        'poll.fields':
'duration_minutes,end_datetime,id,options,voting_status',
        'media.fields':
'alt_text,duration_ms,height,media_key,non_public_metrics,organic_metrics,preview_image_url,public_metrics,type,url,width',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld'
    }

def get_user_mention_timeline_par():
    return {
        'expansions':
'author_id,referenced_tweets.id,referenced_tweets.id.author_id,entities.mentions.username,attachments.poll_ids,attachments.media_keys,in_reply_to_user_id,geo.place_id',
        'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
        'poll.fields':

```

```

'duration_minutes,end_datetime,id,options,voting_status',
    'media.fields':
'alt_text,duration_ms,height,media_key,organic_metrics,preview_image_url,public_metrics,type,url,width',
    'tweet.fields' :
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
    'user.fields' :
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld'
}

#USERS
def get_user_lookup_by_par(usernames):
    return {
        'usernames': usernames,
        'expansions': 'pinned_tweet_id',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,source,text,withheld'
    }

def get_user_lookup_by_username_par():
    return {
        'expansions': 'pinned_tweet_id',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,source,text,withheld'
    }

def get_users_lookup_par(user_ids):
    return {
        'ids': user_ids,
        'expansions': 'pinned_tweet_id',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,source,text,withheld'
    }

def get_user_lookup_by_id_par():
    return {

```

```

        'expansions': 'pinned_tweet_id',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly
_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,s
ource,text,withheld'
    }

def get_my_par():
    return {
        'expansions': 'pinned_tweet_id',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly
_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,s
ource,text,withheld'
    }

#SPACES
def get_spaces_by_id_par(space_id):
    return {
        'ids': space_id,
        'expansions':
'host_ids,creator_id,invited_user_ids,speaker_ids,topic_ids',
        'topic.fields': 'description,id,name',
        'space.fields' :
'created_at,creator_id,ended_at,host_ids,id,invited_user_ids,is_ticketed,lang
,participant_count,scheduled_start,speaker_ids,started_at,state,subscriber_co
unt,title,topic_ids,updated_at',
        'user.fields' :
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld'
    }

def get_search_space_par(str_query):
    return {
        'query': str_query,
        'expansions':
'host_ids,creator_id,invited_user_ids,speaker_ids,topic_ids',
        'state': 'all',
        #live,scheduled,all'
        'topic.fields': 'description,id,name',
        'space.fields' :
'created_at,creator_id,ended_at,host_ids,id,invited_user_ids,is_ticketed,lang
,participant_count,scheduled_start,speaker_ids,started_at,state,subscriber_co
unt,title,topic_ids,updated_at',
        'user.fields' :
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',
    }

```



```

def get_spaces_par(space_ids):
    return {
        'ids': space_ids,
        'expansions':
'host_ids,creator_id,invited_user_ids,speaker_ids,topic_ids',
        'topic.fields': 'id,name,description',
        'space.fields':
'host_ids,created_at,creator_id,id,lang,invited_user_ids,participant_count,speaker_ids,started_at,ended_at,subscriber_count,topic_ids,state,title,updated_at,scheduled_start,is_ticketed',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_spaces_by_creator_ids_par(user_ids):
    return {
        'user_ids': user_ids,
        'expansions': 'invited_user_ids,speaker_ids,creator_id,host_ids',
        'topic.fields': 'id,name,description',
        'space.fields':
'created_at,creator_id,ended_at,host_ids,id,invited_user_ids,is_ticketed,lang,participant_count,scheduled_start,speaker_ids,started_at,state,subscriber_count,title,topic_ids,updated_at',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_spaces_buyers():
    return {
        'expansions': 'pinned_tweet_id',
        'media.fields':
'duration_ms,height,media_key,preview_image_url,type,url,width,public_metrics,non_public_metrics,organic_metrics,promoted_metrics,alt_text',
        'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
        'poll.fields':
'duration_minutes,end_datetime,id,options,voting_status',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,public_metrics,organic_metrics,promoted_metrics,possibly_sensitive,referenced_tweets,reply_settings,source,text,withheld',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_spaces_tweet():
    return {
        'expansions':
'attachments.poll_ids,attachments.media_keys,author_id,entities.mentions.username,geo.place_id,in_reply_to_user_id,referenced_tweets.id,referenced_tweets.id.author_id',
        'media.fields':
'duration_ms,height,media_key,preview_image_url,type,url,width,public_metrics

```

```

,non_public_metrics,organic_metrics,promoted_metrics,alt_text',
    'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
    'poll.fields':
'duration_minutes,end_datetime,id,options,voting_status',
    'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,non_public_metrics,public_metrics,organic_m
etrics,promoted_metrics,possibly_sensitive,referenced_tweets,reply_settings,s
ource,text,withheld',
    'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',
    }

#RETWEETS
def get_retweets_lookup_par(pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        # 'media.fields': 'duration_ms, height, media_key, preview_image_url,
type, url, width, public_metrics, non_public_metrics, organic_metrics,
promoted_metrics, alt_text',
        'pagination_token': pagination_token,
        # 'place.fields':
'id,max_results,pagination_token,expansions,tweet.fields,user.fields',
        # 'poll.fields' :
'id,max_results,pagination_token,expansions,tweet.fields,user.fields',
        'tweet.fields' :
'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly
_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,s
ource,text,withheld',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld'
    }

#SEARCH TWEETS
def get_recent_tweet_count_par(str_query, end_time, granularity, since_id,
start_time, until_id):
    return {
        'query': str_query,
        'end_time': end_time,
        'granularity': granularity,
        #minute, hour or day
        'since_id': since_id,
        'start_time': start_time,
        'until_id': until_id
    }

def get_recent_search_par(str_query, end_time, max_results, next_token,
since_id, start_time, until_id):
    return {
        'query': str_query,
        'end_time': end_time,
        #YYYY-MMMM-DDDDTHH:mm:ssZ
        'expansions':

```

```

'author_id,referenced_tweets.id,referenced_tweets.id.author_id,entities.mentions.username,attachments.poll_ids,attachments.media_keys,in_reply_to_user_id,geo.place_id',
    'max_results': max_results,
    'media.fields':
'alt_text,duration_ms,height,media_key,preview_image_url,public_metrics,type,url,width',
    'next_token': next_token,
    'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
    'poll.fields':
'duration_minutes,end_datetime,id,options,voting_status',
    'since_id': since_id,
    'start_time': start_time,
    'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
    'until_id': until_id,
    'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

#QUOTE TWEETS
def get_quote_tweet_par(max_results, pagination_token):
    return {
        'expansions':
'author_id,referenced_tweets.id,referenced_tweets.id.author_id,entities.mentions.username,attachments.poll_ids,attachments.media_keys,in_reply_to_user_id,geo.place_id',
        'max_results': max_results,
        # 'media.fields': 'duration_ms, height, media_key, preview_image_url, type, url, width, public_metrics, non_public_metrics, organic_metrics, promoted_metrics, alt_text',
        'pagination_token': pagination_token,
        'place.fields':
'contained_within,country,country_code,full_name,geo,id,name,place_type',
        'poll.fields':
'duration_minutes,end_datetime,id,options,voting_status',
        'tweet.fields':
'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld'
    }

#MUTES
def get_mutes_lookup_par(max_results, pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields':

```

```

'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
    'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

#LIST
def user_owned_list(max_results,pagination_token):
    return {
        'expansions': 'owner_id',
        'list.fields':
'created_at,description,follower_count,id,member_count,name,owner_id,private'
    ,
        'max_results': max_results,
        'pagination_token': pagination_token,
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_list_lookup_par():
    return {
        'expansions': 'owner_id',
        'list.fields':
'created_at,description,follower_count,id,member_count,name,owner_id,private'
    ,
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_pinned_list_par():
    return {
        'expansions': 'owner_id',
        'list.fields':
'created_at,description,follower_count,id,member_count,name,owner_id,private'
    ,
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_list_membership_par(max_results, pagination_token):
    return {
        'expansions': 'owner_id',
        'list.fields':
'created_at,description,follower_count,id,member_count,name,owner_id,private'
    ,
        'max_results': max_results,
        'pagination_token': pagination_token,
        'user.fields':
'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

```

```

def get_list_member_lookup_par(max_results, pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields':
            'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,non_public_metrics,organic_metrics,possibly_sensitive,promoted_metrics,public_metrics,referenced_tweets,reply_settings,source,text,withheld',
        'user.fields':
            'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_user_list_followed_par(max_results,pagination_token):
    return {
        'expansions': 'owner_id',
        'list.fields':
            'created_at,description,follower_count,id,member_count,name,owner_id,private',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'user.fields':
            'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_list_followers_lookup(max_results, pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields':
            'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,reference_d_tweets,reply_settings,source,text,withheld',
        'user.fields':
            'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

def get_list_tweets_par(max_results, pagination_token):
    return {
        'expansions': 'author_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields':
            'attachments,author_id,context_annotations,conversation_id,created_at,entities,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,reference_d_tweets,reply_settings,source,text,withheld',
        'user.fields':
            'created_at,description,entities,id,location,name,pinned_tweet_id,profile_image_url,protected,public_metrics,url,username,verified,withheld',
    }

```

```

#FOLLOWS
def get_following_lookup_par(max_results, pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields':
            'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,reference
d_tweets,reply_settings,source,text,withheld',
        'user.fields':
            'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',

    }

def get_followers_lookup_par(max_results, pagination_token):
    return {
        'expansions': 'pinned_tweet_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields':
            'attachments,author_id,context_annotations,conversation_id,created_at,entitie
s,geo,id,in_reply_to_user_id,lang,possibly_sensitive,public_metrics,reference
d_tweets,reply_settings,source,text,withheld',
        'user.fields':
            'created_at,description,entities,id,location,name,pinned_tweet_id,profile_ima
ge_url,protected,public_metrics,url,username,verified,withheld',

    }

#BLOCKS
def block_lookups_par(max_results,pagination_token):
    return {
        'expansion': 'pinned_tweet_id',
        'max_results': max_results,
        'pagination_token': pagination_token,
        'tweet.fields': 'attachments, author_id, context_annotations,
conversation_id, created_at, entities, geo, id, in_reply_to_user_id, lang,
non_public_metrics, public_metrics, organic_metrics, promoted_metrics,
possibly_sensitive, referenced_tweets, reply_settings, source, text,
withheld',
        'user.fields': 'created_at, description, entities, id, location,
name, pinned_tweet_id, profile_image_url, protected, public_metrics, url,
username, verified, withheld',

    }

```

### 3.3 function.py

```

import webbrowser

import requests_oauthlib
from requests_oauthlib import OAuth1Session
import json
from getwit import model
from getwit import api_secret
import requests

```

```

# from urllib import parse
# import base64
# import hashlib
# import re
# import os
# from requests_oauthlib import OAuth2Session

consumer_key = api_secret.CONSUMER_KEY
consumer_secret = api_secret.CONSUMER_SECRET
bearer_token = api_secret.BEARER_TOKEN
client_id = api_secret.CLIENT_ID
redirect_uri = api_secret.CALLBACK_URL
headers = {"Authorization": "Bearer {}".format(bearer_token)}

#TWEETS
def get_tweet_lookup(tweet_ids):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Click this link and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,

```

```

        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_tweet_par(tweet_ids)
    response = oauth.get(
        "https://api.twitter.com/2/tweets", params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/tweet_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_tweet_by(tweet_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]

```



```

access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_tweet_by_par()
response = oauth.get(
    "https://api.twitter.com/2/tweets/{}".format(tweet_id), params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/tweet_by.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

#LIKES
def get_likes_lookup(tweet_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,

```

```

        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_likes_lookup_par(max_results, pagination_token)
    response = oauth.get(
        "https://api.twitter.com/2/tweets/{}/liking_users".format(tweet_id),
        params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
            response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/likes_tweets.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_liked_tweets(user_id,max_results,pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"

```

```

oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_liked_tweets_par(max_results, pagination_token)
response = oauth.get(
    "https://api.twitter.com/2/users/{}/liked_tweets".format(user_id),
params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/liked_tweets.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

#TIMELINE
def get_user_timeline(user_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)

```

```

# print("Please go here and authorize: %s" % authorization_url)
webbrowser.open(authorization_url)
verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_user_timeline_par()
response = oauth.get(
    "https://api.twitter.com/2/users/{}/tweets".format(user_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/user_timeline.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

def get_user_mention_timeline(user_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
            consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")

```

```

print("Got OAuth token: %s" % resource_owner_key)

# Get authorization
base_authorization_url = "https://api.twitter.com/oauth/authorize"
authorization_url = oauth.authorization_url(base_authorization_url)
# print("Please go here and authorize: %s" % authorization_url)
webbrowser.open(authorization_url)
verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_user_mention_timeline_par()
response = oauth.get(
    "https://api.twitter.com/2/users/{}/mentions".format(user_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/user_mention_timeline.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

#USERS
def get_user_lookup_by(usernames):
    url = 'https://api.twitter.com/2/users/by'.format(usernames)
    params = model.get_user_lookup_by_par(usernames)
    respon = requests.request("GET", url, headers=headers, params=params)
    print(respon.status_code)
    for response_line in respon.iter_lines():
        if response_line:
            json_response = json.loads(response_line)

```

```

        with open("databank/user_lookup_by.json", "w") as file:
            json.dump(json_response, file, indent=4, sort_keys=True)
    if respon.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                respon.status_code, respon.text
            )
        )

def get_my_profile():
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_my_par()
    response = oauth.get(
        "https://api.twitter.com/2/users/me", params=params
    )

```

```

    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/my_profile.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_user_lookup_by_username(username):
    url = 'https://api.twitter.com/2/users/by/username/{}'.format(username)
    params = model.get_user_lookup_by_username_par()
    respon = requests.request("GET", url, headers=headers, params=params)
    print(respon.status_code)
    for response_line in respon.iter_lines():
        if response_line:
            json_response = json.loads(response_line)
            with open("databank/user_lookup_by.json", "w") as file:
                json.dump(json_response, file, indent=4, sort_keys=True)
    if respon.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                respon.status_code, respon.text
            )
        )

def get_users_lookup(user_ids):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,

```

```

        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_users_lookup_par(user_ids)
    response = oauth.get(
        "https://api.twitter.com/2/users", params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/users_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_user_lookup_by_id(user_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

```



```

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_user_lookup_by_id_par()
response = oauth.get(
    "https://api.twitter.com/2/users/{}".format(user_id), params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/user_lookup_by_id.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

#SPACES
def get_space_lookup(space_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"

```

```

authorization_url = oauth.authorization_url(base_authorization_url)
# print("Please go here and authorize: %s" % authorization_url)
webbrowser.open(authorization_url)
verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_spaces_by_id_par(space_id)
response = oauth.get(
    "https://api.twitter.com/2/spaces/{}".format(space_id), params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/space_lookup.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

def get_search_space(str_query, state):
    url = 'https://api.twitter.com/2/spaces/search'.format(str_query)
    params = model.get_search_space_par(str_query, state)
    respon = requests.request("GET", url, headers=headers, params=params)
    print(respon.status_code)
    for response_line in respon.iter_lines():
        if response_line:
            json_response = json.loads(response_line)
            with open("databank/search_space.json", "w") as file:
                json.dump(json_response, file, indent=4, sort_keys=True)
            # print(json.dumps(json_response, indent=4, sort_keys=True))
    if respon.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(

```

```

        respon.status_code, respon.text
    )
)

def get_spaces(space_ids):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_spaces_par(space_ids)
    response = oauth.get(
        'https://api.twitter.com/2/spaces', params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,

```

```

response.text)
    )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/spaces_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_spaces_by_creator_ids(user_ids):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_spaces_by_creator_ids_par(user_ids)
    response = oauth.get(
        'https://api.twitter.com/2/spaces/by/creator_ids'.format(user_ids),
        params=params
    )

```

```

    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/spaces_by_creator_id.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

#OAuth2.0 Autorization Code with PKCE
# def get_spaces_buyers(space_id):
#     url = 'https://api.twitter.com/2/spaces/{}/buyers'.format(space_id)
#     params = model.get_spaces_buyers()
#     respon = requests.request("GET", url, headers=headers, params=params)
#     print(respon.status_code)
#     for response_line in respon.iter_lines():
#         if response_line:
#             json_response = json.loads(response_line)
#             print(json.dumps(json_response, indent=4, sort_keys=True))
#     if respon.status_code != 200:
#         raise Exception(
#             "Request returned an error: {} {}".format(
#                 respon.status_code, respon.text
#             )
#         )
#
#
# def get_spaces_tweet(space_id):
#     url = 'https://api.twitter.com/2/spaces/:id/tweets'.format(space_id)
#     params = model.get_search_space_par(space_id)
#     respon = requests.request("GET", url, headers=headers, params=params)
#     print(respon.status_code)
#     for response_line in respon.iter_lines():
#         if response_line:
#             json_response = json.loads(response_line)
#             print(json.dumps(json_response, indent=4, sort_keys=True))
#     if respon.status_code != 200:
#         raise Exception(
#             "Request returned an error: {} {}".format(
#                 respon.status_code, respon.text
#             )
#         )
#
#
#VOLUME STREAMS

def get_sample_stream():
    param = model.sample_stream_par()
    headers = {"Authorization": "Bearer {}".format(bearer_token),
               "User-Agent": "v2SampleStreamJS"}
    url = 'https://api.twitter.com/2/tweets/sample/stream'
    respon = requests.request("GET", url, params= param, headers=headers,
stream=True)
    print(respon.status_code)
    for response_line in respon.iter_lines():

```

```

        if response_line:
            json_response = json.loads(response_line)
            print(json.dumps(json_response, indent=4, sort_keys=True))
    if respon.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                respon.status_code, respon.text
            )
        )

#RETWEETS
def get_retweets_lookup(tweet_id, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_retweets_lookup_par(pagination_token)

```

```

        response = oauth.get(
            "https://api.twitter.com/2/tweets/{}/retweeted_by".format(tweet_id),
            params=params
        )

        if response.status_code != 200:
            raise Exception(
                "Request returned an error: {} {}".format(response.status_code,
                    response.text)
            )

        print("Response code: {}".format(response.status_code))
        json_response = response.json()
        with open("databank/retweets_lookup.json", "w") as file:
            json.dump(json_response, file, indent=4, sort_keys=True)

#SEARCH TWEETS
def get_recent_tweet_count(str_query, end_time, granularity, since_id,
start_time, until_id):
    url = "https://api.twitter.com/2/tweets/counts/recent".format(str_query)
    params = model.get_recent_tweet_count_par(str_query, end_time,
granularity, since_id, start_time, until_id)
    respon = requests.request("GET", url, headers=headers, params=params)
    print(respon.status_code)
    for response_line in respon.iter_lines():
        if response_line:
            json_response = json.loads(response_line)
            with open("databank/recent_tweet_count.json", "w") as file:
                json.dump(json_response, file, indent=4, sort_keys=True)
    if respon.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                respon.status_code, respon.text
            )
        )

def get_recent_search(str_query, end_time, max_results, next_token, since_id,
start_time, until_id):
    url = "https://api.twitter.com/2/tweets/search/recent".format(str_query)
    params = model.get_recent_search_par(str_query, end_time, max_results,
next_token, since_id, start_time, until_id)
    respon = requests.request("GET", url, headers=headers, params=params)
    print(respon.status_code)
    for response_line in respon.iter_lines():
        if response_line:
            json_response = json.loads(response_line)
            with open("databank/recent_search.json", "w") as file:
                json.dump(json_response, file, indent=4, sort_keys=True)
    if respon.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                respon.status_code, respon.text
            )
        )

#QUOTE TWEETS
def get_quote_tweet(tweet_id, max_results, pagination_token):

```

```

request_token_url = "https://api.twitter.com/oauth/request_token"
oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

try:
    fetch_response = oauth.fetch_request_token(request_token_url)
except ValueError:
    print(
        "There may have been an issue with the consumer_key or
consumer_secret you entered."
    )

resource_owner_key = fetch_response.get("oauth_token")
resource_owner_secret = fetch_response.get("oauth_token_secret")
print("Got OAuth token: %s" % resource_owner_key)

# Get authorization
base_authorization_url = "https://api.twitter.com/oauth/authorize"
authorization_url = oauth.authorization_url(base_authorization_url)
# print("Please go here and authorize: %s" % authorization_url)
webbrowser.open(authorization_url)
verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_quote_tweet_par(max_results, pagination_token)
response = oauth.get(
    "https://api.twitter.com/2/tweets/{}/quote_tweets".format(tweet_id)
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()

```



```

with open("databank/quote_tweets.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

#MUTES
def get_mutes_lookup(user_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_mutes_lookup_par(max_results, pagination_token)
    response = oauth.get(
        "https://api.twitter.com/2/users/{}/muting".format(user_id),
        params=params
    )

    if response.status_code != 200:
        raise Exception(

```

```

        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/mutes_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

#LIST
def user_owned_list(user_id,max_results,pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.user_owned_list(max_results,pagination_token)
    response = oauth.get(

```

```

        "https://api.twitter.com/2/users/{}/owned_lists".format(user_id),
params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/user_owned_list.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_list_lookup(list_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,

```

```

        resource_owner_secret=access_token_secret,
    )

    params = model.get_list_lookup_par()
    response = oauth.get(
        "https://api.twitter.com/2/lists/{}".format(list_id),
params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/list_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_pinned_list(user_id):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

```

```

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_pinned_list_par()
response = oauth.get(
    "https://api.twitter.com/2/users/{}/pinned_lists".format(user_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/pinned_list.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

def get_user_list_memberships(user_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
            consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )

```

```

oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_list_membership_par(max_results, pagination_token)
response = oauth.get(
    "https://api.twitter.com/2/users/{}/list_memberships".format(user_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/user_list_membership.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

def get_list_member_lookup(list_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(

```

```

        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_list_member_lookup_par(max_results, pagination_token)
    response = oauth.get(
        "https://api.twitter.com/2/lists/{}/members".format(list_id),
params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/list_member_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

def get_user_list_followed(user_id,max_results,pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)

```

```

verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_user_list_followed_par(max_results, pagination_token)
response = oauth.get(
    "https://api.twitter.com/2/users/{}/followed_lists".format(user_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/list_followed.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

def get_list_followers_lookup(list_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
            consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

```



```

# Get authorization
base_authorization_url = "https://api.twitter.com/oauth/authorize"
authorization_url = oauth.authorization_url(base_authorization_url)
# print("Please go here and authorize: %s" % authorization_url)
webbrowser.open(authorization_url)
verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_list_followers_lookup(max_results, pagination_token)
response = oauth.get(
    "https://api.twitter.com/2/lists/{}/followers".format(list_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/list_followers.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

def get_list_tweets(list_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
            consumer_secret you entered."
        )

```

```

resource_owner_key = fetch_response.get("oauth_token")
resource_owner_secret = fetch_response.get("oauth_token_secret")
print("Got OAuth token: %s" % resource_owner_key)

# Get authorization
base_authorization_url = "https://api.twitter.com/oauth/authorize"
authorization_url = oauth.authorization_url(base_authorization_url)
# print("Please go here and authorize: %s" % authorization_url)
webbrowser.open(authorization_url)
verifier = input("Paste the oauth_verifier here: ")

# Get the access token
access_token_url = "https://api.twitter.com/oauth/access_token"
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=resource_owner_key,
    resource_owner_secret=resource_owner_secret,
    verifier=verifier,
)
oauth_tokens = oauth.fetch_access_token(access_token_url)

access_token = oauth_tokens["oauth_token"]
access_token_secret = oauth_tokens["oauth_token_secret"]

# Make the request
oauth = OAuth1Session(
    consumer_key,
    client_secret=consumer_secret,
    resource_owner_key=access_token,
    resource_owner_secret=access_token_secret,
)

params = model.get_list_tweets_par(max_results, pagination_token)
response = oauth.get(
    "https://api.twitter.com/2/lists/{}/tweets".format(list_id),
    params=params
)

if response.status_code != 200:
    raise Exception(
        "Request returned an error: {} {}".format(response.status_code,
        response.text)
    )

print("Response code: {}".format(response.status_code))
json_response = response.json()
with open("databank/list_tweets.json", "w") as file:
    json.dump(json_response, file, indent=4, sort_keys=True)

#FOLLOWS
def get_following_lookup(user_id, max_results, pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:

```

```

        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_following_lookup_par(max_results, pagination_token)
    response = oauth.get(
        "https://api.twitter.com/2/users/{}/following_lookup".format(user_id),
        params=params
    )

    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code,
response.text)
        )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/following.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

```

```

def get_followers_lookup(user_id, max_results, pagination_token):
    url = "https://api.twitter.com/2/users/{}/followers".format(user_id)
    params = model.get_followers_lookup_par(max_results, pagination_token)

    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.get_followers_lookup_par(max_results, pagination_token)
    response = oauth.get(
        "https://api.twitter.com/2/users/{}/followers".format(user_id),
        params=params
    )

    if response.status_code != 200:
        raise Exception(

```

```

        "Request returned an error: {} {}".format(response.status_code,
response.text)
    )

    print("Response code: {}".format(response.status_code))
    json_response = response.json()
    with open("databank/followers_lookup.json", "w") as file:
        json.dump(json_response, file, indent=4, sort_keys=True)

#BLOCK
def get_block_lookup(user_id,max_results,pagination_token):
    request_token_url = "https://api.twitter.com/oauth/request_token"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)

    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
        print(
            "There may have been an issue with the consumer_key or
consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    # print("Please go here and authorize: %s" % authorization_url)
    webbrowser.open(authorization_url)
    verifier = input("Paste the oauth_verifier here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    # Make the request
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )

    params = model.block_lookups_par(max_results,pagination_token)

```

```

        response = oauth.get(
            "https://api.twitter.com/2/users/{}/blocking".format(user_id),
            params=params
        )

        if response.status_code != 200:
            raise Exception(
                "Request returned an error: {} {}".format(response.status_code,
                    response.text)
            )

        print("Response code: {}".format(response.status_code))
        json_response = response.json()
        with open("databank/block_lookup.json", "w") as file:
            json.dump(json_response, file, indent=4, sort_keys=True)

```

### 3.4 window.py

```

from tkinter import *
import hashlib
from getpass import getuser
from os import getcwd, name
from tkinter import filedialog
from getwit import function
import webbrowser

window = Tk()

window.geometry("1080x796")
window.title("Getwit - Forensic Tools for Acquisition Data from Twitter")
window.iconbitmap("C:/Users/Ahmad/PycharmProjects/getwit
project/win/icon/logo.ico")
window.configure(bg = "#ffffff")
canvas = Canvas(
    window,
    bg = "#ffffff",
    height = 708,
    width = 1080,
    bd = 0,
    highlightthickness = 0,
    relief = "ridge")
canvas.place(x = 0, y = 0)

path = StringVar()
val = StringVar()
val2 = StringVar()
val3 = StringVar()
val4 = StringVar()

def btn_clicked():
    print("Button Clicked")

def settext(text):
    entry0.delete(0, END)
    entry0.insert(0, text)

```

```

    return

def clearbox():
    entry0.delete(0, "end")
    entry1.delete(0, "end")
    return

def getcommand():
    result = entry0.get()
    strpar = entry1.get()
    if strpar != None:
        param = eval(strpar)

        if result == "get_tweet_lookup(tweet_ids)":
            user = function.get_tweet_lookup(param)
            print(user)

        if result == "get_tweet_by(tweet_id)":
            user = function.get_tweet_by(param)
            print(user)

        if result == "get_user_timeline(user_id)":
            user = function.get_user_timeline(param)
            print(user)

        if result == "get_user_mention_timeline(user_id)":
            user = function.get_user_mention_timeline(param)
            print(user)

        if result == "get_user_lookup_by(usernames)":
            user = function.get_user_lookup_by(param)
            print(user)

        if result == "get_my_profile()":
            user = function.get_my_profile()
            print(user)

        if result == "get_user_lookup_by_username(username)":
            user = function.get_user_lookup_by_username(param)
            print(user)

        if result == "get_users_lookup(user_ids)":
            user = function.get_users_lookup(param)
            print(user)

        if result == "get_user_lookup_by_id(user_id)":
            user = function.get_user_lookup_by_id(param)
            print(user)

        if result == "get_space_lookup(space_id)":
            user = function.get_space_lookup(param)
            print(user)

        if result == "get_search_space(str_query)":
            user = function.get_search_space(param)
            print(user)

```

```

    if result == "get_spaces(space_ids)":
        user = function.get_spaces(param)
        print(user)

    if result == "get_spaces_by_creator_ids(user_ids)":
        user = function.get_spaces_by_creator_ids(param)
        print(user)

    if result == "get_sample_stream()":
        strpar = None
        user = function.get_sample_stream()
        print(user)

    if result == "get_likes_lookup(tweet_id, max_results,
pagination_token)":
        user = function.get_likes_lookup(param[0], param[1], param[2])
        print(user)

    if result ==
"get_liked_tweets(user_id,max_results,pagination_token)":
        user = function.get_liked_tweets(param[0], param[1], param[2])
        print(user)

    if result == "get_retweets_lookup(tweet_id, pagination_token)":
        user = function.get_retweets_lookup(param[0], param[1])
        print(user)

    if result == "get_recent_tweet_count(str_query)":
        user = function.get_recent_tweet_count(param)
        print(user)

    if result == "get_recent_search(str_query, end_time, max_results,
next_token, since_id, start_time, until_id)":
        user =
function.get_recent_search(param[0],param[1],param[2],param[3],param[4],param
[5], param[6])
        print(user)

    if result == "get_quote_tweet(tweet_id, max_results,
pagination_token)":
        user = function.get_quote_tweet(param[0],param[1],param[2])
        print(user)

    if result == "get_mutes_lookup(user_id, max_results,
pagination_token)":
        user = function.get_mutes_lookup(param[0],param[1],param[2])
        print(user)

    if result == "user_owned_list(user_id,max_results,pagination_token)":
        user = function.user_owned_list(param[0],param[1],param[2])
        print(user)

    if result == "get_list_lookup(list_id)":
        user = function.get_list_lookup(param)
        print(user)

    if result == "get_pinned_list(user_id)":

```



```

        user = function.get_pinned_list(param)
        print(user)

        if result == "get_user_list_memberships(user_id,max_results,
pagination_token)":
            user =
function.get_user_list_memberships(param[0],param[1],param[2])
            print(user)

        if result == "get_list_member_lookup(list_id, max_results,
pagination_token)":
            user =
function.get_list_member_lookup(param[0],param[1],param[2])
            print(user)

        if result ==
"get_user_list_followed(user_id,max_results,pagination_token)":
            user =
function.get_user_list_followed(param[0],param[1],param[2])
            print(user)

        if result == "get_list_followers_lookup(list_id, max_results,
pagination_token)":
            user =
function.get_list_followers_lookup(param[0],param[1],param[2])
            print(user)

        if result == "get_list_tweets(list_id, max_results,
pagination_token)":
            user = function.get_list_tweets(param[0],param[1],param[2])
            print(user)

        if result == "get_following_lookup(user_id, max_results,
pagination_token)":
            user = function.get_following_lookup(param[0],param[1],param[2])
            print(user)

        if result == "get_followers_lookup(user_id, max_results,
pagination_token = None)":
            user = function.get_followers_lookup(param[0],param[1],param[2])
            print(user)

        if result ==
"get_block_lookup(user_id,max_results,pagination_token)":
            user = function.get_block_lookup(param[0],param[1],param[2])
            print(user)
        clearbox()

def open_file():
    global input_file
    in_path = "/home/" + getuser() if name == "posix" else getcwd
    input_file = filedialog.askopenfilename(
        initialdir=in_path, title="Select file", filetypes=[("All Files ",
"*.*")]
    )
    path.set(input_file)
    calculate()

```

```

def calculate():
    md5 = hashlib.md5()
    sha256 = hashlib.sha256()
    sha1 = hashlib.sha1()
    sha512 = hashlib.sha512()

    try:
        file = open(input_file, "rb")
        data = file.read()
        sha256.update(data)
        md5.update(data)
        sha1.update(data)
        sha512.update(data)

        val.set(md5.hexdigest().upper())
        val2.set(sha256.hexdigest().upper())
        val3.set(sha256.hexdigest().upper())
        val4.set(sha256.hexdigest().upper())

    except:
        pass

def copy(item):
    window.clipboard_clear()
    window.clipboard_append(item.get())

def opendoc():
    url = "https://github.com/sonspyscode/getwit"
    webbrowser.open(url)

background_img = PhotoImage(file=f"win/background.png")
background = canvas.create_image(
    520.0, 398.0,
    image=background_img)

entry0_img = PhotoImage(file=f"win/img_textBox0.png")
entry0_bg = canvas.create_image(
    681.5, 182.0,
    image=entry0_img)

entry0 = Entry(
    bd = 0,
    bg = "#e5e5e5",
    highlightthickness = 0)

entry0.place(
    x = 533.0, y = 165,
    width = 297.0,
    height = 32)

entry1_img = PhotoImage(file=f"win/img_textBox1.png")
entry1_bg = canvas.create_image(
    946.0, 182.0,
    image=entry1_img)

```

```

entry1 = Entry(
    bd = 0,
    bg = "#e5e5e5",
    highlightthickness = 0)

entry1.place(
    x = 880.0, y = 165,
    width = 132.0,
    height = 32)

img0 = PhotoImage(file =f"win/img0.png")
b0 = Button(
    image = img0,
    borderwidth = 0,
    highlightthickness = 0,
    command = getcommand,
    relief = "flat")

b0.place(
    x = 690, y = 235,
    width = 143,
    height = 34)

entry2_img = PhotoImage(file =f"win/img_textBox2.png")
entry2_bg = canvas.create_image(
    772.5, 431.0,
    image = entry2_img)

entry2 = Entry(
    bd = 0,
    bg = "#e5e5e5",
    highlightthickness = 0,
    textvariable= path)

entry2.place(
    x = 533.0, y = 414,
    width = 479.0,
    height = 32)

entry3_img = PhotoImage(file =f"win/img_textBox3.png")
entry3_bg = canvas.create_image(
    740.5, 545.0,
    image = entry3_img)

entry3 = Entry(
    bd = 0,
    bg = "#e5e5e5",
    highlightthickness = 0,
    textvariable= val)

entry3.place(
    x = 596.0, y = 528,
    width = 289.0,
    height = 32)

entry4_img = PhotoImage(file =f"win/img_textBox4.png")

```

```

entry4_bg = canvas.create_image(
    740.5, 595.0,
    image = entry4_img)

entry4 = Entry(
    bd = 0,
    bg = "#e5e5e5",
    highlightthickness = 0,
    textvariable=val2)

entry4.place(
    x = 596.0, y = 578,
    width = 289.0,
    height = 32)

img1 = PhotoImage(file =f"win/img1.png")
b1 = Button(
    image = img1,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda: copy(val),
    relief = "flat")

b1.place(
    x = 908, y = 528,
    width = 121,
    height = 34)

img2 = PhotoImage(file =f"win/img2.png")
b2 = Button(
    image = img2,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda: copy(val2),
    relief = "flat")

b2.place(
    x = 908, y = 578,
    width = 121,
    height = 34)

img3 = PhotoImage(file =f"win/img3.png")
b3 = Button(
    image = img3,
    borderwidth = 0,
    highlightthickness = 0,
    command = opendoc,
    relief = "flat")

b3.place(
    x = 880, y = 102,
    width = 160,
    height = 25)

img4 = PhotoImage(file =f"win/img4.png")
b4 = Button(
    image = img4,

```

```

        borderwidth = 0,
        highlightthickness = 0,
        command = open_file,
        relief = "flat")

b4.place(
    x = 659, y = 469,
    width = 205,
    height = 34)

#get function
img5 = PhotoImage(file =f"win/img5.png")
b5 = Button(
    image = img5,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda: settext("get_user_lookup_by(usernames)"),
    relief = "flat")

b5.place(
    x = 55, y = 95,
    width = 359,
    height = 18)

img6 = PhotoImage(file =f"win/img6.png")
b6 = Button(
    image = img6,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda: settext("get_search_space(str_query)"),
    relief = "flat")

b6.place(
    x = 55, y = 692,
    width = 359,
    height = 18)

img7 = PhotoImage(file =f"win/img7.png")
b7 = Button(
    image = img7,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda: settext("get_my_profile()"),
    relief = "flat")

b7.place(
    x = 55, y = 113,
    width = 359,
    height = 18)

img8 = PhotoImage(file =f"win/img8.png")
b8 = Button(
    image = img8,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda: settext("get_sample_stream()"),
    relief = "flat")

```

```

b8.place(
    x = 55, y = 710,
    width = 359,
    height = 18)

img9 = PhotoImage(file =f"win/img9.png")
b9 = Button(
    image = img9,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_user_lookup_by_username(username)"),
    relief = "flat")

b9.place(
    x = 55, y = 131,
    width = 359,
    height = 18)

img10 = PhotoImage(file =f"win/img10.png")
b10 = Button(
    image = img10,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_list_lookup(list_id)"),
    relief = "flat")

b10.place(
    x = 55, y = 728,
    width = 359,
    height = 18)

img11 = PhotoImage(file =f"win/img11.png")
b11 = Button(
    image = img11,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_users_lookup(user_ids)"),
    relief = "flat")

b11.place(
    x = 55, y = 149,
    width = 359,
    height = 18)

img12 = PhotoImage(file =f"win/img12.png")
b12 = Button(
    image = img12,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_user_lookup_by_id(user_id)"),
    relief = "flat")

b12.place(
    x = 55, y = 167,
    width = 359,
    height = 18)

```

```

img13 = PhotoImage(file =f"win/img13.png")
b13 = Button(
    image = img13,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_followers_lookup(user_id, max_results,
pagination_token)"),
    relief = "flat")

b13.place(
    x = 55, y = 184,
    width = 359,
    height = 18)

img14 = PhotoImage(file =f"win/img14.png")
b14 = Button(
    image = img14,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_tweet_lookup(tweet_ids)"),
    relief = "flat")

b14.place(
    x = 55, y = 250,
    width = 359,
    height = 18)

img15 = PhotoImage(file =f"win/img15.png")
b15 = Button(
    image = img15,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_tweet_by(tweet_id)"),
    relief = "flat")

b15.place(
    x = 55, y = 268,
    width = 359,
    height = 18)

img16 = PhotoImage(file =f"win/img16.png")
b16 = Button(
    image = img16,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_likes_lookup(tweet_id, max_results,
pagination_token)"),
    relief = "flat")

b16.place(
    x = 55, y = 286,
    width = 359,
    height = 18)

img17 = PhotoImage(file =f"win/img17.png")
b17 = Button(

```

```

        image = img17,
        borderwidth = 0,
        highlightthickness = 0,
        command = lambda :
settext("get_liked_tweets(user_id,max_results,pagination_token)"),
        relief = "flat")

b17.place(
    x = 55, y = 304,
    width = 359,
    height = 18)

img18 = PhotoImage(file =f"win/img18.png")
b18 = Button(
    image = img18,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_user_timeline(user_id)"),
    relief = "flat")

b18.place(
    x = 55, y = 322,
    width = 359,
    height = 18)

img19 = PhotoImage(file =f"win/img19.png")
b19 = Button(
    image = img19,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_user_mention_timeline(user_id)"),
    relief = "flat")

b19.place(
    x = 55, y = 340,
    width = 359,
    height = 18)

img20 = PhotoImage(file =f"win/img20.png")
b20 = Button(
    image = img20,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_retweets_lookup(tweet_id,
pagination_token)"),
    relief = "flat")

b20.place(
    x = 55, y = 358,
    width = 359,
    height = 18)

img21 = PhotoImage(file =f"win/img21.png")
b21 = Button(
    image = img21,
    borderwidth = 0,
    highlightthickness = 0,

```



```

        command = lambda : settext("get_recent_tweet_count(str_query)"),
        relief = "flat")

b21.place(
    x = 55, y = 375,
    width = 359,
    height = 18)

img22 = PhotoImage(file = f"win/img22.png")
b22 = Button(
    image = img22,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_recent_search(str_query, end_time,
max_results, next_token, since_id, start_time, until_id)"),
    relief = "flat")

b22.place(
    x = 55, y = 393,
    width = 359,
    height = 18)

img23 = PhotoImage(file = f"win/img23.png")
b23 = Button(
    image = img23,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_quote_tweet(tweet_id, max_results,
pagination_token)"),
    relief = "flat")

b23.place(
    x = 55, y = 411,
    width = 359,
    height = 18)

img24 = PhotoImage(file = f"win/img24.png")
b24 = Button(
    image = img24,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_list_tweets(list_id, max_results,
pagination_token)"),
    relief = "flat")

b24.place(
    x = 55, y = 429,
    width = 359,
    height = 18)

img25 = PhotoImage(file = f"win/img25.png")
b25 = Button(
    image = img25,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_space_lookup(space_id)"),
    relief = "flat")

```

```

b25.place(
    x = 55, y = 474,
    width = 359,
    height = 18)

img26 = PhotoImage(file =f"win/img26.png")
b26 = Button(
    image = img26,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_spaces(space_ids)"),
    relief = "flat")

b26.place(
    x = 55, y = 492,
    width = 359,
    height = 18)

img27 = PhotoImage(file =f"win/img27.png")
b27 = Button(
    image = img27,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_spaces_by_creator_ids(user_ids)"),
    relief = "flat")

b27.place(
    x = 55, y = 510,
    width = 359,
    height = 18)

img28 = PhotoImage(file =f"win/img28.png")
b28 = Button(
    image = img28,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_mutes_lookup(user_id, max_results,
pagination_token)"),
    relief = "flat")

b28.place(
    x = 55, y = 528,
    width = 359,
    height = 18)

img29 = PhotoImage(file =f"win/img29.png")
b29 = Button(
    image = img29,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda :
settext("user_owned_list(user_id,max_results,pagination_token)"),
    relief = "flat")

b29.place(
    x = 55, y = 546,

```

```

        width = 359,
        height = 18)

img30 = PhotoImage(file =f"win/img30.png")
b30 = Button(
    image = img30,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_pinned_list(user_id)"),
    relief = "flat")

b30.place(
    x = 55, y = 564,
    width = 359,
    height = 18)

img31 = PhotoImage(file =f"win/img31.png")
b31 = Button(
    image = img31,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda :
settext("get_user_list_memberships(user_id,max_results, pagination_token)"),
    relief = "flat")

b31.place(
    x = 55, y = 582,
    width = 359,
    height = 18)

img32 = PhotoImage(file =f"win/img32.png")
b32 = Button(
    image = img32,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_list_member_lookup(list_id, max_results,
pagination_token)"),
    relief = "flat")

b32.place(
    x = 55, y = 600,
    width = 359,
    height = 18)

img33 = PhotoImage(file =f"win/img33.png")
b33 = Button(
    image = img33,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda :
settext("get_user_list_followed(user_id,max_results,pagination_token)"),
    relief = "flat")

b33.place(
    x = 55, y = 618,
    width = 359,
    height = 18)

```

```

img34 = PhotoImage(file =f"win/img34.png")
b34 = Button(
    image = img34,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_list_followers_lookup(list_id,
max_results, pagination_token)"),
    relief = "flat")

b34.place(
    x = 55, y = 636,
    width = 359,
    height = 18)

img35 = PhotoImage(file =f"win/img35.png")
b35 = Button(
    image = img35,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda : settext("get_following_lookup(user_id, max_results,
pagination_token)"),
    relief = "flat")

b35.place(
    x = 55, y = 202,
    width = 359,
    height = 18)

img36 = PhotoImage(file =f"win/img36.png")
b36 = Button(
    image = img36,
    borderwidth = 0,
    highlightthickness = 0,
    command = lambda :
settext("get_block_lookup(user_id,max_results,pagination_token)"),
    relief = "flat")

b36.place(
    x = 55, y = 654,
    width = 359,
    height = 18)

window.resizable(False, False)
window.mainloop()

```