

# Notes Assignment

- **Scraping with Twitter :**

Step1: start by importing libraries in python :

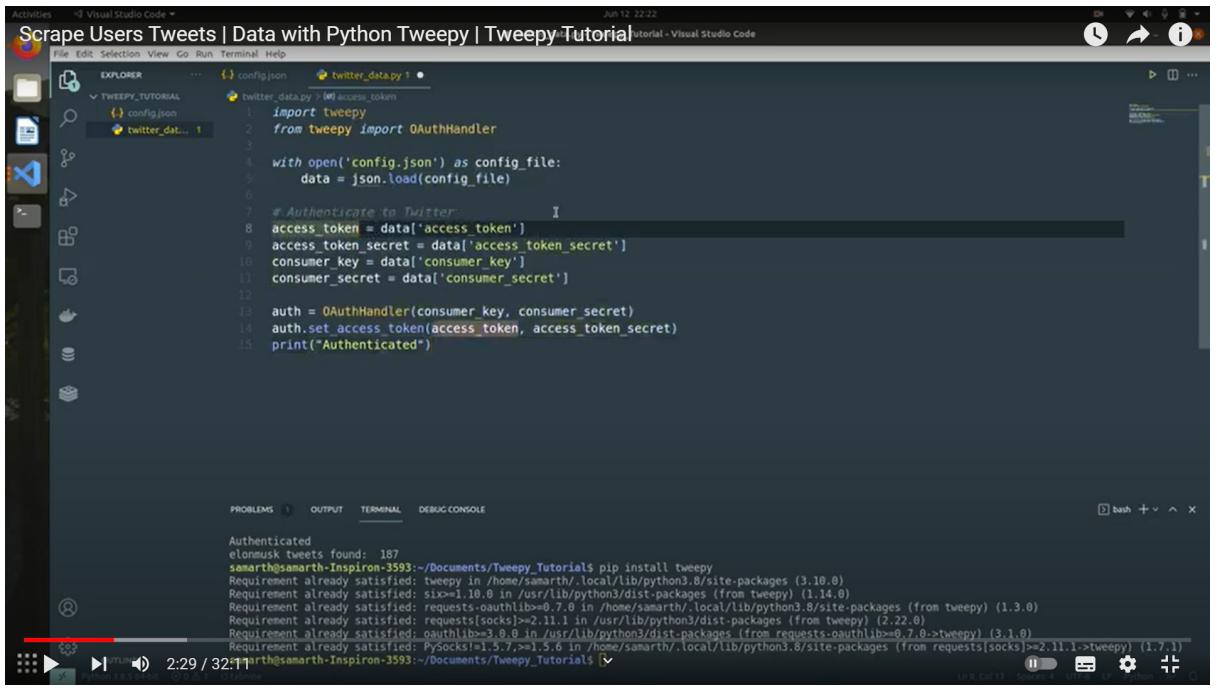
## Import tweepy

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities, Visual Studio Code, July 12, 22:22, tutorial - Visual Studio Code
- File Explorer:** Shows a folder named "TWEETPY\_TUTORIAL" containing "config.json" and "twitter\_data.py". "twitter\_data.py" is open in the editor.
- Editor:** Displays the following Python code:

```
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import json
import time
import requests
import socket
import sys
import os
import re
import csv
import io
import pandas as pd
import numpy as np
import tensorflow as tf
import tensorflow.python
import tensorflow.python.
```
- Bottom Status Bar:** bash + v x
- Terminal:** Shows the output of a pip install command for tweepy, listing various dependencies like requests, oauthlib, and tweepy itself.

Step 2 : we need access token, access token secret, consumer key, consumer secret key.



The screenshot shows the Visual Studio Code interface with a dark theme. In the Explorer sidebar, there's a folder named 'TWEETPY\_TUTORIAL' containing 'config.json' and 'twitter\_data.py'. The 'twitter\_data.py' file is open in the editor, displaying Python code for authenticating to Twitter using Tweepy. The code reads a configuration file, extracts access tokens and secrets, and creates an OAuthHandler object. A terminal window at the bottom shows the command 'pip install tweepy' being run, along with its output, which includes dependency requirements like requests and oauthlib.

```

import tweepy
from tweepy import OAuthHandler

with open('config.json') as config_file:
    data = json.load(config_file)

# Authenticate to Twitter
access_token = data['access_token']
access_token_secret = data['access_token_secret']
consumer_key = data['consumer_key']
consumer_secret = data['consumer_secret']

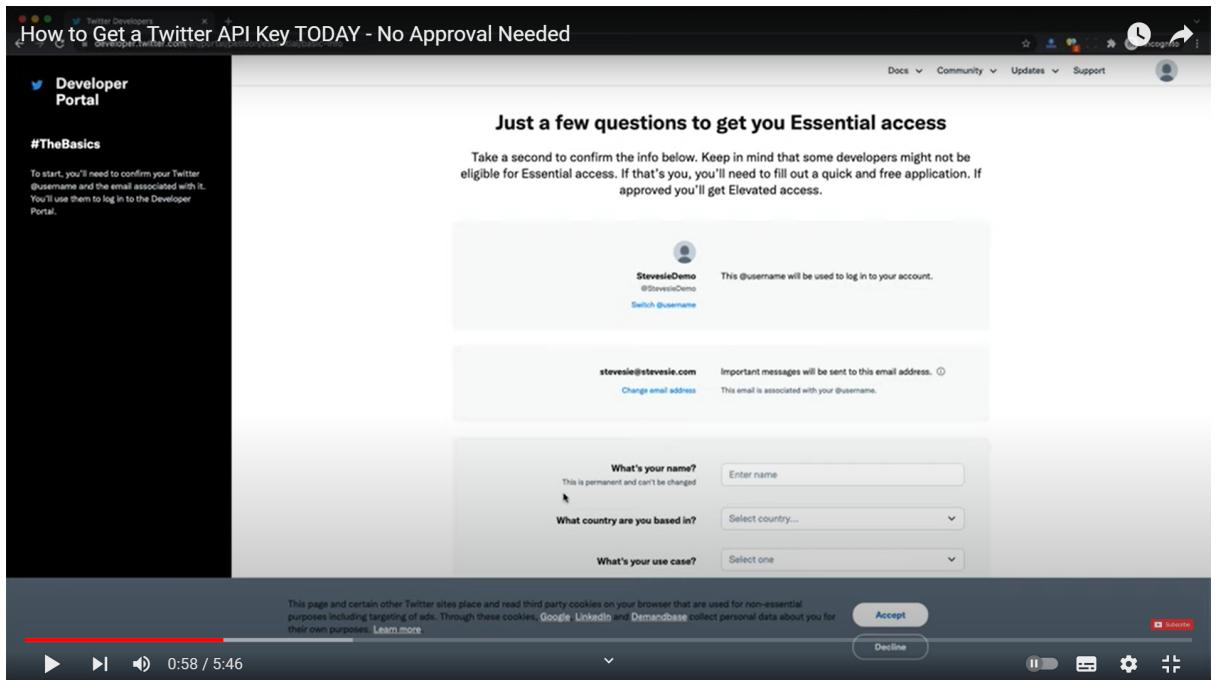
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
print("Authenticated")

```

To get these keys we need access of the twitter handling account i.e developer access for twitter account.

### 3. Step 3 : Getting access of developer account of twitter

Sign Up to developer account



Answer questions related purpose of your access.

Once done we get the api keys save the keys on notepad as these are visible for one time.

**Step 4**: after getting the api keys store it in variables and authenticate the keys .

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities, Visual Studio Code, Jun 12 22:29, Scrape Users Tweets | Data with Python Tweepy | Tweepy-Tutorial
- File Explorer:** Shows a folder structure for 'TWEETPY\_TUTORIAL' containing 'config.json' and 'twitter\_data.py'. The 'twitter\_data.py' file is open.
- Code Editor:** Displays the following Python code:

```
from tweepy import OAuthHandler
import json

with open('config.json') as config_file:
    data = json.load(config_file)

# Authenticate to Twitter
access_token = data['access_token']
access_token_secret = data['access_token_secret']
consumer_key = data['consumer_key']
consumer_secret = data['consumer_secret']

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
print("Authenticated")
I
```

- Terminal:** Shows the command line output:

```
/bin/python3 /home/samarth/Documents/Tweepy_Tutorial/twitter_data.py
samarth@samarth-Inspiron-3593:~/Documents/Tweepy_Tutorials /bin/python3 /home/samarth/Documents/Tweepy_Tutorial/twitter_data.py
Authenticated
samarth@samarth-Inspiron-3593:~/Documents/Tweepy_Tutorials$
```
- Bottom Status Bar:** Includes icons for play/pause, volume, battery, and settings.

Once we are done with the authentication then we start the scrapping



The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities, Visual Studio Code, Jun 12 22:31, Scrape Users Tweets | Data with Python Tweepy | Tweepy Tutorial - Visual Studio Code
- File Explorer:** Shows files: config.json, twitter\_data.py, and a folder TWEETY\_TUTORIAL containing config.json and another folder TWEETY\_TUTORIAL.
- Code Editor:** Displays Python code for interacting with Twitter using Tweepy. The code includes importing OAuthHandler and json, opening config.json, and authenticating with Twitter using access token, access token secret, and consumer key.
- Terminal:** Shows the command "python twitter\_data.py" being run.
- Output:** Shows the output of the script execution.
- Debug Console:** Shows the output of the script execution.
- Sidebar:** Includes icons for file operations like Open, Save, Find, and others.

```
from tweepy import OAuthHandler
import json

with open('config.json') as config_file:
    data = json.load(config_file)

# Authenticate to Twitter
access_token = data['access_token']
access_token_secret = data['access_token_secret']
consumer_key = data['consumer_key']
```

For getting the user we do the following :

```
api=tweepy.API(auth,wait_on_rate_limit=True)
```

After doing this command we get to the api and for scrapping the user we can simply use

```
api.get_user(user)
```

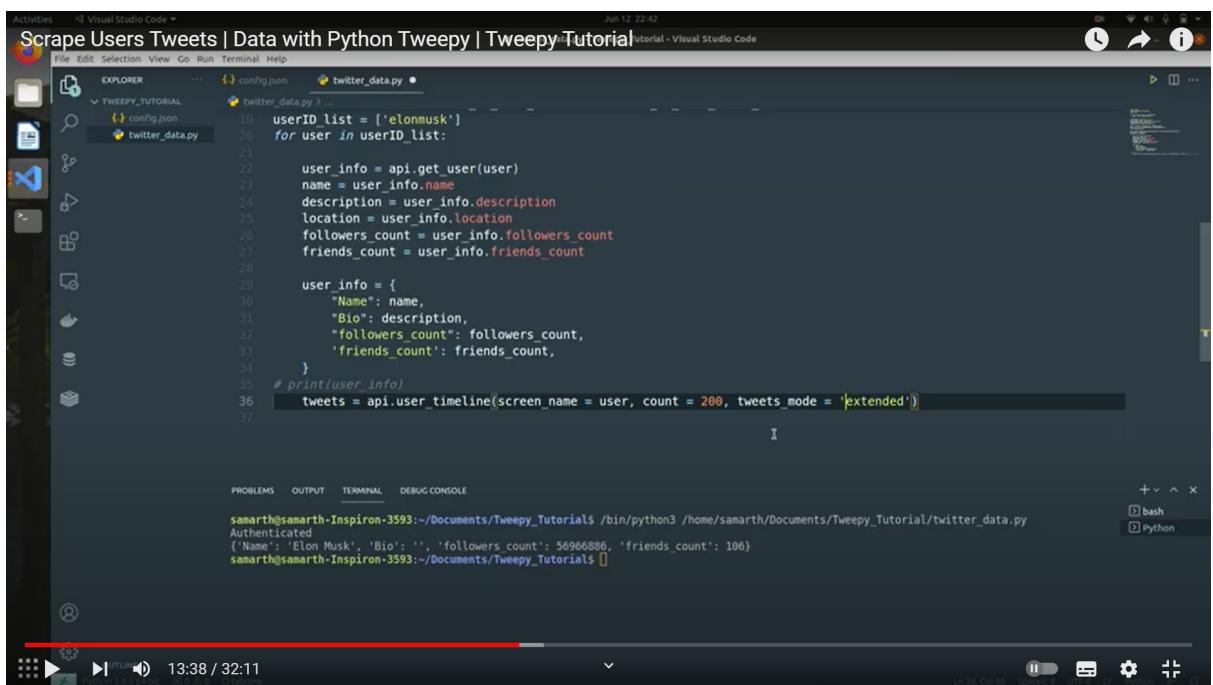
Through this command we get access of information related to user

For description : user\_info.description()

For location : user\_Info.location()

For followers\_count : usier\_info.followers\_count()

For friends \_count : usier\_info..friens\_count()



The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities - Visual Studio Code | Scrape Users Tweets | Data with Python Tweepy | Tweepy Tutorial - Visual Studio Code
- File Explorer:** Shows a folder named "TWEETPY\_TUTORIAL" containing "config.json" and "twitter\_data.py".
- Code Editor:** Displays Python code for scraping user data:

```
19 userID_list = ['elonmusk']
20 for user in userID_list:
21
22     user_info = api.get_user(user)
23     name = user_info.name
24     description = user_info.description
25     location = user_info.location
26     followers_count = user_info.followers_count
27     friends_count = user_info.friends_count
28
29     user_info = {
30         "Name": name,
31         "Bio": description,
32         "followers_count": followers_count,
33         'friends_count': friends_count,
34     }
35     # print(user_info)
36     tweets = api.user_timeline(screen_name = user, count = 200, tweets_mode = 'extended')
```
- Terminal:** Shows the output of running the script:

```
samarth@samarth-Inspiron-3593:~/Documents/Tweepy_Tutorials /bin/python3 /home/samarth/Documents/Tweepy_Tutorial/twitter_data.py
Authenticated
{'Name': 'Elon Musk', 'Bio': '', 'followers_count': 56966886, 'friends_count': 106}
samarth@samarth-Inspiron-3593:~/Documents/Tweepy_Tutorials [ ]
```

We can see what we can access on developer docs :

The screenshot shows a Mozilla Firefox browser window displaying the Twitter Developer Platform at <https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/overview>. The page is titled "Scrape Users Tweets | Data with Python Tweepy | Tweepy Tutorial". The left sidebar shows navigation categories like "Getting started", "Tutorials", "Tools and libraries", "Migrate", "API reference index", "The new Twitter API" (selected), "Fundamentals", "Tweets", "Users", and "Enterprise". The main content area discusses the "Important notice" regarding rate limits and provides a table of API endpoints and their descriptions:

API endpoint	Description
GET statuses / home_timeline	Returns a collection of the most recent Tweets posted by the authenticating user and the users they follow.
GET statuses / user_timeline	Returns a collection of the most recent Tweets posted by the indicated by the screen_name or user_id parameters.
GET statuses/mentions_timeline	Returns the 20 most recent mentions (Tweets containing a user's @handle) for the authenticating user.

At the bottom, there is a poll asking "Was this document helpful?" with two smiley face icons.

Scraping for date the tweet was posted , what tweet was posted , the user who posted this tweet can be fetched through the api