**DAY-1:**

#Write a program in Jupyter Notebook to declare variables of different data types (integer, float, string, and boolean). Print each variable and its type.

# Integer

my\_integer = 42

print("Value:", my\_integer, "| Type:", type(my\_integer))

# Float

my\_float = 3.14

print("Value:", my\_float, "| Type:", type(my\_float))

# String

my\_string = "Hello, Jupyter!"

print("Value:", my\_string, "| Type:", type(my\_string))

# Boolean

my\_boolean = True

print("Value:", my\_boolean, "| Type:", type(my\_boolean))

**DAY-2:**

#Create a List, tuple and Dictionary with 5 elements in it and how to access few elements based on the index. Try  with different examples

# List with 5 elements

my\_list = [10, 20, 30, 40, 50]

print("List:", my\_list)

print("Element at index 0:", my\_list[0])

print("Element at index 3:", my\_list[3])

print("Slice from index 1 to 3:", my\_list[1:4])  # Slicing the list

# Tuple with 5 elements

my\_tuple = ("apple", "banana", "cherry", "date", "elderberry")

print("\nTuple:", my\_tuple)

print("Element at index 2:", my\_tuple[2])

print("Last element:", my\_tuple[-1])  # Accessing with negative index

# Dictionary with 5 key-value pairs

my\_dict = {

    "name": "Alice",

    "age": 25,

    "city": "New York",

    "job": "Engineer",

    "hobby": "Reading"

}

print("\nDictionary:", my\_dict)

print("Value for 'name':", my\_dict["name"])

print("Value for 'hobby':", my\_dict["hobby"])

print("Value for 'job':", my\_dict.get("job"))

**DAY-3:**

#Write a Python program that takes a student's marks in three subjects as input.

  #If the average is greater than or equal to 90, print "Grade: A".

  #If the average is between 80 and 89, print "Grade: B".

  #If the average is between 70 and 79, print "Grade: C".

  #Otherwise, print "Grade: Fail".

# Taking marks for three subjects as input

subject1 = float(input("Enter marks for Subject 1: "))

subject2 = float(input("Enter marks for Subject 2: "))

subject3 = float(input("Enter marks for Subject 3: "))

# Calculating the average

average = (subject1 + subject2 + subject3) / 3

# Determining the grade based on the average

if average >= 90:

    print("Grade: A")

elif 80 <= average < 90:

    print("Grade: B")

elif 70 <= average < 80:

    print("Grade: C")

else:

    print("Grade: Fail")

**DAY-4:**

#Write a Python program to calculate the sum of all even numbers between 1 and a given positive integer n

# Taking input for the positive integer n

n = int(input("Enter a positive integer: "))

# Initialize the sum

even\_sum = 0

# Calculate the sum of all even numbers between 1 and n

for i in range(2, n + 1, 2):  # Start at 2 and increment by 2

    even\_sum += i

# Print the result

print(f"The sum of all even numbers between 1 and {n} is: {even\_sum}")

**DAY-5:**

#Write a Python program to calculate the frequency of each word in a given text. Print the words and their corresponding counts

# Taking input text from the user

text = input("Enter a text: ")

# Converting the text to lowercase and splitting it into words

words = text.lower().split()

# Initializing an empty dictionary to store word frequencies

word\_counts = {}

# Counting the frequency of each word

for word in words:

    word\_counts[word] = word\_counts.get(word, 0) + 1

# Printing the words and their counts

print("\nWord Frequencies:")

for word, count in word\_counts.items():

    print(f"{word}: {count}")

**DAY-6:**

#Write a Python program to using NLTK and Spacy

  #Convert text to lowercase.

  #Remove stopwords using NLTK

import nltk

from nltk.corpus import stopwords

import spacy

# Download NLTK stopwords

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

# Load spaCy model

nlp = spacy.load("en\_core\_web\_sm")

# Input text

text = "This is a simple example to demonstrate how to process text using NLTK and spaCy."

# Convert text to lowercase using spaCy

doc = nlp(text.lower())

# Remove stopwords using NLTK

filtered\_words = [word.text for word in doc if word.text not in stop\_words and word.is\_alpha]

# Print the processed text

print("Original Text:", text)

print("Processed Text:", " ".join(filtered\_words))

**DAY-7:**

pip install gensim nltk

import gensim

from gensim.utils import simple\_preprocess

from nltk.stem import PorterStemmer

from nltk.corpus import wordnet

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

import nltk

# Download NLTK resources

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('omw-1.4')

# Initialize tools

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

lemmatizer = WordNetLemmatizer()

# Load sample text file

file\_path = "sample\_text.rtf"  # Replace with your text file path

with open(file\_path, "r") as file:

    text = file.read()

# Preprocessing: Tokenization using Gensim

tokens = simple\_preprocess(text, deacc=True)  # deacc=True removes punctuation

# Remove stopwords

filtered\_tokens = [word for word in tokens if word not in stop\_words]

# Stemming

stemmed\_tokens = [stemmer.stem(word) for word in filtered\_tokens]

# Lemmatization

lemmatized\_tokens = [lemmatizer.lemmatize(word, wordnet.VERB) for word in stemmed\_tokens]

# Display results

print("Original Text:")

print(text)

print("\nTokenized Text:")

print(tokens)

print("\nAfter Removing Stopwords:")

print(filtered\_tokens)

print("\nAfter Stemming:")

print(stemmed\_tokens)

print("\nAfter Lemmatization:")

print(lemmatized\_tokens)

**DAY-8:**

#Tokenizes a sample paragraph into words and sentences.

!pip install nltk

import nltk

nltk.download('punkt')

nltk.download('wordnet')  # Optional for other tasks

import nltk

from nltk.tokenize import word\_tokenize, sent\_tokenize

# Re-download the 'punkt' package to ensure all files are intact

nltk.download('punkt')

nltk.download('punkt\_tab')

# Sample paragraph

paragraph = """

Natural Language Processing (NLP) is a fascinating field of artificial intelligence.

It helps machines understand, interpret, and respond to human language. Tokenization is one of the first steps in NLP!

"""

# Sentence Tokenization

try:

    sentences = sent\_tokenize(paragraph)

    print("Sentence Tokenization:")

    for i, sentence in enumerate(sentences, 1):

        print(f"{i}. {sentence}")

except LookupError as e:

    print("Error during sentence tokenization:", e)

# Word Tokenization

try:

    words = word\_tokenize(paragraph)

    print("\nWord Tokenization:")

    print(words)

except LookupError as e:

    print("Error during word tokenization:", e)

**DAY-9:**

#Write a Python function to clean a given text by removing special characters and converting it to lowercase. Test it with the input: 'Hello, World! Welcome to NLP 101.'

import re

def clean\_text(text):

    # Convert text to lowercase

    text = text.lower()

    # Remove special characters using regular expressions (retain only alphabets and spaces)

    text = re.sub(r'[^a-z\s]', '', text)

    return text

# Test the function with the input text

input\_text = "Hello, World! Welcome to NLP 101."

cleaned\_text = clean\_text(input\_text)

print("Original Text:", input\_text)

print("Cleaned Text:", cleaned\_text)

**DAY-10:**

#Write a Python function using regular expressions to extract all email addresses from a given string. Test it with the input: 'Contact us at support@example.com and sales@example.org.'

import re

def extract\_emails(text):

    # Regular expression pattern for matching email addresses

    email\_pattern = r'[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'

    # Find all matching email addresses using re.findall()

    emails = re.findall(email\_pattern, text)

    return emails

# Test the function with the provided input

test\_input = 'Contact us at support@example.com and sales@example.org.'

emails = extract\_emails(test\_input)

# Print the extracted email addresses

print("Extracted email addresses:", emails)

**DAY-11:**

#Write a Python script to fetch and print the title of a webpage using the 'requests' and 'BeautifulSoup' libraries. Test it with the URL: 'https://example.com'.

pip install requests beautifulsoup4

# Import necessary libraries

import requests

from bs4 import BeautifulSoup

# Define the URL

url = 'https://example.com'

# Fetch the content of the webpage

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

    # Parse the content using BeautifulSoup

    soup = BeautifulSoup(response.content, 'html.parser')

    # Find and print the title of the webpage

    title = soup.title.string

    print("Title of the webpage:", title)

else:

    print(f"Failed to retrieve the webpage. Status code: {response.status\_code}")

**DAY-12:**

#Write a Python script to generate a WordCloud from the text: 'data science machine learning artificial intelligence'. Save the WordCloud as an image.

pip install wordcloud matplotlib numpy

# Import necessary libraries

from wordcloud import WordCloud

import matplotlib.pyplot as plt

# Input text

text = 'data science machine learning artificial intelligence'

# Generate the WordCloud

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text)

# Display the WordCloud using matplotlib

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')  # Hide the axes

plt.show()

# Save the WordCloud as an image

wordcloud.to\_file('wordcloud\_image.png')