Logo

Description automatically generated**EAST WEST UNIVERSITY**

**CSE477**

**Section: 02**

**Lab: 01 Report**

**Topic: Exploratory Data Analysis on YouTube Data**

**Submitted By:**

**Name: Md Sifatullah Sheikh**

**ID: 2022-1-60-029**

**Submitted To:**

**Amit Mandal**

**Lecturer**

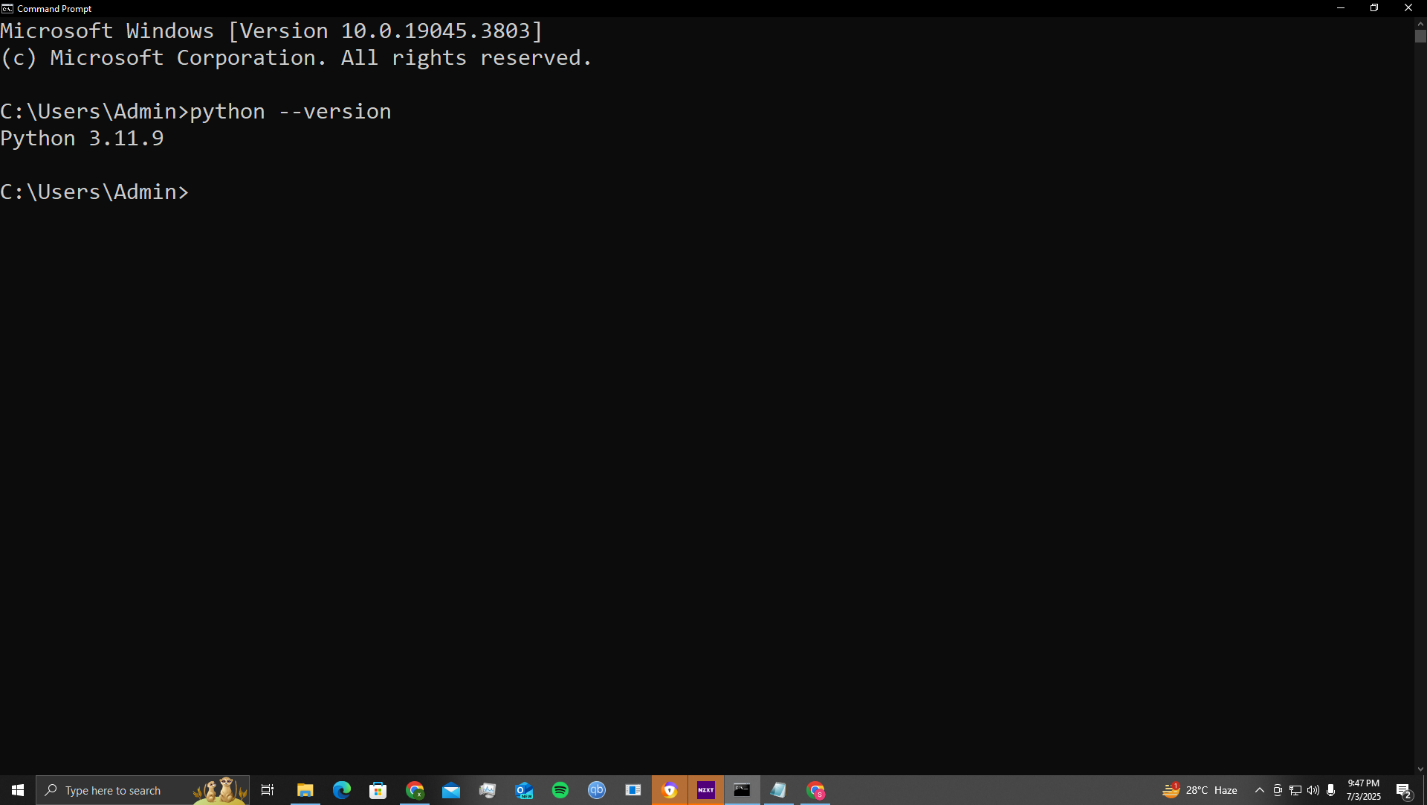
**Department of Computer Science & Engineering**

**Date: 5 July 2025**

# **PART 1: Getting Ready (Tools, Setup, and Verified References)**

**A. Install Python:**  I already have Python, and the version is **Python 3.11.9**

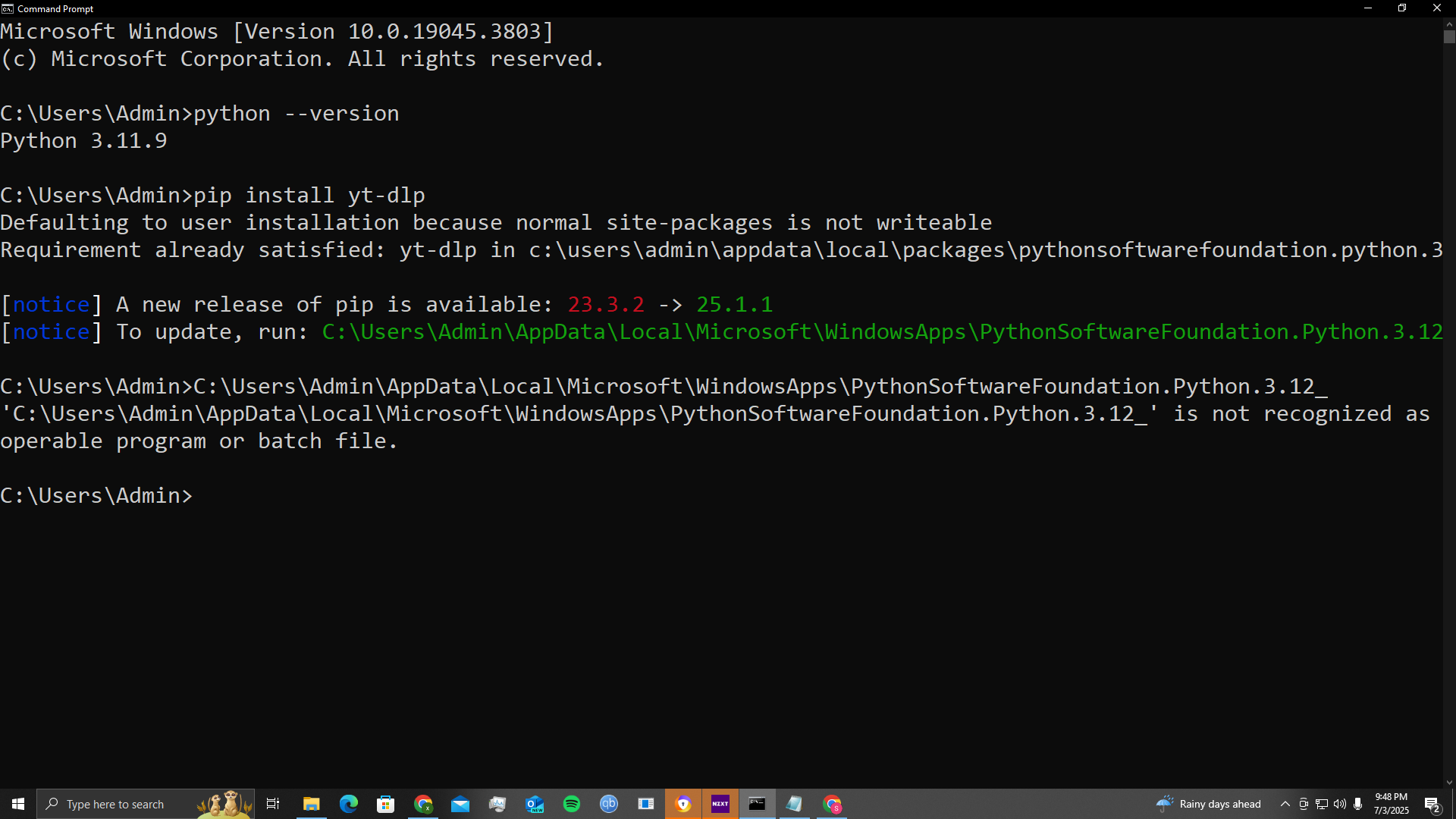
**Command: python --version**

******

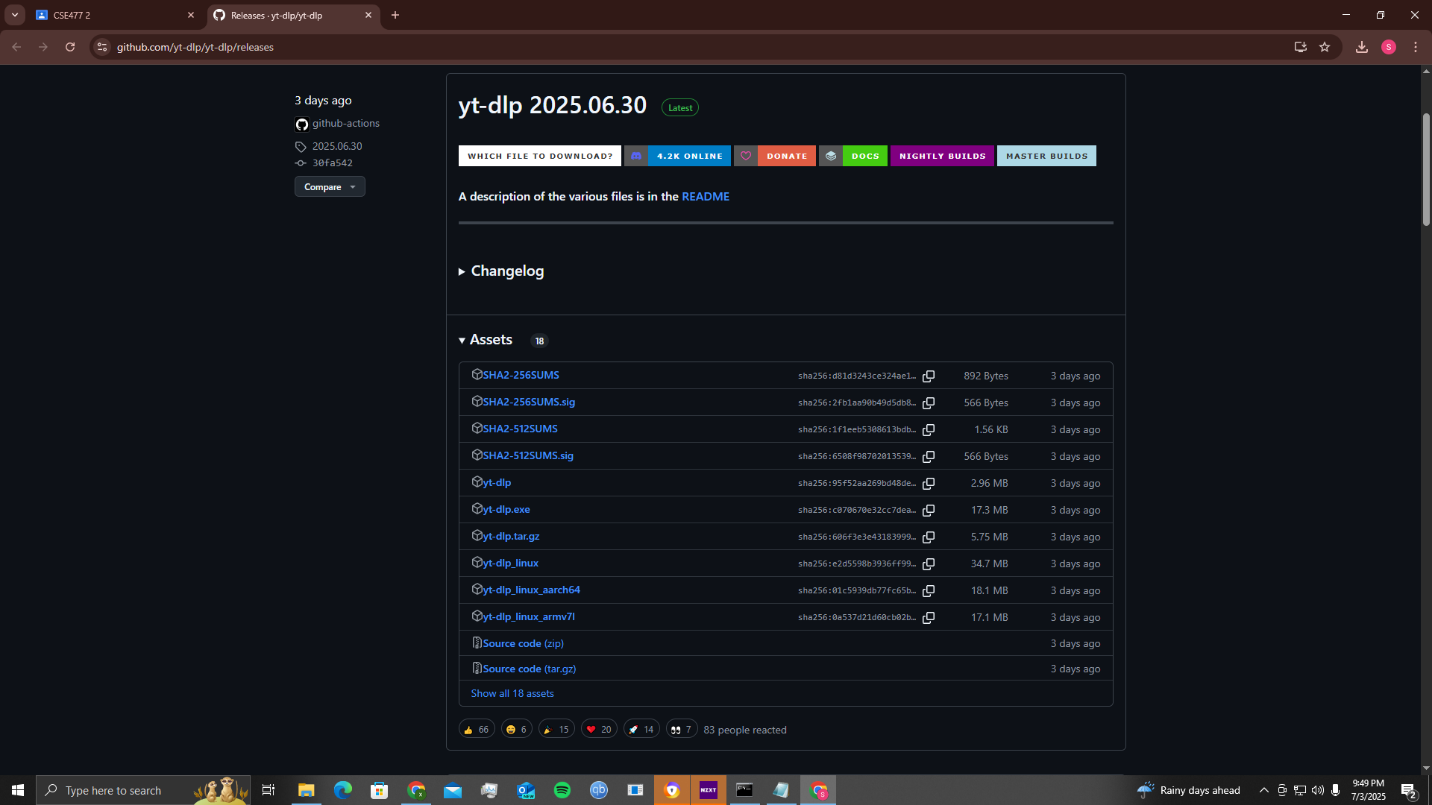
**B. Install yt-dlp (Both Methods, All Verified):**

**Method 1 (Install with pip):** I already have yt-dlp on my computer environment.

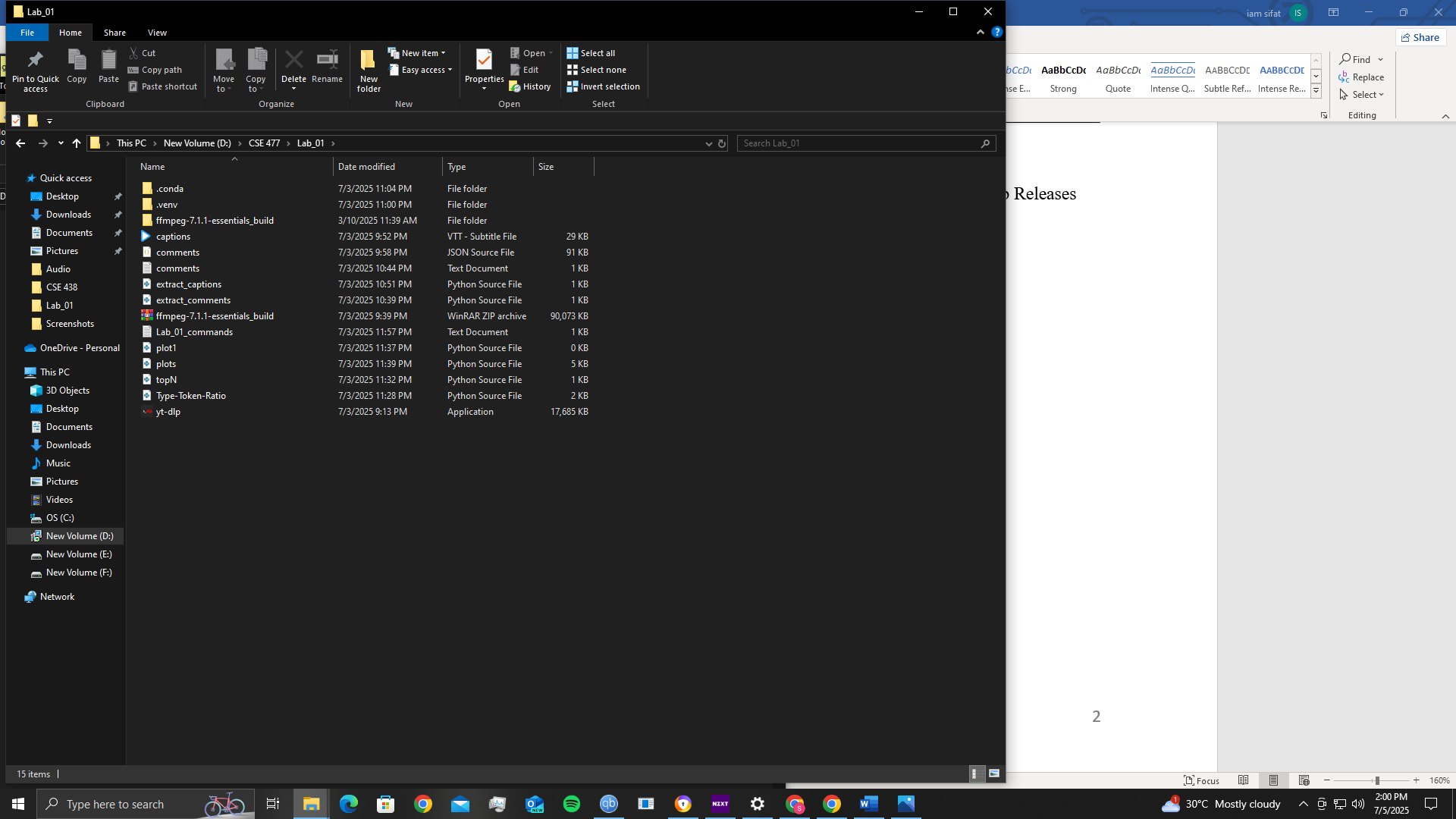
**Command: pip install yt-dlp**

****

**Method 2:** Download Standalone Executable: Download **yt-dlp.exe** from the official GitHub Releases page: <https://github.com/yt-dlp/yt-dlp/releases>

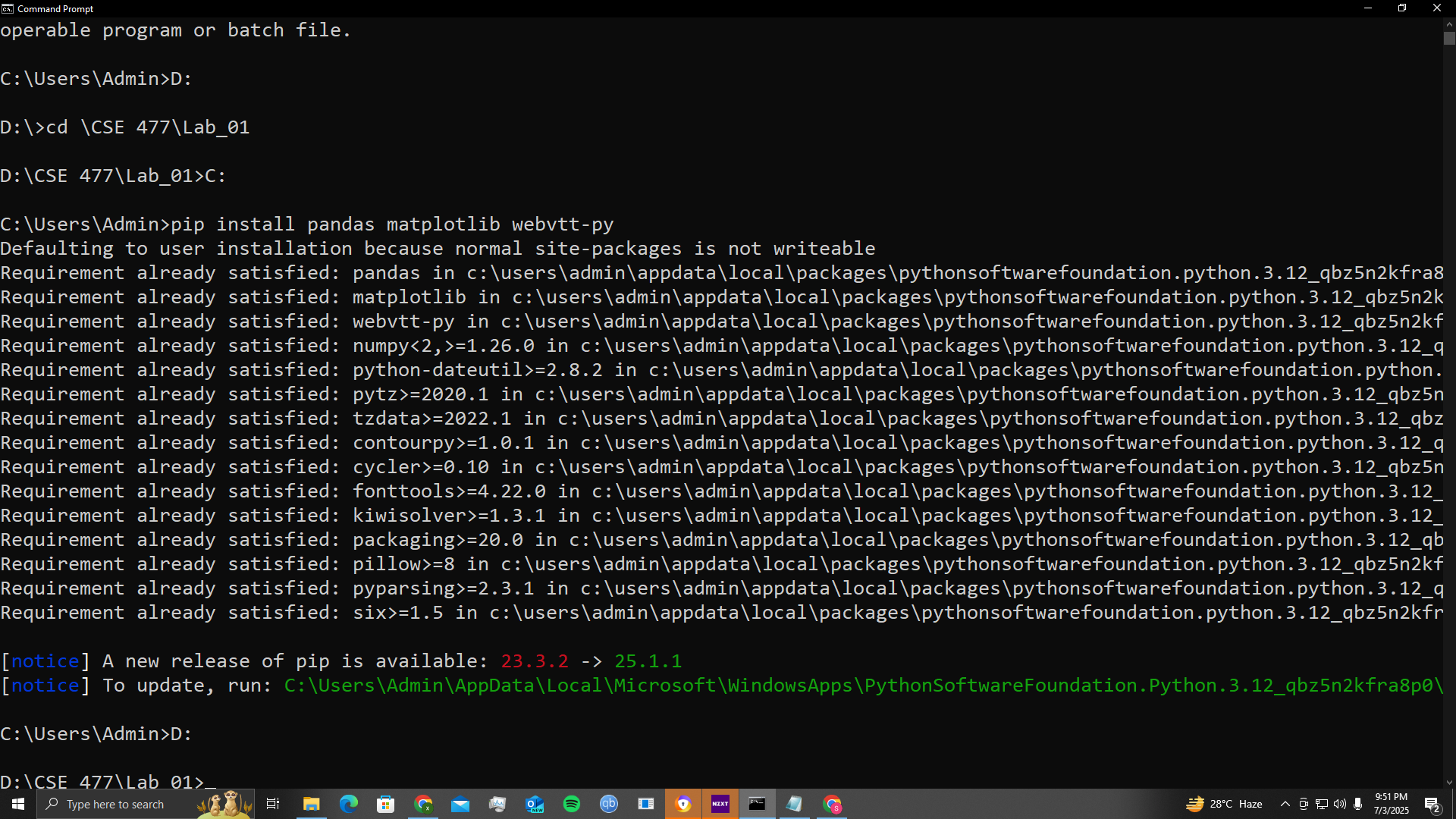


* Download it to a particular workable folder.



**C. Install Python Libraries:** Change to the particular folder disk path, which is **D:**

**Command:** pip install pandas matplotlib webvtt-py



# **PART 2: Data Collection &amp; Selection (Channels, Ethics, and Procedures)**

**A. Selecting a YouTube Channel/Video (All Channels Verified Neutral) Criteria:** I chose a video, and it is from **BBC Earth**

**YouTube link:** <https://www.youtube.com/watch?v=T7oExc711xE>

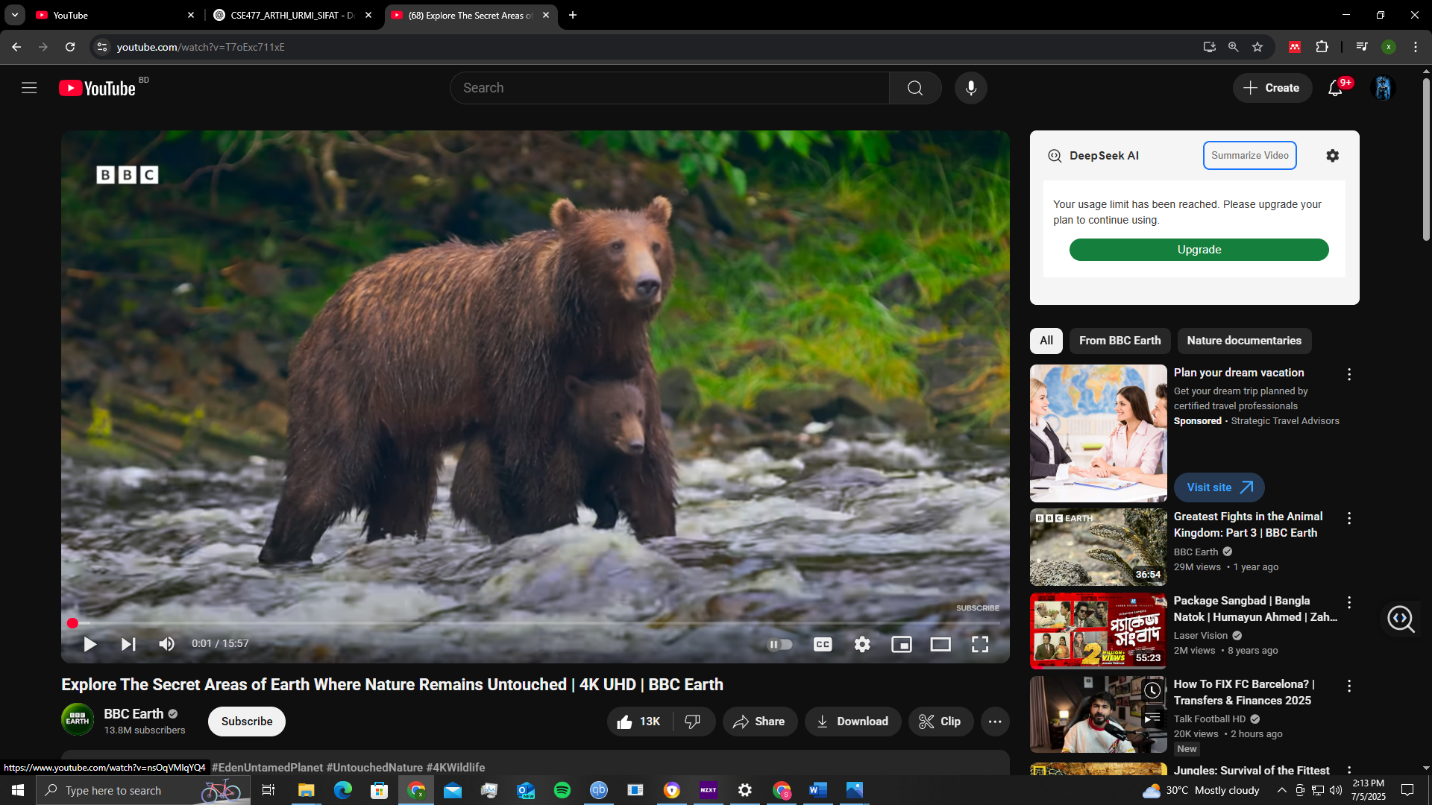
**Video Description:**

**“Explore The Secret Areas of Earth Where Nature Remains Untouched | 4K UHD | BBC Earth”**

1. Time length is 15.57 mins

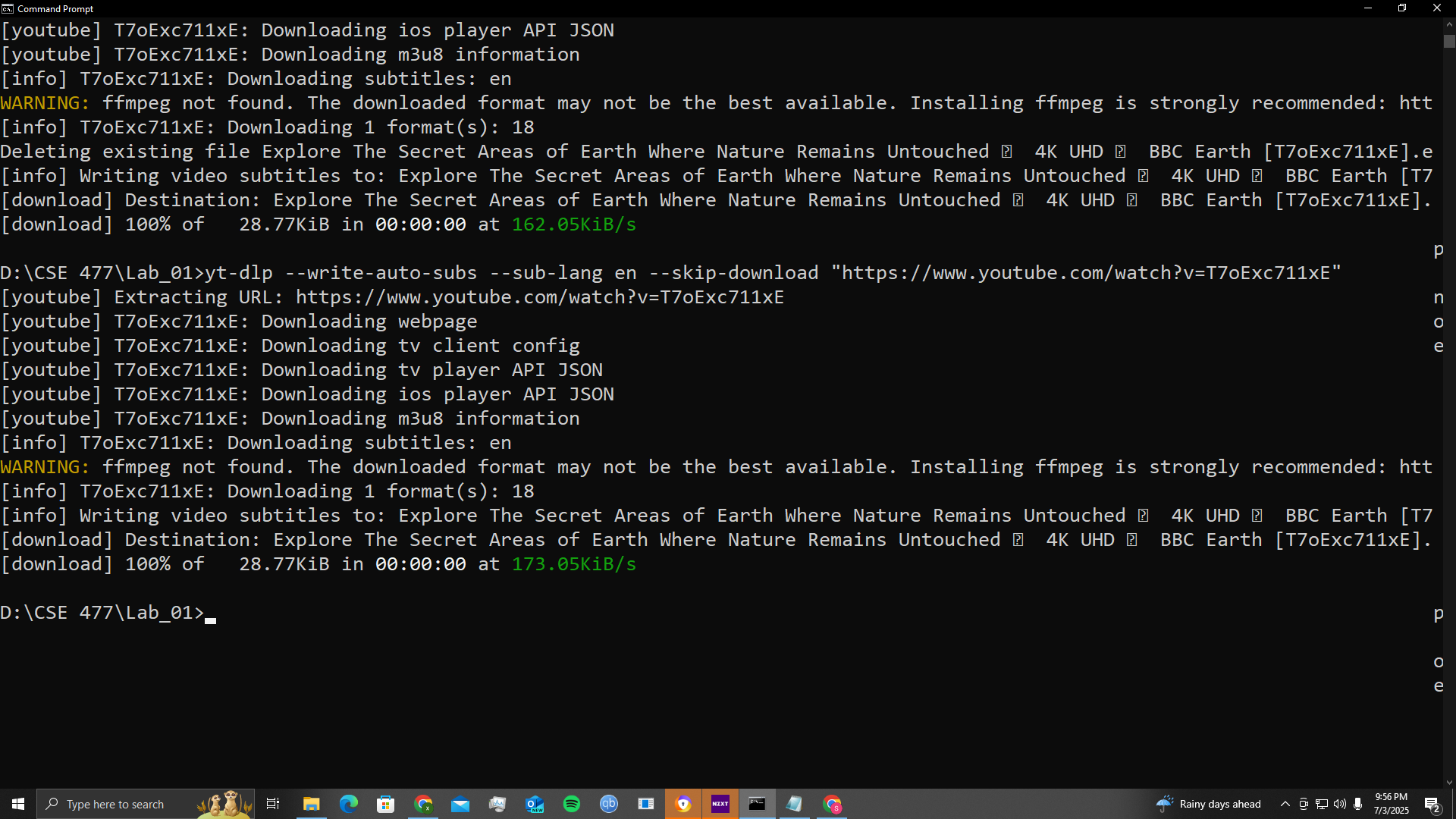
2. 258 comments

3. CC English generated

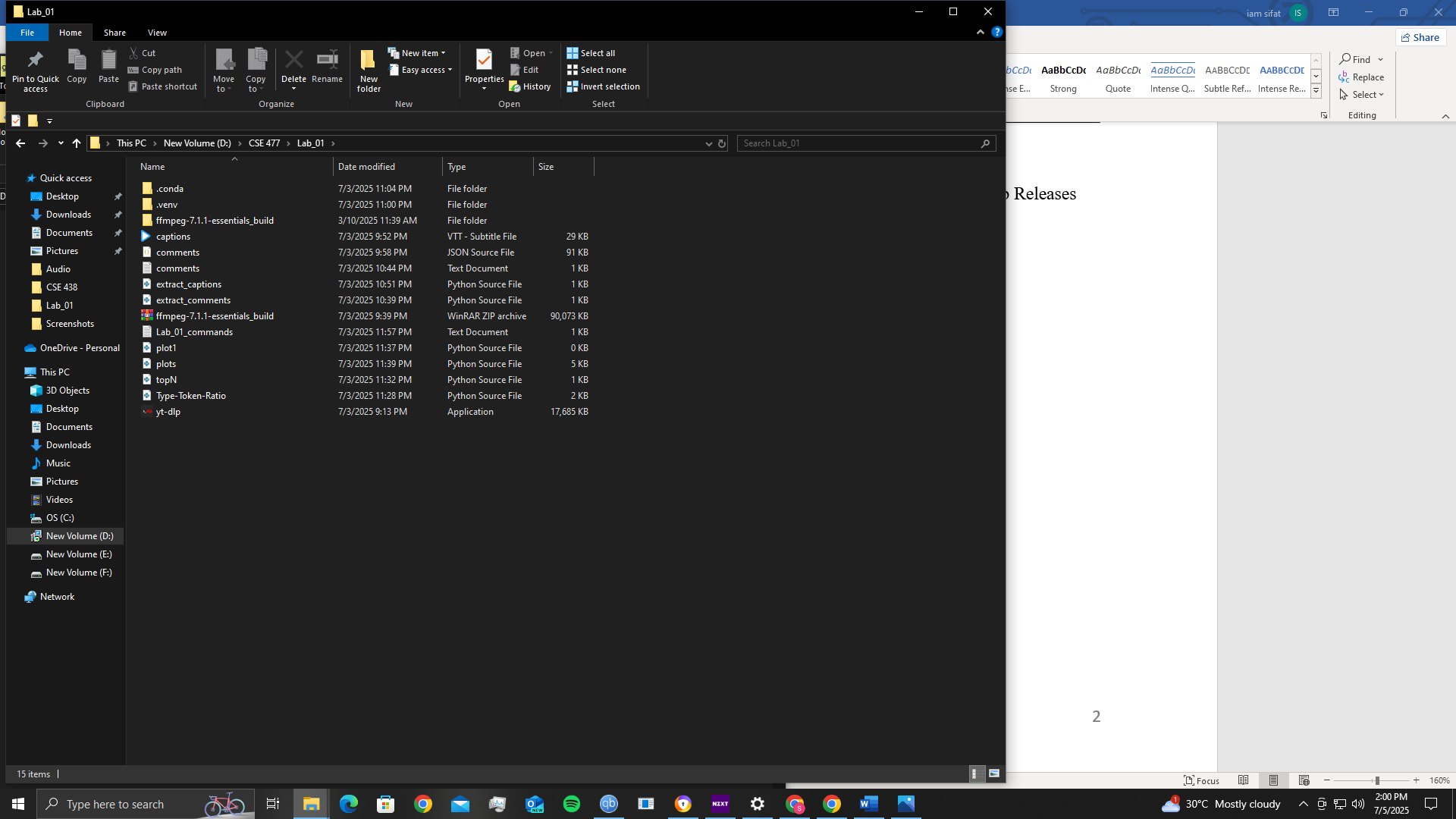


**B. Download Captions (Subtitles) Using yt-dlp:**

**Command**: yt-dlp --write-auto-subs --sub-lang en --skip-download <https://www.youtube.com/watch?v=T7oExc711xE>

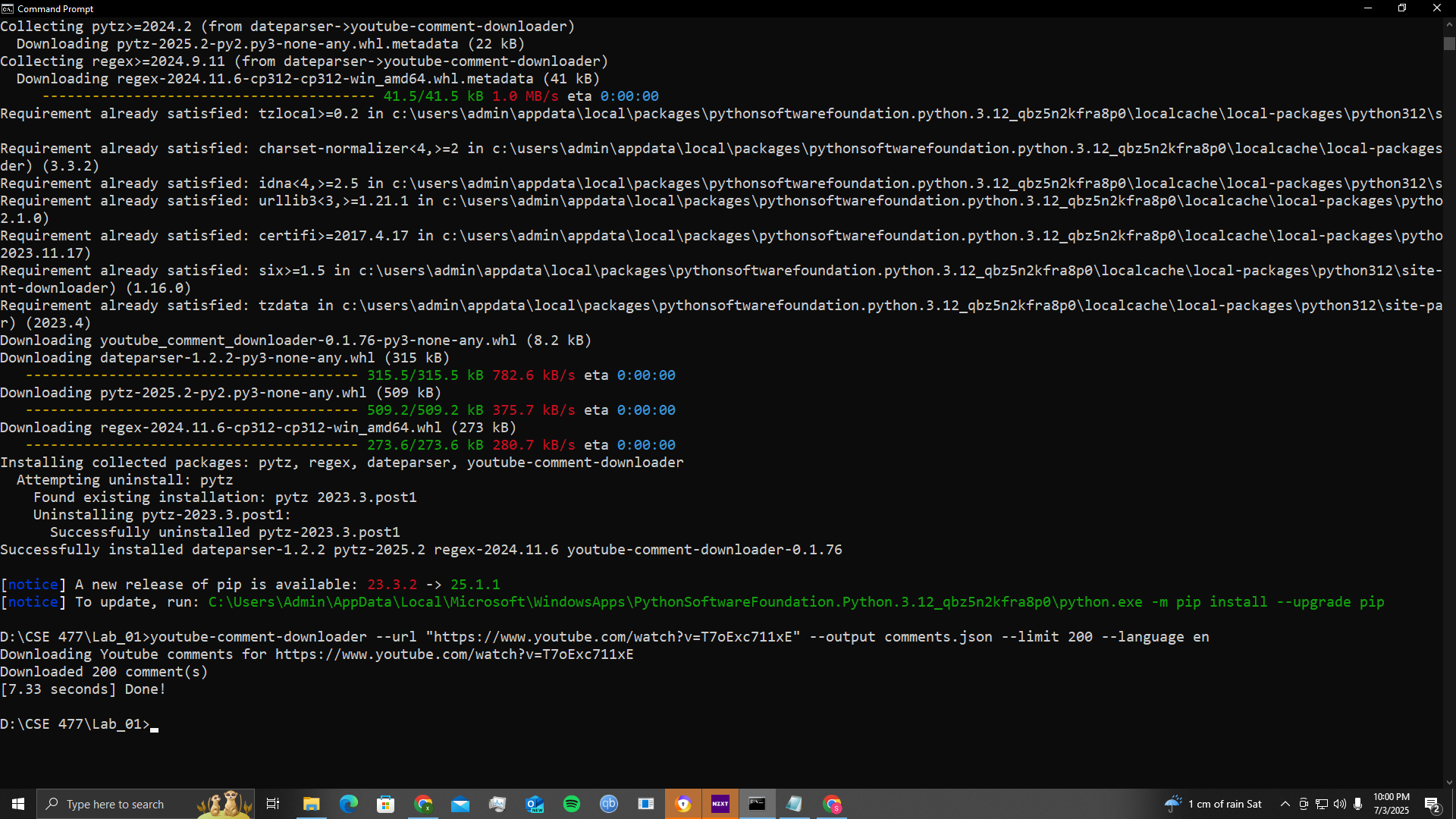


* A VTT file is downloaded. Which I named **captions.VTT**



**C. Downloading Comments Using a Script:**

**Commands:** pip install youtube-comment-downloader

****

# **PART 3: Data Processing &amp; Lab Experiments (Python**

# **Skeletons &amp; Analysis)**

**A. Loading Comments and Captions in Python:**

**Comments:**

import json

def load\_json\_comments(filepath='comments.json'):

comments = []

with open(filepath, 'r', encoding='utf-8') as f:

for line in f:

try:

obj = json.loads(line)

comments.append(obj['text']) # You can also access other fields like obj['time']

except json.JSONDecodeError:

continue # Skip lines that aren't valid JSON

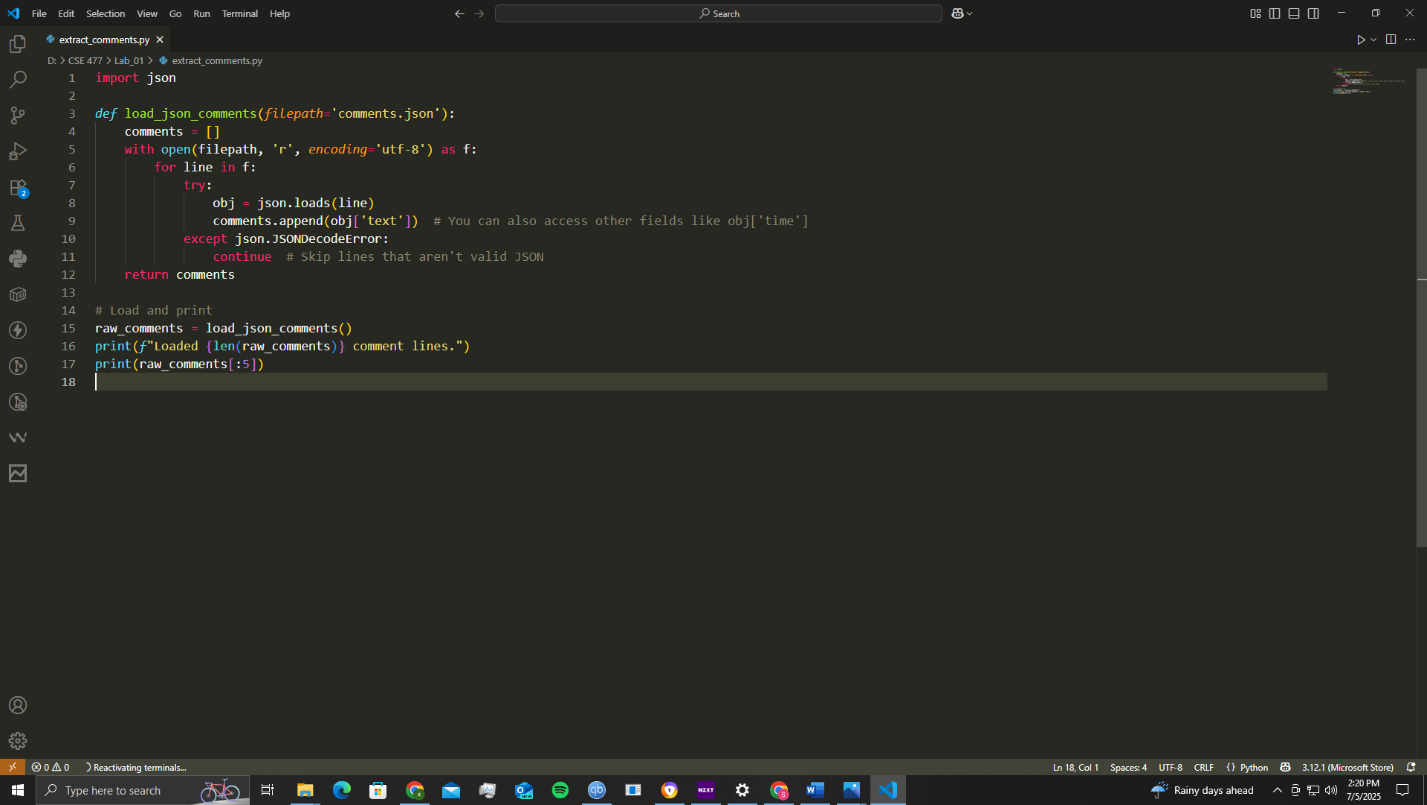
return comments

# Load and print

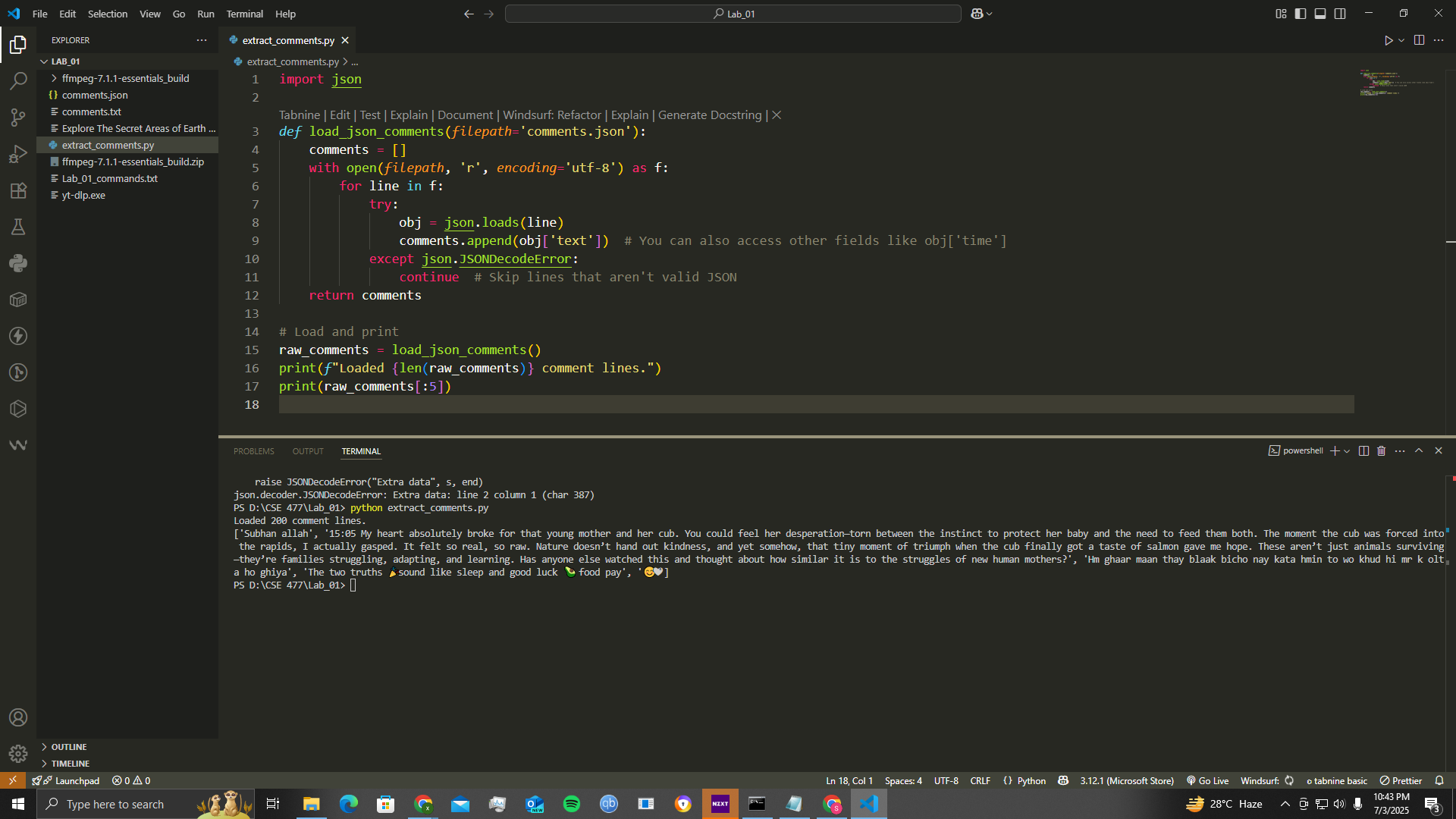
raw\_comments = load\_json\_comments()

print(f"Loaded {len(raw\_comments)} comment lines.")

print(raw\_comments[:5])



**Comments Output :**

****

**Captions:**

def load\_vtt\_captions(filepath='captions.vtt'):

captions = []

with open(filepath, 'r', encoding='utf-8') as f:

for line in f:

line = line.strip()

# Ignore metadata and timestamps

if '-->' not in line and line and not line.isdigit() and 'WEBVTT' not in line:

captions.append(line)

return captions

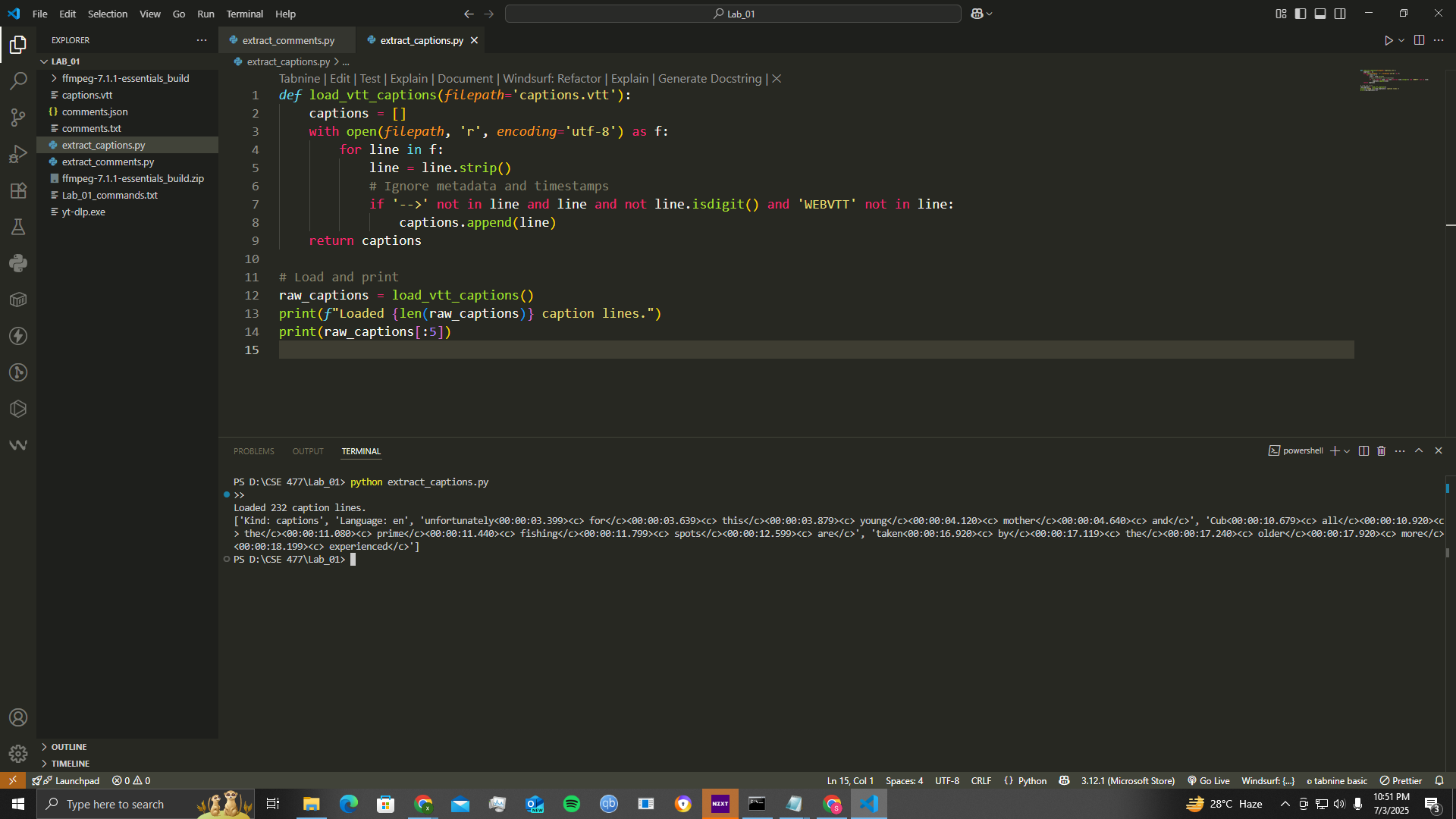
# Load and print

raw\_captions = load\_vtt\_captions()

print(f"Loaded {len(raw\_captions)} caption lines.")

print(raw\_captions[:5])

**Captions Output:**

****

**B. Required Lab Experiments**

**1. Histogram of Lengths (Captions vs. Comments):**

import json

import matplotlib.pyplot as plt

# -----------------------------

# Load captions from .vtt file

# -----------------------------

def load\_vtt\_captions(filepath='captions.vtt'):

captions = []

with open(filepath, 'r', encoding='utf-8') as f:

for line in f:

line = line.strip()

# Ignore metadata and timestamps

if '-->' not in line and line and not line.isdigit() and 'WEBVTT' not in line:

captions.append(line)

return captions

# -----------------------------

# Load comments from .json file

# -----------------------------

def load\_json\_comments(filepath='comments.json'):

comments = []

with open(filepath, 'r', encoding='utf-8') as f:

for line in f:

try:

obj = json.loads(line)

comments.append(obj['text'])

except json.JSONDecodeError:

continue

return comments

# Load data

raw\_captions = load\_vtt\_captions()

raw\_comments = load\_json\_comments()

# Print previews

print(f"Loaded {len(raw\_captions)} caption lines.")

print(raw\_captions[:5])

print(f"Loaded {len(raw\_comments)} comment lines.")

print(raw\_comments[:5])

# Plot Histogram of Lengths

caption\_lengths = [len(x) for x in raw\_captions]

comment\_lengths = [len(x) for x in raw\_comments]

plt.hist(caption\_lengths, bins=20, alpha=0.7, label='Captions')

plt.hist(comment\_lengths, bins=20, alpha=0.7, label='Comments')

plt.legend()

plt.xlabel('Length (characters)')

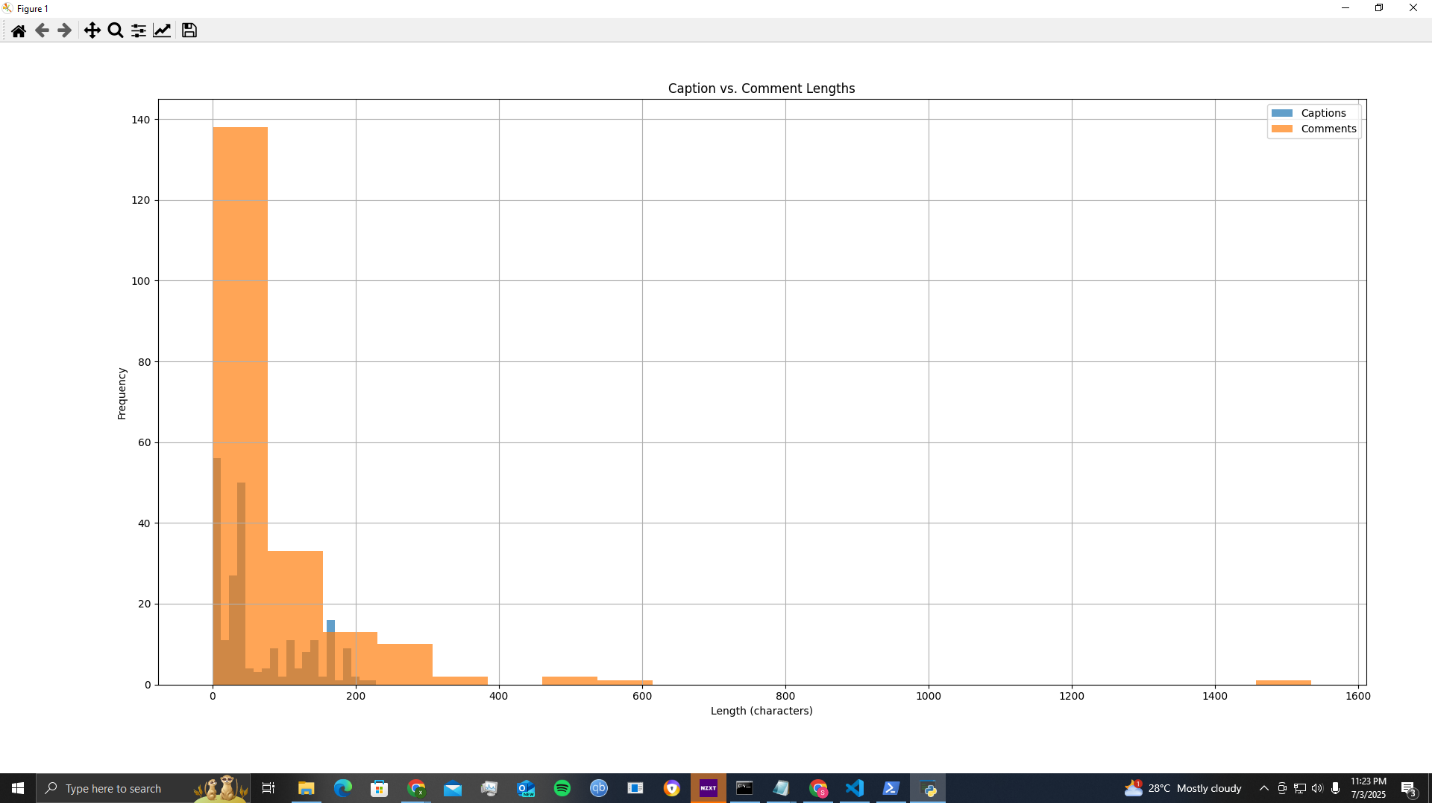
plt.ylabel('Frequency')

plt.title('Caption vs. Comment Lengths')

plt.grid(True)

plt.tight\_layout()

plt.show()

**Output :**

**2. Vocabulary Diversity (Type-Token Ratio):**

def type\_token\_ratio(lines):

words = [word.lower() for line in lines for word in line.split()]

unique = set(words)

return len(unique) / len(words) if words else 0

# Print TTR for captions and comments

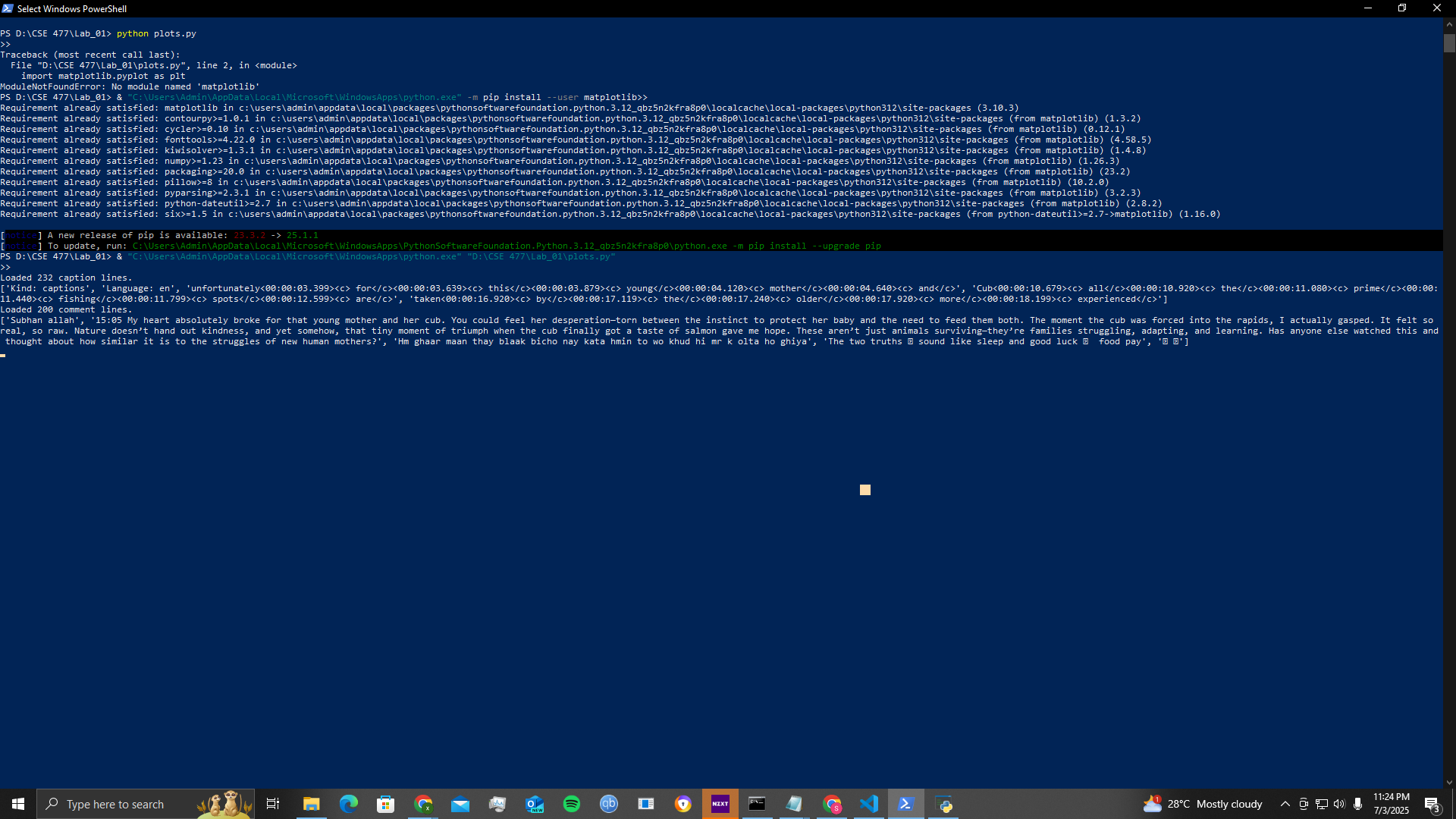
print("Caption TTR:", type\_token\_ratio(raw\_captions))

print("Comment TTR:", type\_token\_ratio(raw\_comments))

print("Total Captions:", len(raw\_captions))

print("Total Comments:", len(raw\_comments))

**Output :**



**3. Top-N Word Frequency (After Stopword Removal):**

import matplotlib.pyplot as plt

plt.bar(['Captions', 'Comments'], [len(raw\_captions), len(raw\_comments)], color=['skyblue', 'orange'])

plt.ylabel('Count')

plt.title('Number of Captions vs Comments')

plt.show()

# Print first few captions and comments

import pandas as pd

df\_caps = pd.DataFrame({'text': raw\_captions})

df\_caps['length'] = df\_caps['text'].apply(len)

print(df\_caps.describe())

plt.hist(df\_caps['length'], bins=20, alpha=0.7, label='Caption Lengths')

plt.xlabel('Caption Length (characters)')

plt.ylabel('Frequency')

plt.title('Caption Length Distribution')

plt.legend()

plt.grid(True)

plt.show()

# Assuming you’ve extracted timestamps already

df\_caps['timestamp'] = [...] # e.g., in seconds or minutes

plt.scatter(df\_caps['timestamp'], df\_caps['length'])

plt.xlabel('Time (sec)')

plt.ylabel('Caption Length')

plt.title('Caption Length vs Timestamp')

plt.show()

# Example: parsing comment JSON into a DataFrame

import json

comments = []

with open('comments.json', 'r', encoding='utf-8') as f:

for line in f:

obj = json.loads(line)

comments.append({

'text': obj['text'],

'likes': obj.get('likes', 0),

'length': len(obj['text'])

})

df\_comments = pd.DataFrame(comments)

plt.scatter(df\_comments['length'], df\_comments['likes'])

plt.xlabel('Comment Length')

plt.ylabel('Likes')

plt.title('Comment Length vs Likes')

plt.grid(True)

plt.show()

plt.scatter(df\_caps['timestamp'], df\_caps['length'])

plt.title('Caption Lengths Over Time')

plt.xlabel('Time (sec)')

plt.ylabel('Length')

plt.grid(True)

plt.show()

plt.hist(df\_comments['length'], bins=30, color='orange')

plt.title('Comment Length Distribution')

plt.xlabel('Comment Length')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Assuming `df\_comments['minute']` is the parsed minute mark

df\_by\_minute = df\_comments.groupby('minute').size()

df\_by\_minute.plot(kind='line')

plt.xlabel('Minute')

plt.ylabel('Number of Comments')

plt.title('Comments per Minute')

plt.grid(True)

plt.show()

**Output :**

