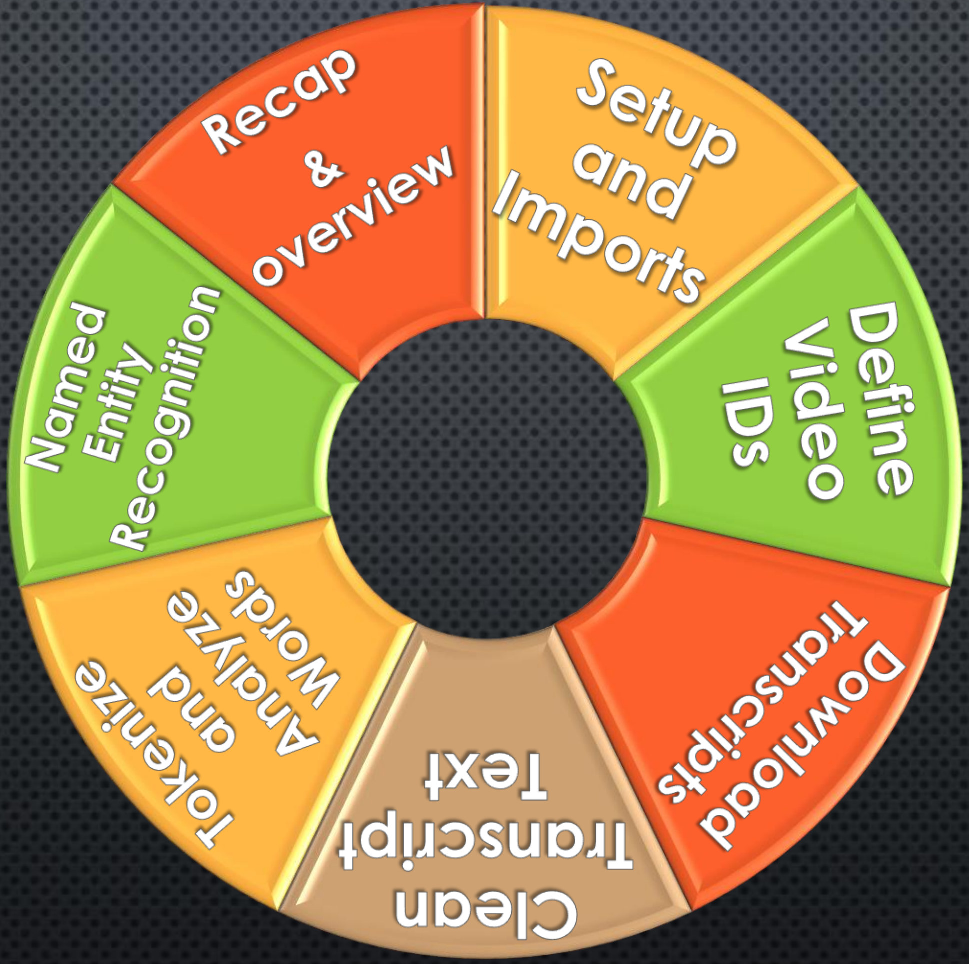
YOUTUBE TRANSCRIPT

NON-GCP VERSION

YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT

YOUTUBE TRANSCRIPT

```
# Import necessary libraries for the script
from youtube_transcript_api import YouTubeTranscriptApi  # For fetching YouTube transcripts
import re  # For regular expression operations
import nltk # For natural language processing
from nltk.tokenize import word_tokenize  # For tokenizing text
from collections import Counter  # For counting the frequency of words
import spacy  # For advanced natural language processing

# Define your API key (optional, not used here)
API_KEY = ‘’”  # If you have an API key, you can use it, but it's not necessary for this script
# Create YouTubeTranscriptApi object
transcript_api = YouTubeTranscriptApi()  # This will be used to fetch transcripts
```

YOUTUBE TRANSCRIPT

YOUTUBE_TRANSCRIPT_API

Applications

1. **Natural Language Processing (NLP):** Analyze the textual content of videos for various linguistic insights.
2. **Sentiment Analysis:** Determine the sentiment expressed in the video content.
3. **Content Summarization:** Generate concise summaries of lengthy video content.
4. **Keyword Extraction:** Identify key terms and phrases from video transcripts.
5. **Search Engine Optimization (SEO):** Improve video discoverability by extracting and using relevant keywords.
6. **Accessibility:** Provide text transcripts for users who are deaf or hard of hearing.
7. **Content Moderation:** Automatically review and flag inappropriate content.
8. **Educational Tools:** Create study guides or educational materials from video lectures.
9. **Market Research:** Analyze product reviews or public opinion expressed in videos.
10. **Customer Support:** Enhance customer support by analyzing and summarizing help videos.

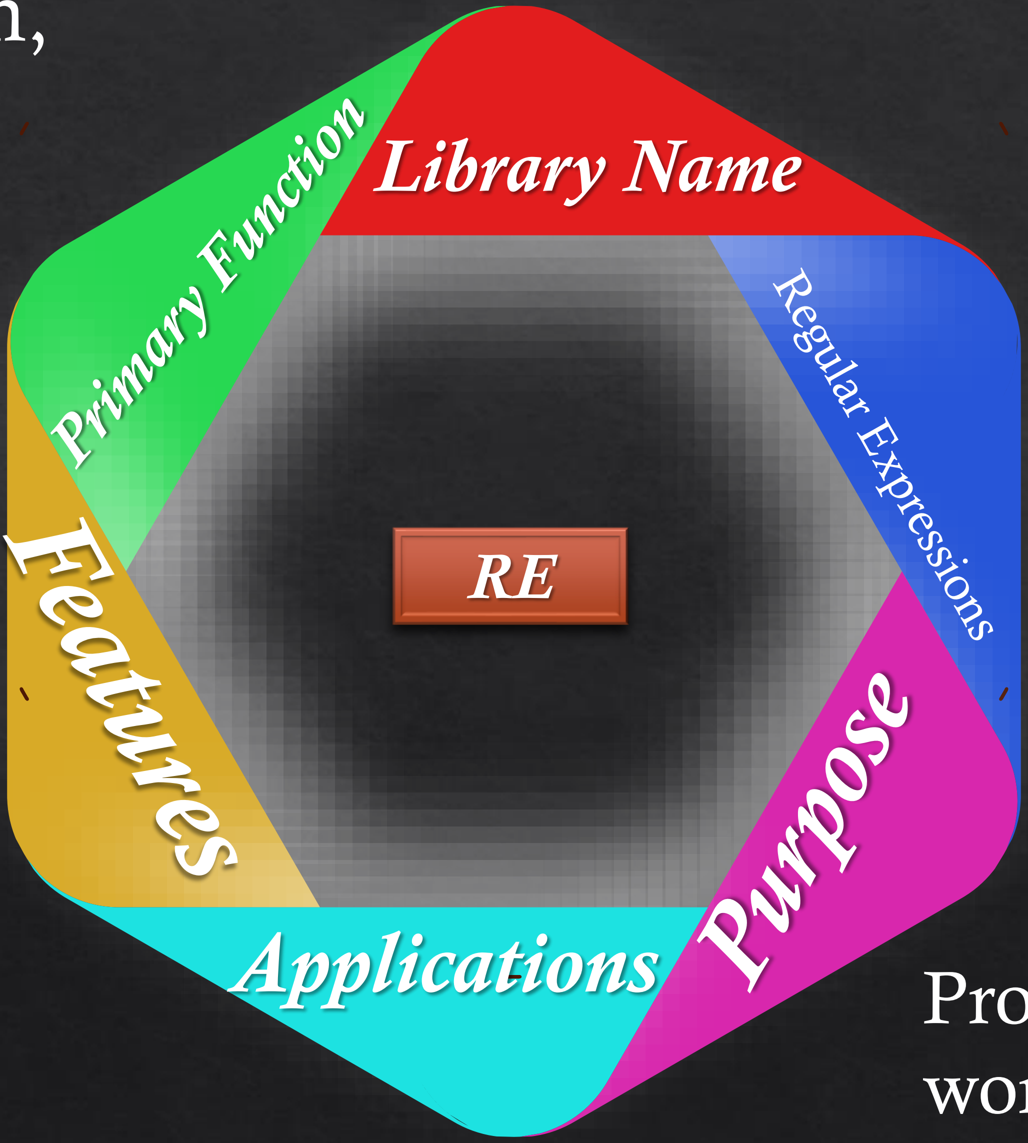
YOUTUBE TRANSCRIPT

RE

YOUTUBE TRANSCRIPT

Allows for the search, match, and manipulation of strings using regular expressions.

Includes functions for searching, matching, splitting, substituting, and more with regular expressions.



Provides support for working with regular expressions.

Applications

YOUTUBE TRANSCRIPT

RE

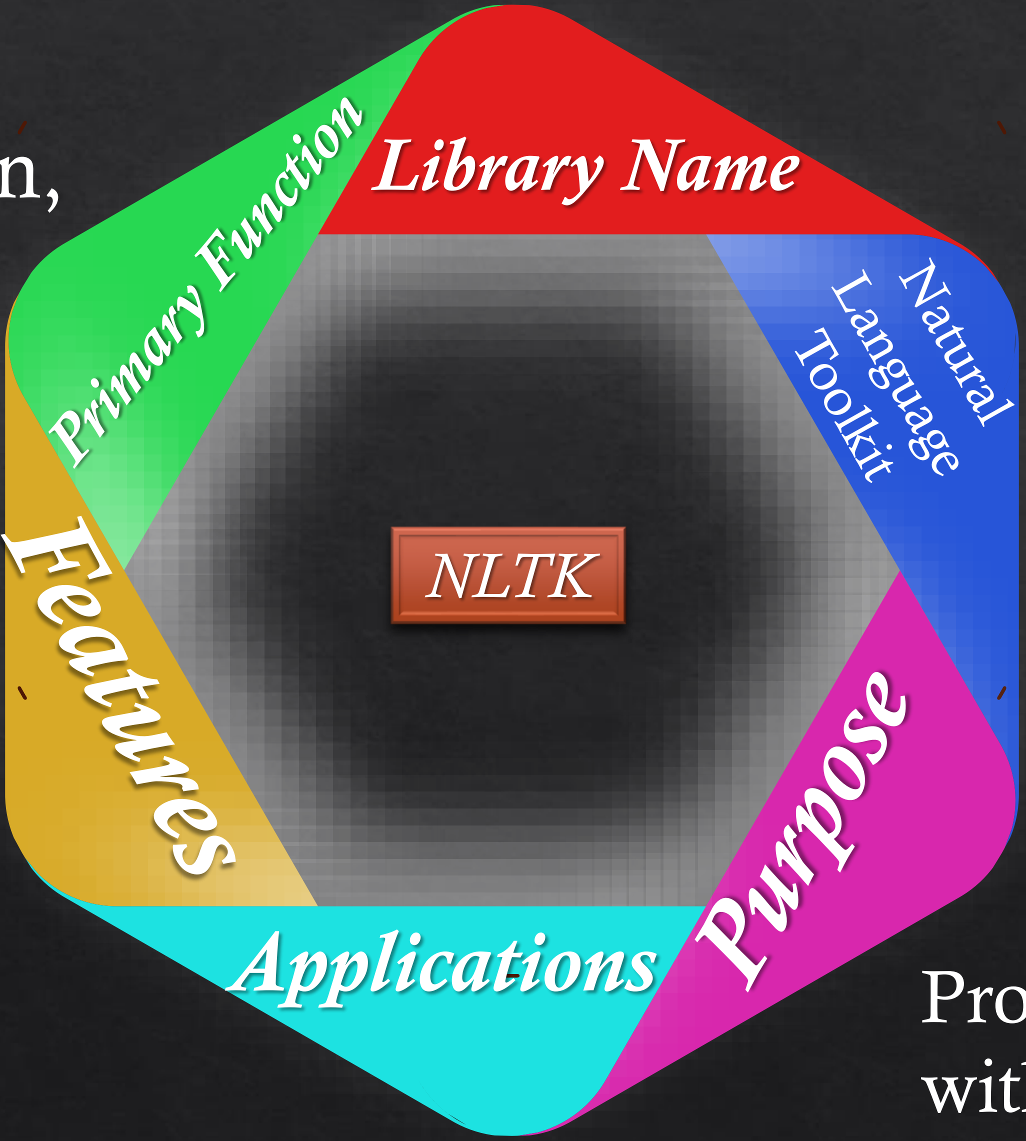
1. Pattern Matching: Checking if a string contains a specified pattern.
   Example: Verifying if an email address is valid.
2. Searching within Text: Finding the first or all occurrences of a pattern within a string.
   Example: Extracting all phone numbers from a text document.
3. String Substitution: Replacing parts of a string that match a pattern with another string.
   Example: Censoring specific words in user comments.
4. Text Splitting: Splitting a string into a list using a pattern as the delimiter.
   Example: Splitting a paragraph into sentences based on punctuation marks.
5. Data Validation: Validating the format of user input or data fields.
   Example: Ensuring a password meets complexity requirements.
6. Data Extraction: Extracting specific string parts that match a pattern.
   Example: Pulling out dates from a block of text.
7. Text Cleaning: Removing unwanted characters or substrings from the text.
   Example: Stripping out HTML tags from a web page's content.
8. Log File Analysis: Parsing and analyzing log files for specific patterns or errors.
   Example: Identifying failed login attempts in server logs.
9. Lexical Analysis: Tokenizing text in compilers and interpreters.
   Example: Breaking down source code into tokens.
10. Complex String Manipulations: Performing advanced string operations beyond simple search and replace.
    Example: Formatting text to meet specific presentation requirements.

YOUTUBE TRANSCRIPT

NLTK

Facilitates text processing tasks such as tokenization, parsing, classification, stemming, tagging, and semantic reasoning.

YOUTUBE TRANSCRIPT



Extensive libraries, corpora, and lexical resources for various natural language processing (NLP) tasks.

Provides tools for working with human language data (text).

Applications

YOUTUBE TRANSCRIPT

NLTK

- **Text Tokenization:** Splitting text into words or sentences.
- **Part-of-Speech Tagging:** Identifying grammatical parts of speech in text.
- **Named Entity Recognition (NER):** Extracting names of people, places, organizations, etc.
- **Text Classification:** Categorizing text into predefined classes.
- **Sentiment Analysis:** Analyzing the sentiment or emotional tone of text.
- **Machine Translation:** Translating text from one language to another.
- **Information Retrieval:** Extracting relevant information from large text datasets.
- **Language Modeling:** Predicting the likelihood of sequences of words.
- **Text Summarization:** Generating concise summaries from long documents.
- **Speech Recognition:** Converting spoken language into text.

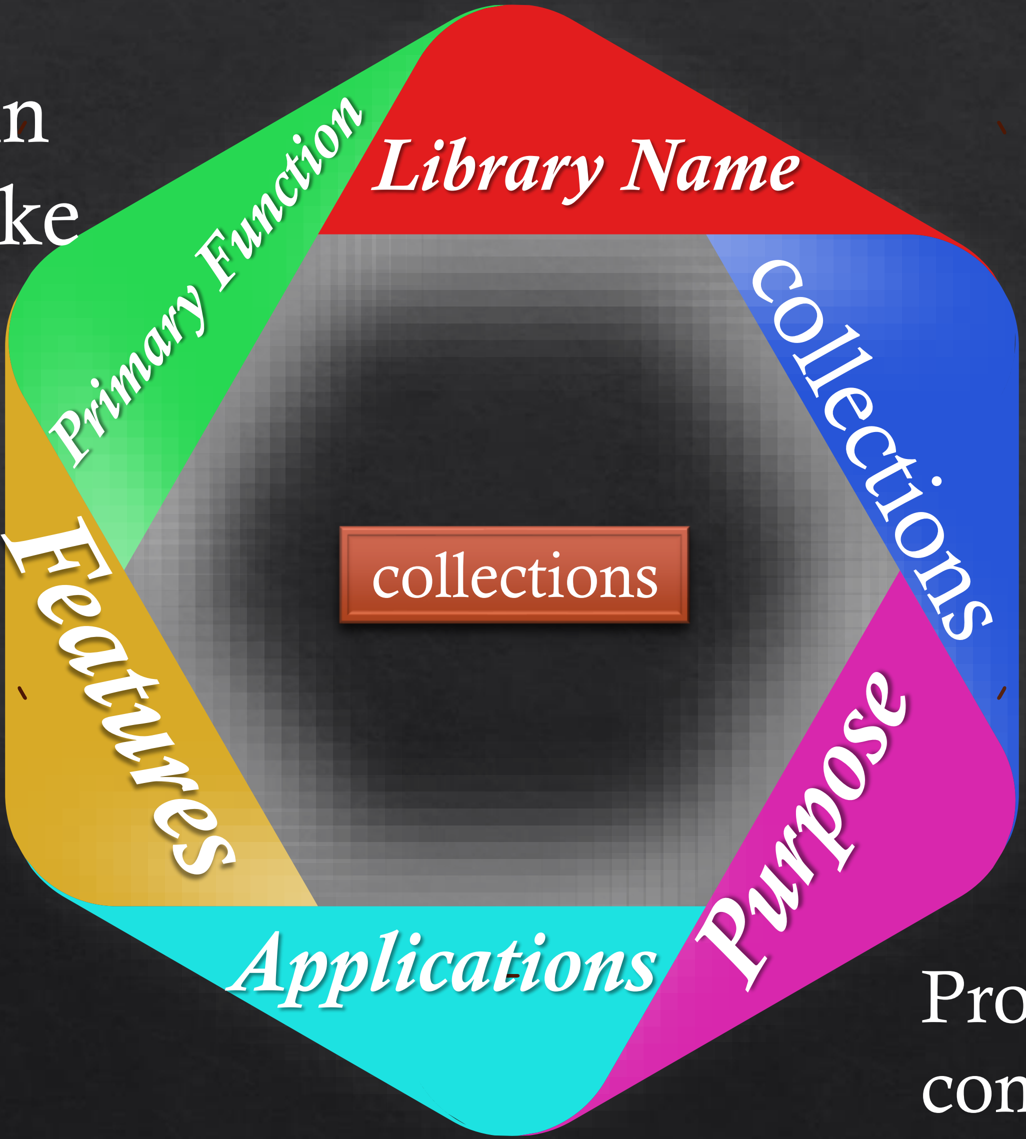
YOUTUBE TRANSCRIPT

collections

YOUTUBE TRANSCRIPT

Enhances the capabilities of built-in Python collections like lists, tuples, and dictionaries.

Includes namedtuples, deque, Counter, OrderedDict, defaultdict, ChainMap, and more.



Provides specialized container datatypes.

Applications

YOUTUBE TRANSCRIPT

collections

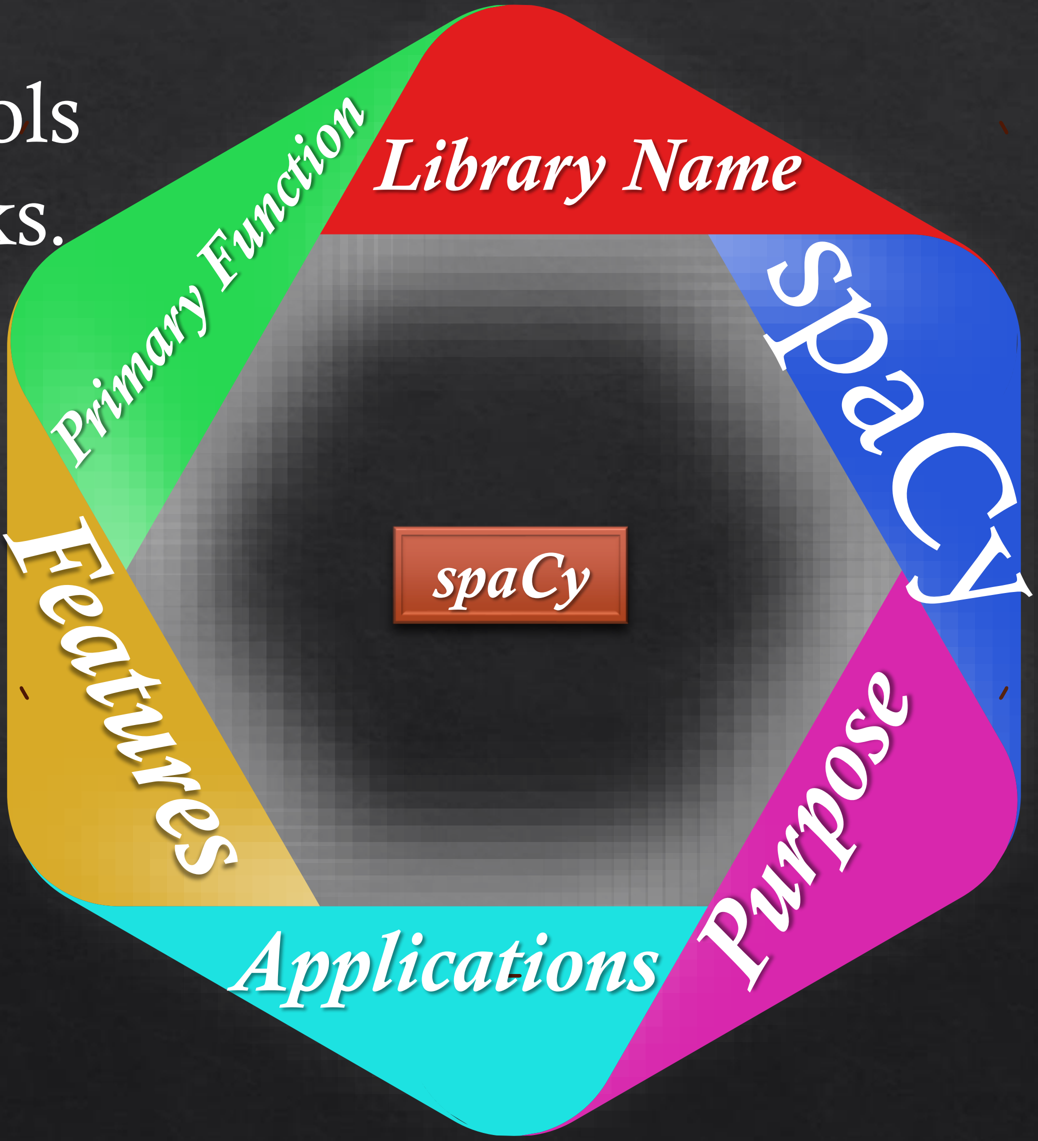
- **Counting Elements:** Using Counter to count occurrences of elements in a collection.
- **Efficient Stacks and Queues:** Using deque for fast appends and pops from both ends of the sequence.
- **Grouping Data:** Using defaultdict to group data into categories.
- **Maintaining Order:** Using OrderedDict to maintain the insertion order of items.
- **Data Structures:** Using namedtuple for creating immutable, self-documenting tuples.
- **Combining Mappings:** Using ChainMap to manage multiple dictionaries as a single unit.
- **Handling Missing Keys:** Using defaultdict to provide default values for missing dictionary keys.
- **Frequency Distribution:** Using Counter for frequency distribution analysis.
- **Reversible Data Structures:** Using deque to implement reversible data structures.
- **Improving Readability:** Using namedtuple to enhance code readability and maintainability.

YOUTUBE TRANSCRIPT

spaCy

YOUTUBE TRANSCRIPT

Provides efficient, production-ready tools for various NLP tasks.



Named entity recognition, part-of-speech tagging, dependency parsing, word vectors, and more.

Industrial-strength NLP library for processing and understanding large volumes of text.

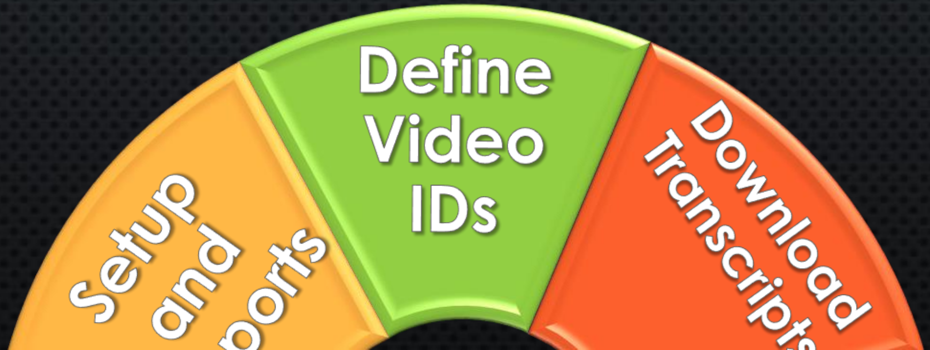
Applications

YOUTUBE TRANSCRIPT

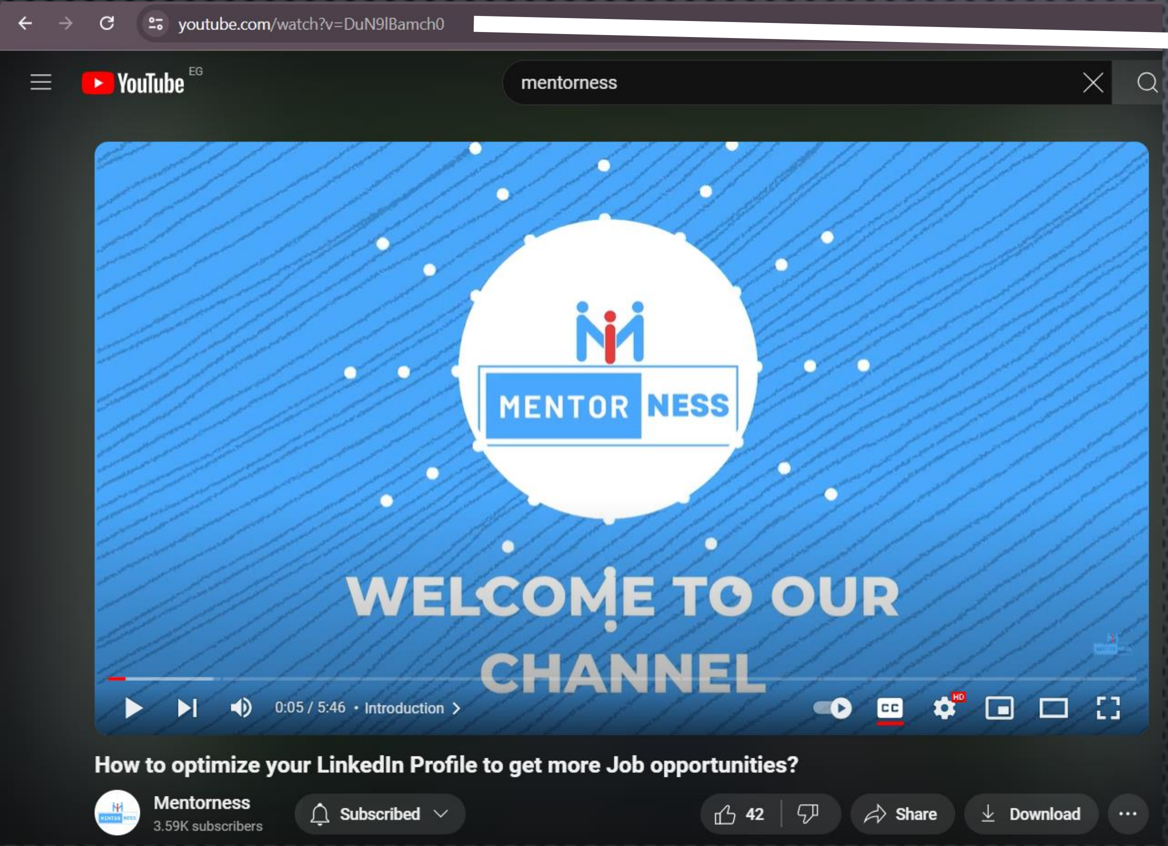
spaCy

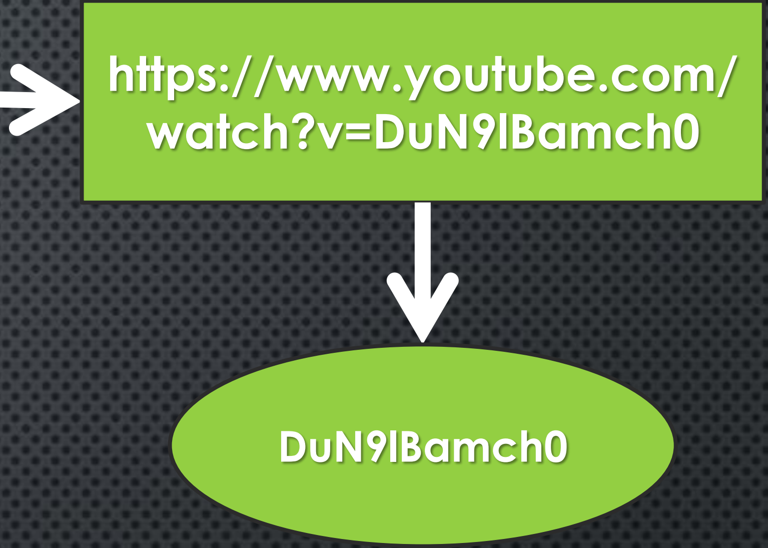
- **Named Entity Recognition (NER):** Identifying entities like names, dates, and locations in text.
- **Part-of-Speech Tagging:** Labeling words with their grammatical parts of speech.
- **Dependency Parsing:** Analyzing the grammatical structure of sentences.
- **Text Classification:** Categorizing text into predefined categories.
- **Word Vectors:** Using pre-trained word embeddings for semantic analysis.
- **Information Extraction:** Extracting structured information from unstructured text.
- **Question Answering:** Building systems that can answer questions based on text input.
- **Sentiment Analysis:** Determining the sentiment expressed in a piece of text.
- **Machine Translation:** Translating text between languages.
- **Chatbots:** Creating intelligent conversational agents.

YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT

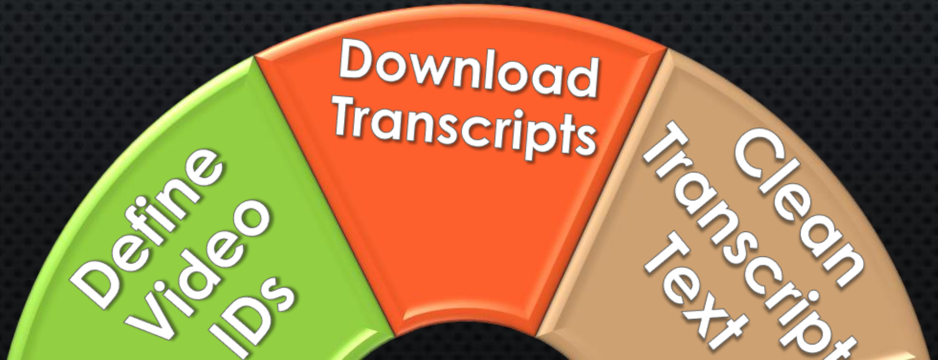

youtube.com/watch?v=DuN9lBamch0
YouTube
mentorness
MENTOR NESS
WELCOME TO OUR CHANNEL
0:05 / 5:46 · Introduction
How to optimize your LinkedIn Profile to get more Job opportunities?
Mentorness
3.59K subscribers
Subscribed
42
Share
Download



```
# List of YouTube video IDs you want to process
video_ids = ['_ygUxnPhiQo', 'qgdgBlOwQ58']
# Replace with your video IDs
```

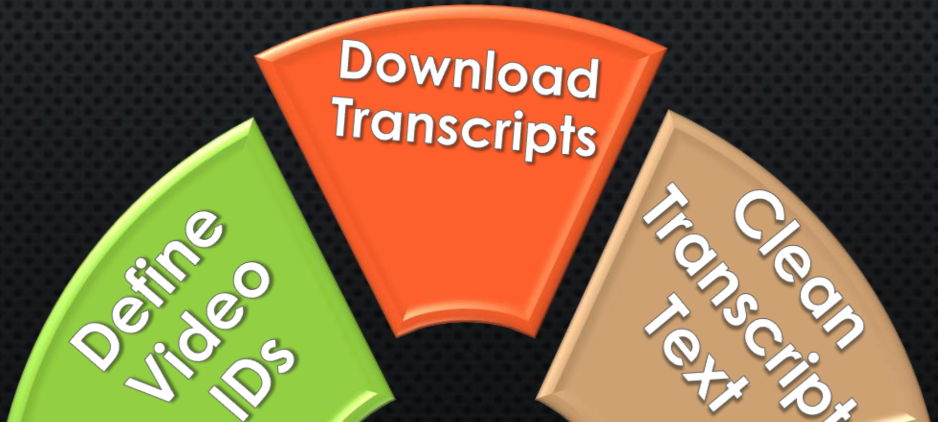
Setup and Imports
Define Video IDs
Download Transcripts

YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT

```
def download_transcript(video_id):
    """
    Downloads transcript for a given YouTube video ID.

    Parameters:
    video_id (str): The ID of the YouTube video.

    Returns:
    None
    """
```



YOUTUBE TRANSCRIPT

```
try:
        transcript = transcript_api.get_transcript(video_id, languages=['en'])  # Get transcript in English (or adjust 'en' for other languages)

        transcript_text = " ".join([item['text'] for item in transcript])  # Combine transcript text into a single string

        with open(f"transcript_{video_id}.txt", "w", encoding="utf-8") as f:  # Save transcript to a file (modify filename as needed)
      f.write(transcript_text)

    print(f"Transcript downloaded for video: {video_id}")
```

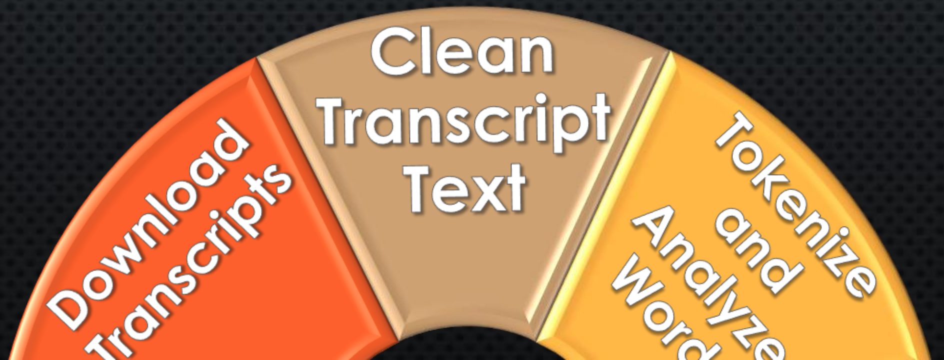
Define Video IDs
Download Transcripts
Clean Transcript Text

YOUTUBE TRANSCRIPT

```
    except Exception as e:
        print(f"Error downloading transcript for video {video_id}: {e}")

for video_id in video_ids:  # Download transcripts for each video ID
    download_transcript(video_id)

print("Download complete!")
```

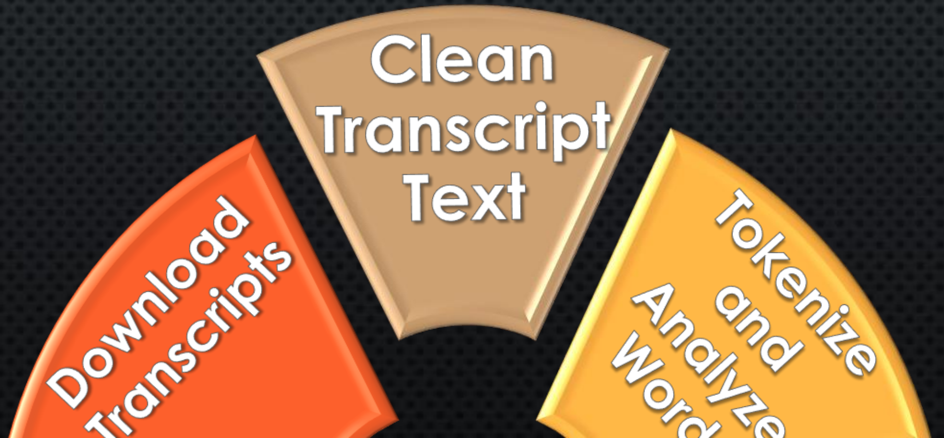
Define Video IDs
Download Transcripts
Clean Transcript Text

YOUTUBE TRANSCRIPT

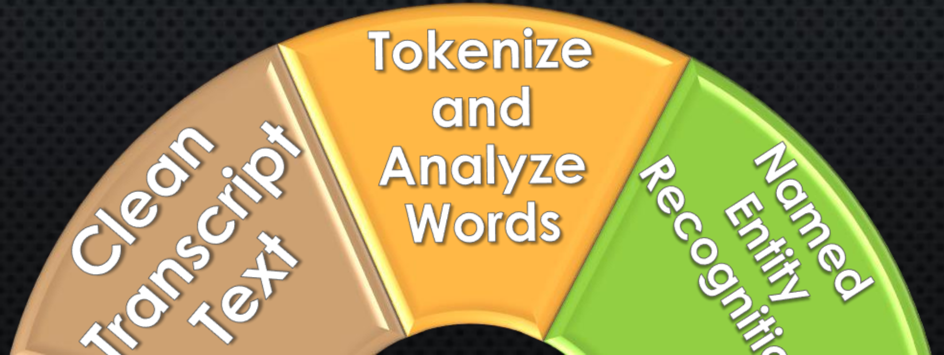


YOUTUBE TRANSCRIPT

```
def clean_transcript(text):
    """
    Cleans and preprocesses the transcript text.

    Parameters:
    text (str): The raw transcript text.

    Returns:
    str: The cleaned transcript text.
    """
# Remove special characters
cleaned_text = re.sub(r'[^\w\s]', '', text)

# Normalize whitespace
cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text

# Load the transcript for the first video and clean it
with open("transcript__ygUxnPhiQo.txt", "r",
encoding="utf-8") as f:
    transcript_text = f.read()

cleaned_text = clean_transcript(transcript_text)

print("Cleaned Transcript:")
print(cleaned_text)
```



YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT

```
# Download necessary NLTK resources (may take some time)
nltk.download('punkt')
nltk.download('stopwords’)

# Tokenize the cleaned text into words
tokens = word_tokenize(cleaned_text)

# Find the 10 most frequent words (excluding stop words)
stop_words = set(nltk.corpus.stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]
word_counts = Counter(filtered_tokens).most_common(10)

# Print the 10 most frequent words
print("10 Most Frequent Words:")
for word, count in word_counts:
    print(f"{word}: {count}")
```

Clean Transcript Text
Tokenize and Analyze Words
Named Entity Recognitio

YOUTUBE TRANSCRIPT

YOUTUBE TRANSCRIPT

```
# Load the English model (or a different model if needed)
nlp = spacy.load("en_core_web_sm")

# Process the text with spaCy
doc = nlp(cleaned_text)

# Extract named entities
print("Named Entities:")
for entity in doc.ents:
    print(f"Entity: {entity.text} ({entity.label_})")
```

YOUTUBE TRANSCRIPT

YOUTUBE TRANSCRIPT

```
# Import necessary libraries for the script
from youtube_transcript_api import YouTubeTranscriptApi  # For
fetching YouTube transcripts
import re  # For regular expression operations
import nltk  # For natural language processing
from nltk.tokenize import word_tokenize  # For tokenizing text
from collections import Counter  # For counting frequency of words
import spacy  # For advanced natural language processing

# Define your API key (optional, not used here)
API_KEY = ''  # If you have an API key, you can use it, but it's not
necessary for this script

# Create YouTubeTranscriptApi object
transcript_api = YouTubeTranscriptApi()  # This will be used to fetch
transcripts

# List of YouTube video IDs you want to process
video_ids = ['_ygUxnPhiQo', 'qgdgBlOwQ58']  # Replace with actual
video IDs

def download_transcript(video_id):
    """
    Downloads transcript for a given YouTube video ID.

    Parameters:
    video_id (str): The ID of the YouTube video.

    Returns:
    None
    """
    try:
        # Get transcript in English (or adjust 'en' for other languages)
        transcript = transcript_api.get_transcript(video_id, languages=['en'])

        # Combine transcript text into a single string
        transcript_text = " ".join([item['text'] for item in transcript])

        # Save transcript to a file (modify filename as needed)
        with open(f"transcript_{video_id}.txt", "w", encoding="utf-8") as f:
            f.write(transcript_text)

        print(f"Transcript downloaded for video: {video_id}")
    except Exception as e:
        print(f"Error downloading transcript for video {video_id}: {e}")

# Download transcripts for each video ID
for video_id in video_ids:
    download_transcript(video_id)

print("Download complete!")
```

YOUTUBE TRANSCRIPT

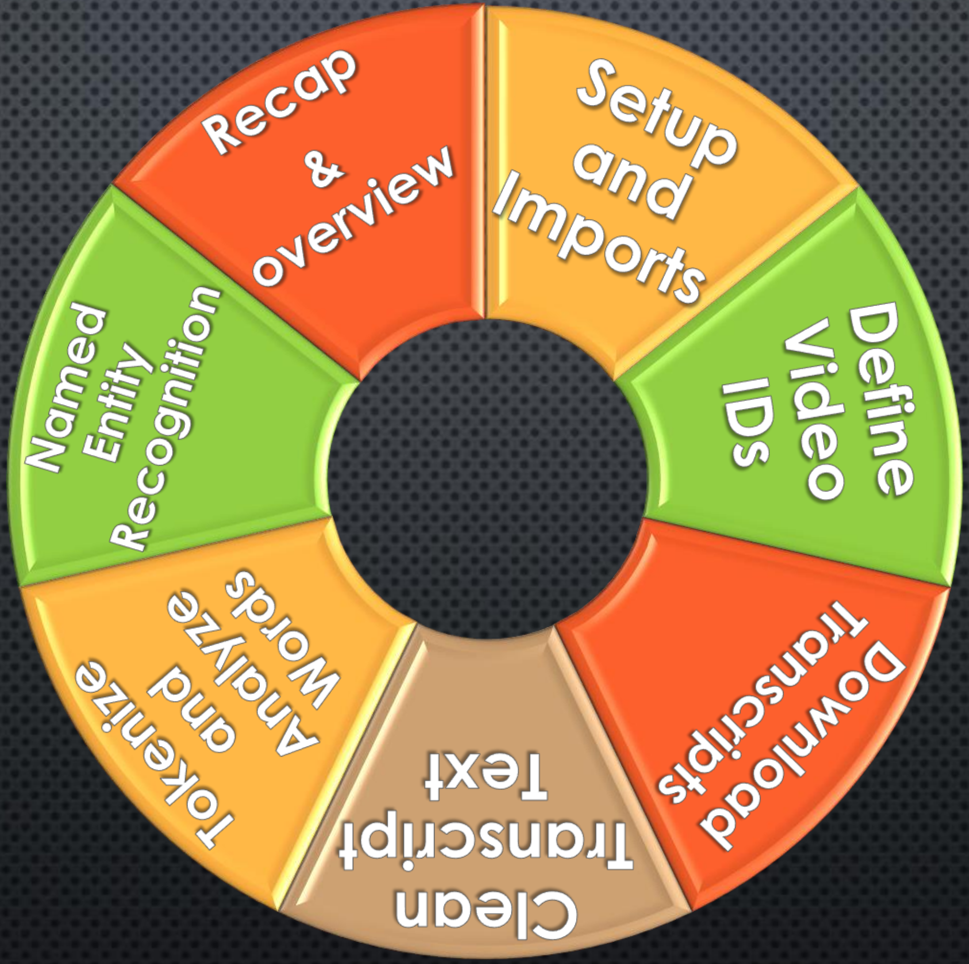
```
def clean_transcript(text):
    """
    Cleans and preprocesses the transcript text.

    Parameters:
    text (str): The raw transcript text.

    Returns:
    str: The cleaned transcript text.
    """
    # Remove special characters
    cleaned_text = re.sub(r'[^\w\s]', '', text)
    # Normalize whitespace
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text

# Load the transcript for the first video and clean it
with open("transcript__ygUxnPhiQo.txt", "r", encoding="utf-8") as f:
    transcript_text = f.read()

cleaned_text = clean_transcript(transcript_text)

print("Cleaned Transcript:")
print(cleaned_text)

# Download necessary NLTK resources (may take some time)
nltk.download('punkt')
nltk.download('stopwords')
```

```
# Tokenize the cleaned text into words
tokens = word_tokenize(cleaned_text)

# Find the 10 most frequent words (excluding stop words)
stop_words = set(nltk.corpus.stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]
word_counts = Counter(filtered_tokens).most_common(10)

# Print the 10 most frequent words
print("10 Most Frequent Words:")
for word, count in word_counts:
    print(f"{word}: {count}")

# Load the English model (or a different model if needed)
nlp = spacy.load("en_core_web_sm")

# Process the text with spaCy
doc = nlp(cleaned_text)

# Extract named entities
print("Named Entities:")
for entity in doc.ents:
    print(f"Entity: {entity.text} ({entity.label_})")
```

YOUTUBE TRANSCRIPT

GCP VERSION

YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT

Recap
&
verview
Setup
and
Imports
Define
Video
IDs

YOUTUBE TRANSCRIPT

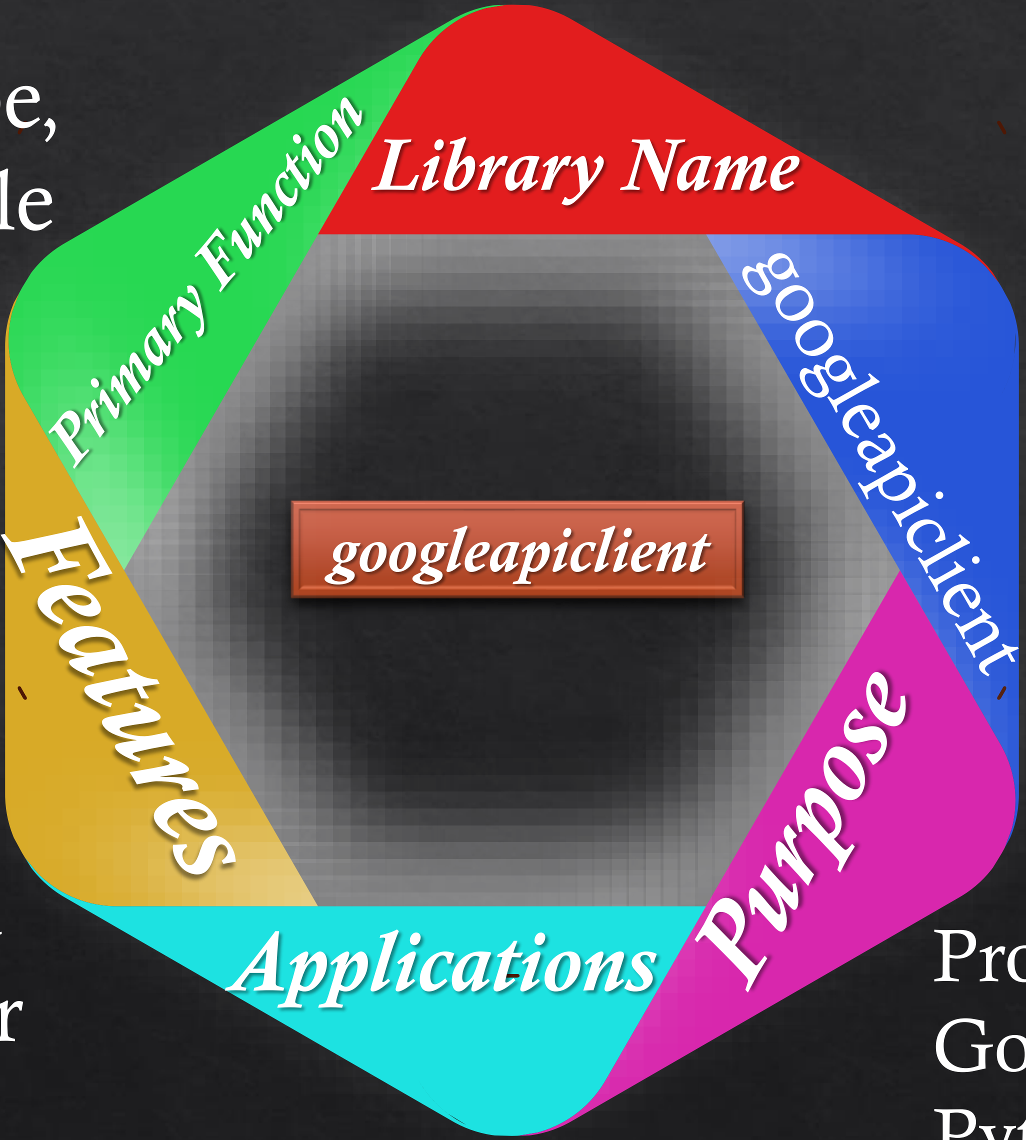
```
# Import necessary libraries for the script
from googleapiclient.discovery import build  # For accessing the YouTube API
import re  # For regular expression operations
import nltk  # For natural language processing
from nltk.tokenize import word_tokenize  # For tokenizing text
from collections import Counter  # For counting the frequency of words
import spacy  # For advanced natural language processing

# Define your API details
API_SERVICE_NAME = "youtube"
API_VERSION = "v3"
DEVELOPER_KEY = "YOUR_API_KEY"  # Replace with your actual API key
# Create a resource object for the YouTube API
youtube = build(API_SERVICE_NAME, API_VERSION, developerKey=DEVELOPER_KEY)
```

YOUTUBE TRANSCRIPT

googleapiclient

YOUTUBE TRANSCRIPT

Facilitates interaction with various Google services like YouTube, Google Drive, Google Sheets, etc.



Simplifies authentication, request building, and response handling for Google APIs.

Provides access to Google APIs using Python.

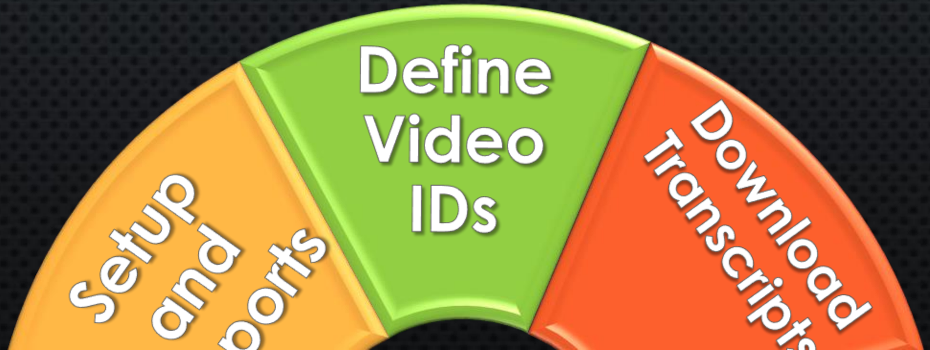
Applications

YOUTUBE TRANSCRIPT

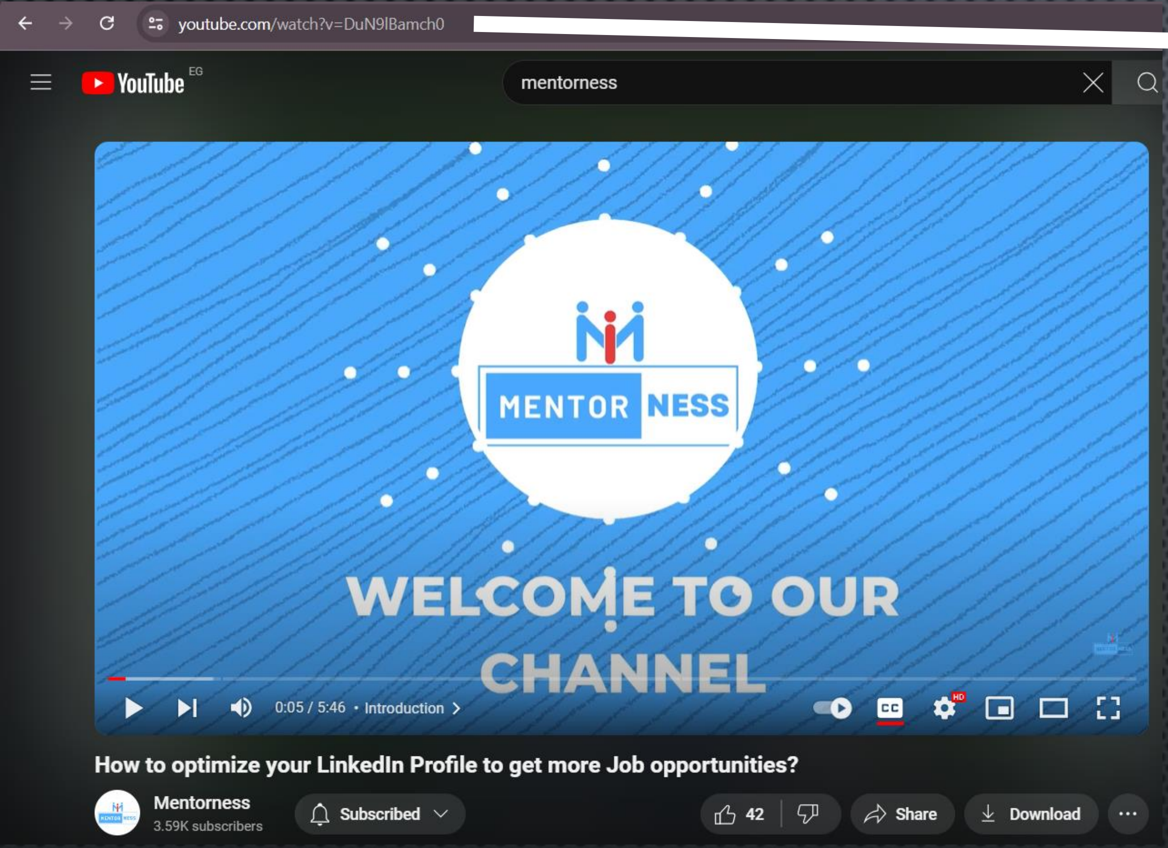
googleapiclient

- **YouTube Data API:** Accessing and manipulating YouTube video data and metadata.
- **Google Drive API:** Managing files and folders in Google Drive.
- **Google Sheets API:** Reading and writing data to Google Sheets.
- **Google Calendar API:** Creating and managing calendar events.
- **Google Maps API:** Accessing geographic data and maps services.
- **Gmail API:** Reading and sending emails programmatically.
- **Google Analytics API:** Accessing and analyzing web traffic data.
- **Google Vision API:** Analyzing images for various features like text, objects, and faces.
- **Google Cloud Storage API:** Managing cloud storage buckets and objects.
- **Google BigQuery API:** Performing large-scale data analysis and querying.

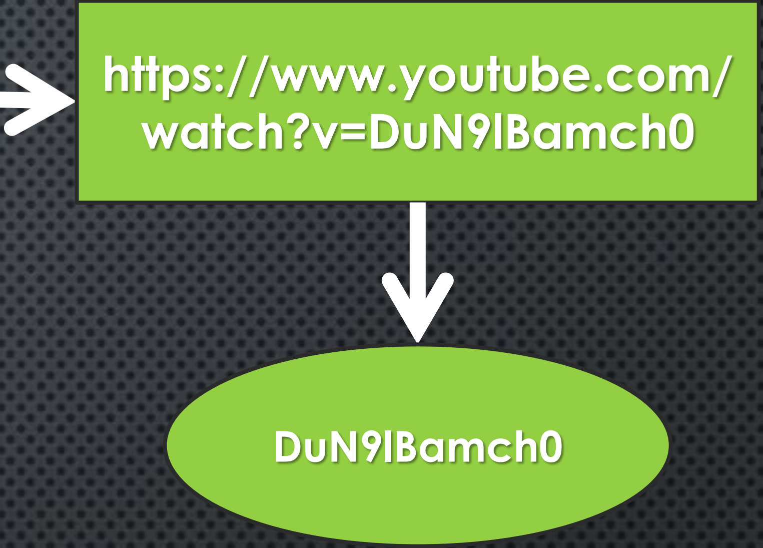
YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT



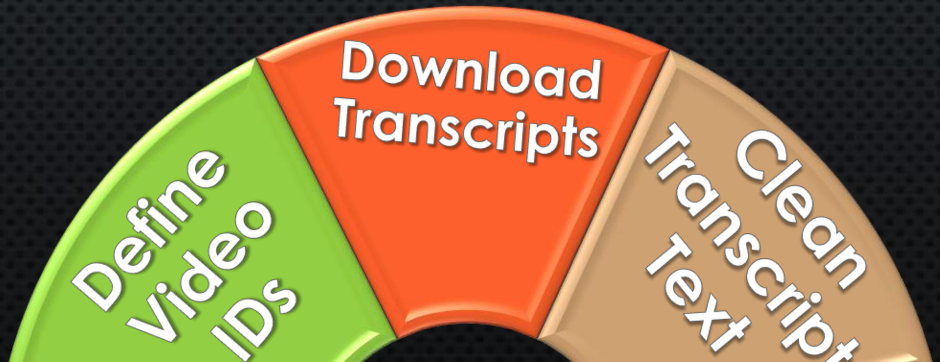
How to optimize your LinkedIn Profile to get more Job opportunities?



```
# List of YouTube video IDs you want to process
video_ids = ["VIDEO_ID_1", "VIDEO_ID_2", "VIDEO_ID_3"]
# Replace with your video IDs
```

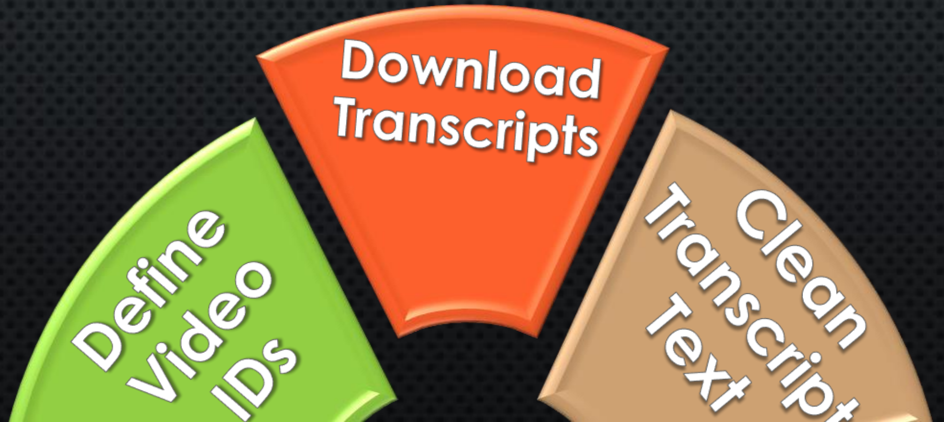
Setup and Imports
Define Video IDs
Download Transcripts

YOUTUBE TRANSCRIPT



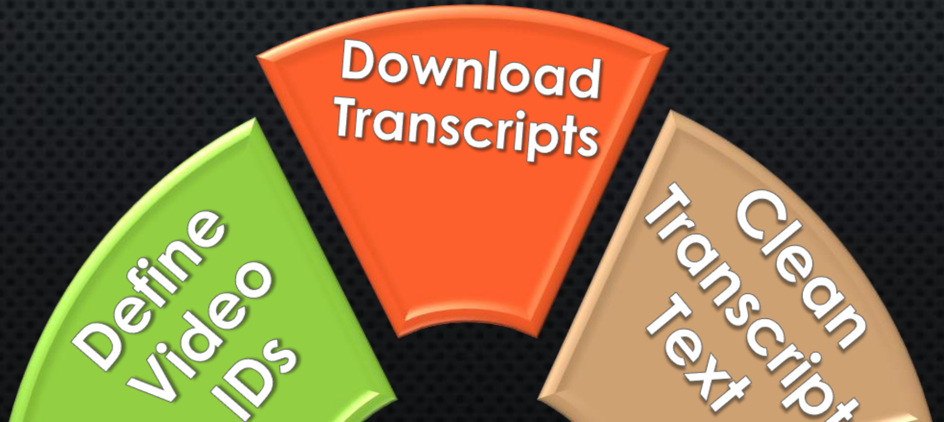
YOUTUBE TRANSCRIPT

```
def download_transcript(video_id):
    """
    Downloads transcript for a given YouTube video ID and saves it to a file.

    Parameters:
    video_id (str): The ID of the YouTube video.

    Returns:
    None
    """
```



YOUTUBE TRANSCRIPT

```
try:
        # Define request parameters
        request = youtube.videos().list(
        part="snippet",
        id=video_id
      )
        # Execute the request
        response = request.execute()

        # Extract transcript
        transcript = response["items"][0]["snippet"]["localized"]["en"]["transcript"]

        # Save transcript to a file (replace with desired filename format)
        filename = f"transcript_{video_id}.txt"
        with open(filename, "w", encoding="utf-8") as f:
              f.write(transcript)
```

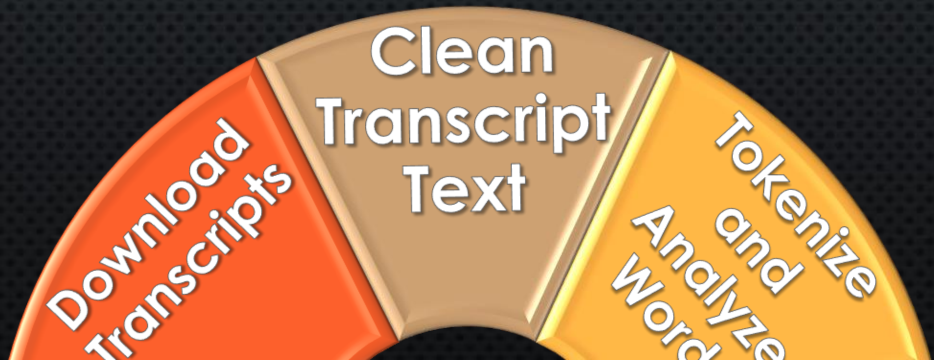


YOUTUBE TRANSCRIPT

```
print(f"Downloaded transcript for video {video_id} to {filename}")
    except (KeyError, ConnectionError) as e:
        print(f"Error downloading transcript for video {video_id}: {e}")

# Download transcripts for each video ID
for video_id in video_ids:
    download_transcript(video_id)

print("Download complete!")
```

Define Video IDs
Download Transcripts
Clean Transcript Text

YOUTUBE TRANSCRIPT

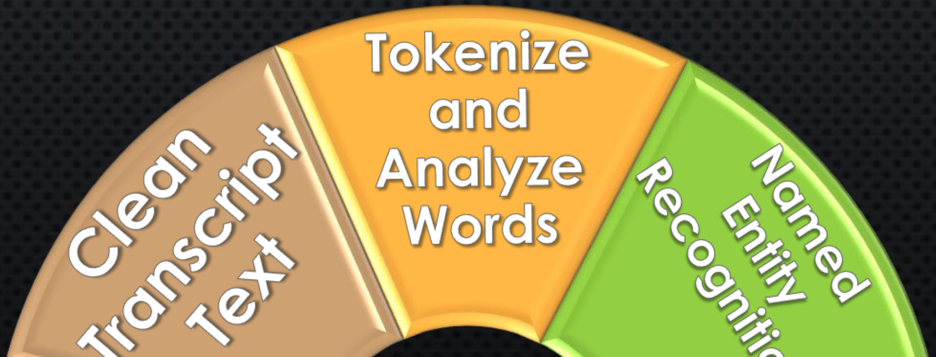


YOUTUBE TRANSCRIPT

```
def clean_transcript(text):
    """
    Cleans and preprocesses the transcript text.

    Parameters:
    text (str): The raw transcript text.

    Returns:
    str: The cleaned transcript text.
    """
# Remove special characters
cleaned_text = re.sub(r'[^\w\s]', '', text)

# Normalize whitespace
cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text

# Load the transcript for the first video and clean it
with open("transcript__ygUxnPhiQo.txt", "r",
encoding="utf-8") as f:
    transcript_text = f.read()

cleaned_text = clean_transcript(transcript_text)

print("Cleaned Transcript:")
print(cleaned_text)
```

Download Transcripts
Clean Transcript Text
Tokenize and Analyze Word

YOUTUBE TRANSCRIPT



YOUTUBE TRANSCRIPT

```
# Download necessary NLTK resources (may take some time)
nltk.download('punkt')
nltk.download('stopwords’)

# Tokenize the cleaned text into words
tokens = word_tokenize(cleaned_text)

# Find the 10 most frequent words (excluding stop words)
stop_words = set(nltk.corpus.stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]
word_counts = Counter(filtered_tokens).most_common(10)

# Print the 10 most frequent words
print("10 Most Frequent Words:")
for word, count in word_counts:
    print(f"{word}: {count}")
```

Clean Transcript Text
Tokenize and Analyze Words
Named Entity Recognitio

YOUTUBE TRANSCRIPT

YOUTUBE TRANSCRIPT

```
# Load the English model (or a different model if needed)
nlp = spacy.load("en_core_web_sm")

# Process the text with spaCy
doc = nlp(cleaned_text)

# Extract named entities
print("Named Entities:")
for entity in doc.ents:
    print(f"Entity: {entity.text} ({entity.label_})")
```

YOUTUBE TRANSCRIPT

YOUTUBE TRANSCRIPT

```
# Import necessary libraries for the script
from googleapiclient.discovery import build  # For accessing the
YouTube API
import re  # For regular expression operations
import nltk  # For natural language processing
from nltk.tokenize import word_tokenize  # For tokenizing text
from collections import Counter  # For counting frequency of words
import spacy  # For advanced natural language processing

# Define your API details
API_SERVICE_NAME = "youtube"
API_VERSION = "v3"
DEVELOPER_KEY = "YOUR_API_KEY"  # Replace with your actual API
key

# Create a resource object for the YouTube API
youtube = build(API_SERVICE_NAME, API_VERSION,
developerKey=DEVELOPER_KEY)

# List of YouTube video IDs you want to process
video_ids = ["VIDEO_ID_1", "VIDEO_ID_2", "VIDEO_ID_3"]  # Replace
with actual video IDs

def download_transcript(video_id):
    """
    Downloads transcript for a given YouTube video ID and saves it
to a file.

    Parameters:
    video_id (str): The ID of the YouTube video.

    Returns:
    None
    """
    try:
        # Define request parameters
        request = youtube.videos().list(
            part="snippet",
            id=video_id
        )

        # Execute the request
        response = request.execute()

        # Extract transcript (if available)
        transcript =
response["items"][0]["snippet"]["localized"]["en"]["tr
anscript"]

        # Save transcript to a file (replace with
desired filename format)
        filename = f"transcript_{video_id}.txt"
        with open(filename, "w", encoding="utf-8") as
f:
            f.write(transcript)

        print(f"Downloaded transcript for video
{video_id} to {filename}")
    except (KeyError, ConnectionError) as e:
        print(f"Error downloading transcript for video
{video_id}: {e}")

# Download transcripts for each video ID
for video_id in video_ids:
    download_transcript(video_id)

print("Download complete!")
```

YOUTUBE TRANSCRIPT

```python
def clean_transcript(text):
    """
    Cleans and preprocesses the transcript text.

    Parameters:
    text (str): The raw transcript text.

    Returns:
    str: The cleaned transcript text.
    """
    # Remove special characters
    cleaned_text = re.sub(r'[^\w\s]', '', text)
    # Normalize whitespace
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text

# Load the transcript for the first video and clean it
with open("transcript__ygUxnPhiQo.txt", "r", encoding="utf-8") as f:
    transcript_text = f.read()

cleaned_text = clean_transcript(transcript_text)

print("Cleaned Transcript:")
print(cleaned_text)

# Download necessary NLTK resources (may take some time)
nltk.download('punkt')
nltk.download('stopwords')
```

```python
# Tokenize the cleaned text into words
tokens = word_tokenize(cleaned_text)

# Find the 10 most frequent words (excluding stop words)
stop_words = set(nltk.corpus.stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]
word_counts = Counter(filtered_tokens).most_common(10)

# Print the 10 most frequent words
print("10 Most Frequent Words:")
for word, count in word_counts:
    print(f"{word}: {count}")

# Load the English model (or a different model if needed)
nlp = spacy.load("en_core_web_sm")

# Process the text with spaCy
doc = nlp(cleaned_text)

# Extract named entities
print("Named Entities:")
for entity in doc.ents:
    print(f"Entity: {entity.text} ({entity.label_})")
```

Thank
You