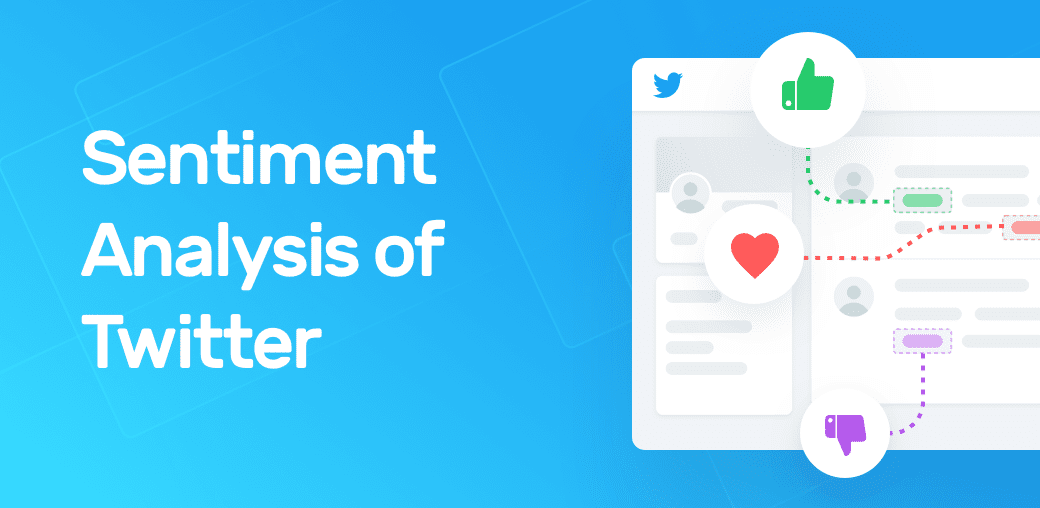
**INTRODUCTION:**

**What is Opinion Analysis?**

Opinion analysis, also known as **sentiment analysis**, is a computational technique used to extract and classify people’s opinions, emotions, and attitudes from text data. In the context of social media, it helps us understand what people think or feel about a particular topic, product, person, or event.  
In the digital age, social media platforms have become integral to public communication, opinion sharing, and trendsetting. Twitter, with millions of users tweeting daily about personal thoughts, current events, brands, politics, and more, provides a massive repository of real-time data. Analyzing this data can offer valuable insights into public sentiment and opinion dynamics.  
This project focuses on opinion analysis, a subfield of Natural Language Processing (NLP), which involves classifying textual content into sentiment categories such as positive, negative, or neutral. By applying opinion analysis to real-time tweets, we can assess public attitudes toward a specific keyword, product, person, or event.  
This project aims to implement an opinion analysis system in Python that utilizes the Twitter API to collect tweets based on user input and analyzes them using simple keyword-based logic.



### Why Use Twitter?

Twitter is a **microblogging platform** where users regularly post short messages (“tweets”) about current events, brands, and social issues. With over 500 million tweets shared daily, it offers a **real-time, large-scale source of public opinion**.

Key reasons to choose Twitter for opinion analysis:

* Tweets are **publicly available** and frequently updated.
* Topics range from **local news** to **global trends**.
* The data is **diverse, real-time, and opinion-rich**.

**OBJECTIVES:**

* **To collect real-time data** from Twitter based on a user-specified keyword using the Twitter API v2.
* **To classify tweets** as Positive, Negative, or Neutral based on sentiment word occurrences.
* **To display results** in a readable format for easy interpretation.

.

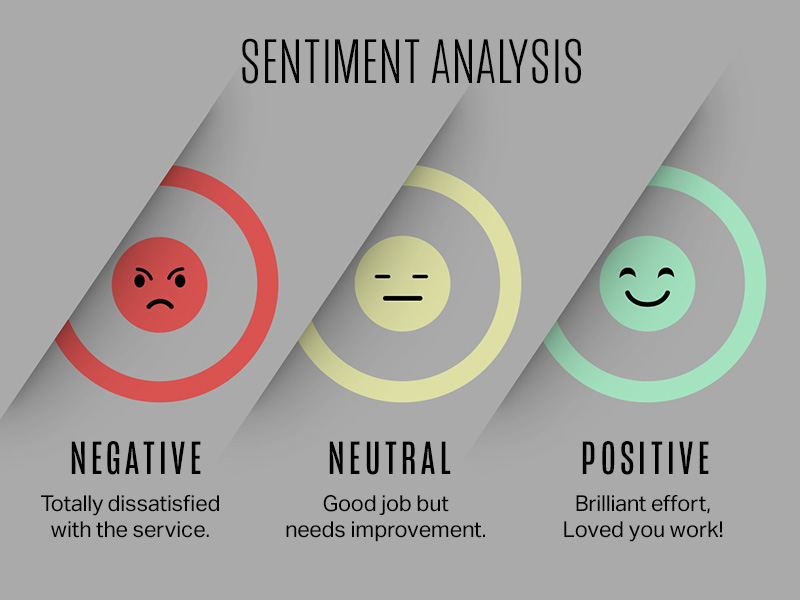
**METHODOLOGY:**

#### Tools & Technologies Used

* **Tweepy(CO2)**: A Python library that allows easy access to the Twitter API (v2).
* **TextBlob(CO3)**: A Python library for processing textual data, offering a simple API for NLP tasks like sentiment analysis.
* **Twitter API v2(CO1)**: Provided by Twitter Developer Platform, used to access recent tweet data.

#### System Flow

1. **User Input**: The user provides a keyword or hashtag to analyze.
2. **API Connection**: The bot uses the Tweepy Client object to connect to Twitter using a Bearer Token.
3. **Tweet Retrieval**: Using Paginator from Tweepy, it fetches a fixed number of recent tweets that match the keyword, excluding retweets and non-English content.
4. **Text Processing & Sentiment Classification**:
   * Each tweet's content is analyzed using TextBlob.
   * TextBlob assigns a **polarity score** (ranging from -1 to 1):
     + **> 0** = Positive
     + **< 0** = Negative
     + **= 0** = Neutral
5. **Sentiment Tallying**: Counts of each sentiment category are maintained using a dictionary.
6. **Rate Limit Handling**: If rate limits are encountered, the bot waits before resuming (Tweepy’s wait\_on\_rate\_limit=True ensures this automatically).
7. **Output**: Each tweet is printed along with its sentiment. A final sentiment summary is displayed.



#### Code Explanation

Here is a summary of the key sections of the code:

* get\_sentiment(text): Takes tweet text and returns the sentiment category based on TextBlob’s polarity score.
* analyze\_tweets(keyword, max\_tweets): Core function that fetches tweets, performs sentiment analysis, and prints the results.
* if \_\_name\_\_ == "\_\_main\_\_": Accepts user input for the keyword and initiates the analysis process.

**SOURCE CODE:**

import tweepy

from textblob import TextBlob

import time

BEARER\_TOKEN = 'AAAAAAAAAAAAA---------------------------------gBH4d5q----------------lBzivHKzN-------------------NPhr2q4m'

client = tweepy.Client(bearer\_token=BEARER\_TOKEN, wait\_on\_rate\_limit=True)

def get\_sentiment(text):

    blob = TextBlob(text)

    polarity = blob.sentiment.polarity

    if polarity > 0:

        return 'Positive'

    elif polarity < 0:

        return 'Negative'

    else:

        return 'Neutral'

def analyze\_tweets(keyword, max\_tweets=50):

    query = f"{keyword} -is:retweet lang:en"

    sentiment\_count = {'Positive': 0, 'Negative': 0, 'Neutral': 0}

    tweets\_analyzed = 0

    try:

        for tweet in tweepy.Paginator(client.search\_recent\_tweets, query=query, tweet\_fields=['text'], max\_results=10).flatten(limit=max\_tweets):

            sentiment = get\_sentiment(tweet.text)

            sentiment\_count[sentiment] += 1

            tweets\_analyzed += 1

            print(f"[{sentiment}] {tweet.text}\n")

    except tweepy.TooManyRequests:

        print("Rate limit reached. Waiting before retrying...")

        time.sleep(900)

    except Exception as e:

        print("Error:", e)

    print("\nSentiment Summary:")

    print(sentiment\_count)

    print(f"Total Tweets Analyzed: {tweets\_analyzed}")

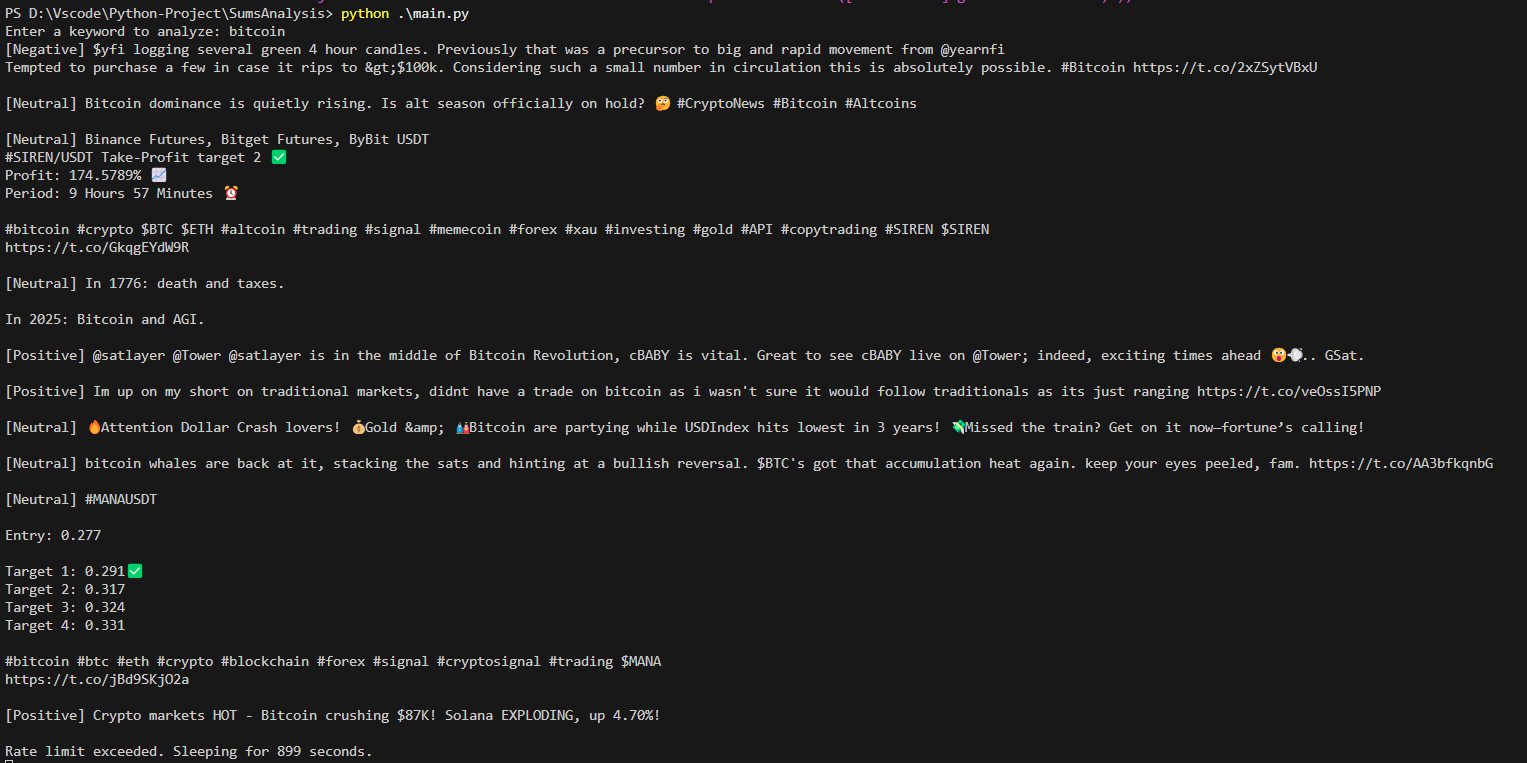
if \_\_name\_\_ == "\_\_main\_\_":

    keyword = input("Enter a keyword to analyze: ")

    analyze\_tweets(keyword)

### OUTPUT:

Keyword - Bitcoin



 **Neutral**: 6 tweets

 **Positive**: 3 tweets

 **Negative**: 1 tweet

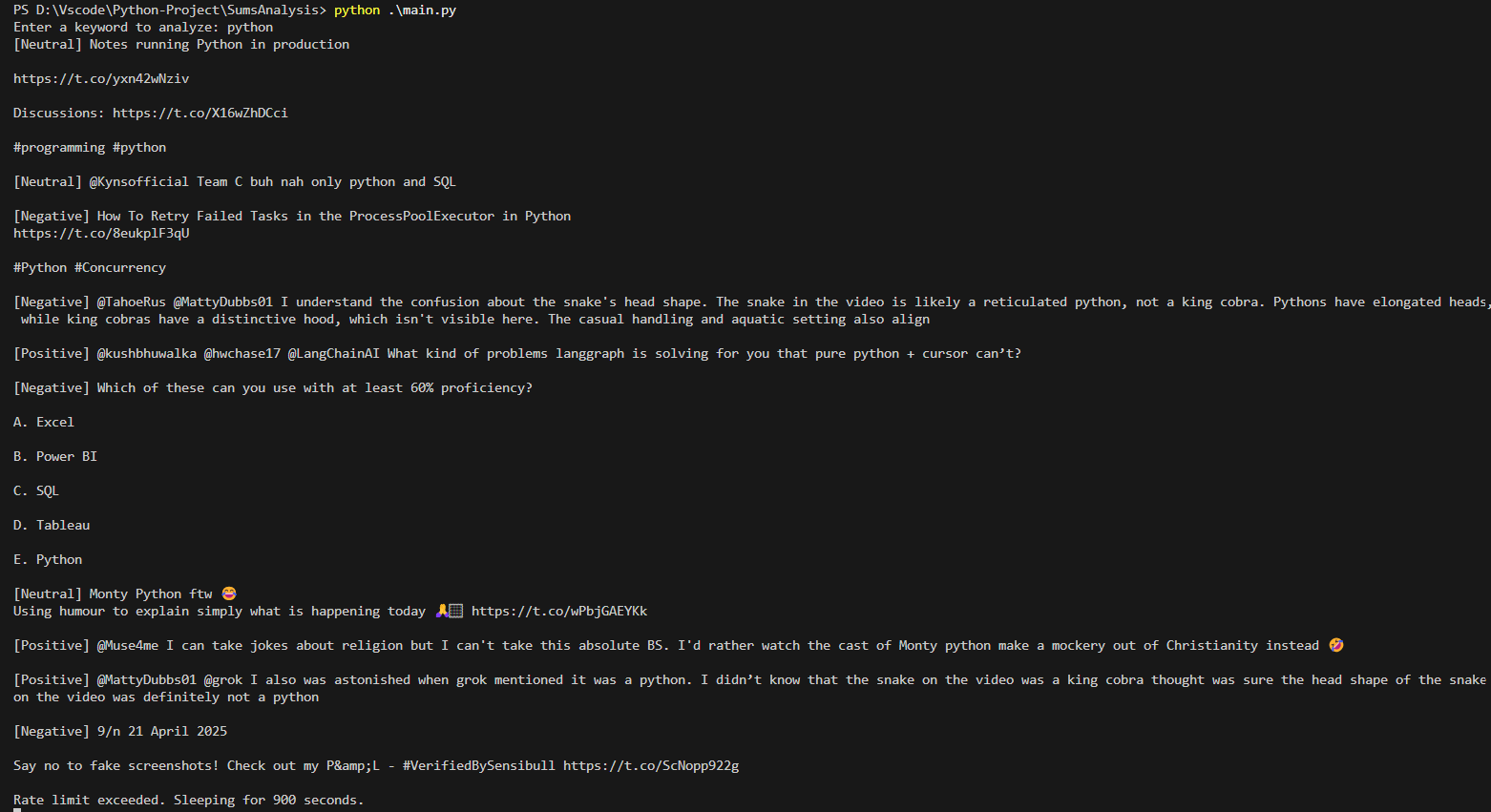
GRAPH REPRESENTATION OF THE OUTPUT (CO6):

### 

The chart gives a quick visual of how people are feeling about "bitcoin" in the analyzed tweets

### ANOTHER OUTPUT:

Keyword - Python



 **Positive**: 3 tweets

 **Negative**: 4 tweets

 **Neutral**: 3 tweets

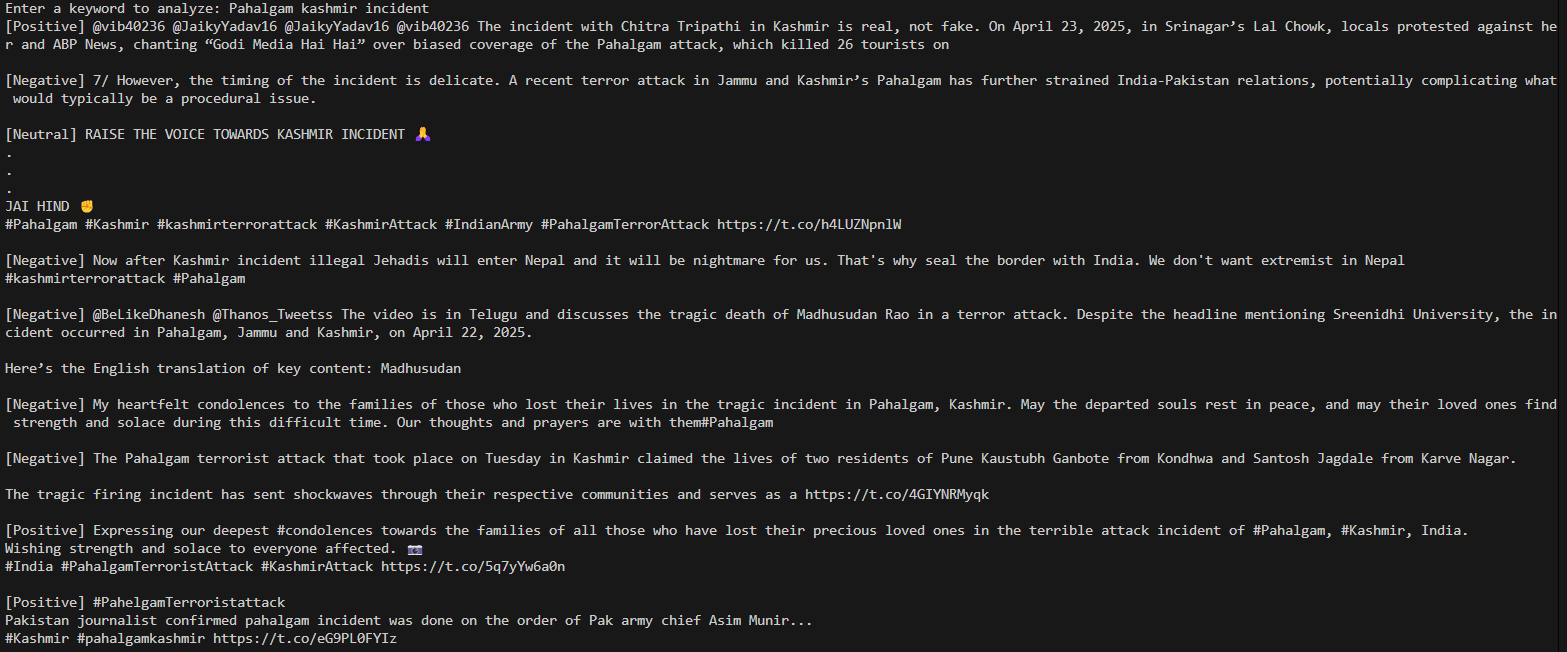
GRAPH REPRESENTATION OF THE OUTPUT(CO6):

### 

The chart gives a quick visual of how people are feeling about "python" in the analyzed tweets

### ANOTHER OUTPUT:

Keyword – Pahalgam Kashmir incident



### 

 **Positive**: 8 tweets

 **Negative**: 4 tweets

 **Neutral**: 3 tweets

GRAPH REPRESENTATION OF THE OUTPUT(CO6):

### 

The chart gives a quick visual of how people are feeling about "python" in the analyzed tweets

### CONCLUSION:

This project successfully demonstrates how Twitter data can be leveraged to analyze public sentiment on a specific topic using modern data tool. we created a pipeline that:

* **Collects tweets** in real-time based on a user-defined keyword,
* **Analyzes opinion polarity** using natural language processing (NLP),
* And **lays the foundation** for future visualization and reporting
* Provide a clear summary of the sentiment distribution among the fetched tweets.

This system enables both technical and non-technical users to understand public opinion trends without manually reading thousands of tweets. It also showcases the potential of combining social media data with sentiment analysis for applications like:

* Market research
* Political opinion tracking
* Brand monitoring
* Event feedback and crisis management

Overall, this project offers a simple but powerful framework that can be scaled with improvements such as advanced NLP models, real-time dashboards, or machine learning-based opinion classifiers.

### REFERENCES:

1. **Twitter Developer Platform** – Official documentation for accessing Twitter’s API using Tweepy  
   <https://developer.twitter.com/en/docs>
2. **Tweepy Library Documentation** – Python library for interacting with the Twitter API  
   https://docs.tweepy.org/en/stable/
3. **TextBlob Documentation** – Simplified text processing and sentiment analysis in Python  
   <https://textblob.readthedocs.io/en/dev/>
4. **Python Official Documentation** – General reference for Python programming  
   <https://docs.python.org/3/>
5. **Github Link** – (code idea from) – <https://github.com/ujjwalkarn/Twitter-Sentiment-Analysis>
6. Google Gemini – (for graph representation) – https://gemini.google.com