# PRACTICAL 1

## Aim:- Write a python program to implement Link Analysis. Description:-

Link analysis in web mining examines hyperlink structures to determine relationships between web pages. It is used for search engine ranking (e.g., PageRank), web structure mining, spam detection, and recommendation systems. By analyzing link patterns, it helps improve search accuracy, content organization, and user navigation.

## Code:-

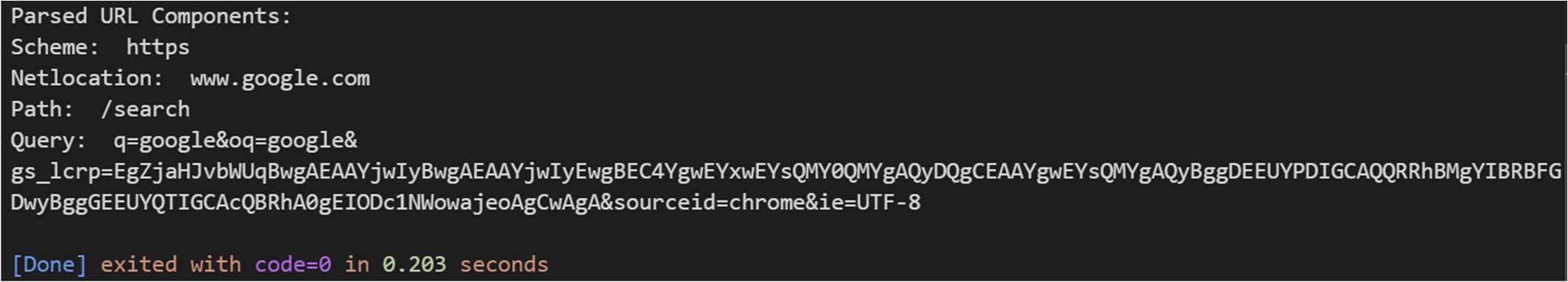
### !pip install requests beautifulsoup4 #Link analysis

s

"https://[www.google.com/search?q=google&oq=google&gs\_lcrp=EgZjaHJvbWUqBwgAEAAYjwIyBw](http://www.google.com/search?q=google&oq=google&gs_lcrp=EgZjaHJvbWUqBwgAEAAYjwIyBw) gAEAAYjwIyEwgBEC4YgwEYxwEYsQMY0QMYgAQyDQgCEAAYgwEYsQMYgAQyBggDEEUY PDIGCAQQRRhBMgYIBRBFGDwyBggGEEUYQTIGCAcQBRhA0gEIODc1NWowajeoAgCwAgA&s ourceid=chrome&ie=UTF-8"

parsed\_url = urlparse(url) print("Parsed URL Components:") print("Scheme: ",parsed\_url.scheme) print("Netlocation: ",parsed\_url.netloc) print("Path: ",parsed\_url.path) print("Query: ",parsed\_url.query)

**Output:-**

****

# PRACTICAL 2

## Aim:- Write a python program to make a web graph Description:-

A web graph in web mining represents web pages as nodes and hyperlinks as edges, forming a directed graph. It helps in search engine ranking (e.g., PageRank), web structure analysis, and efficient web crawling by identifying important and interconnected pages.

## Code:-

import networkx as nx

import matplotlib.pyplot as plt

### # create a directed graph

G = nx.DiGraph()

### # Add edges representing links between web pages

edges = [ ("Home","About"),

("Home", "Services"),

("About","Team"),

("Services","Contact"),

("Contact","Home"),

("Services","About")

]

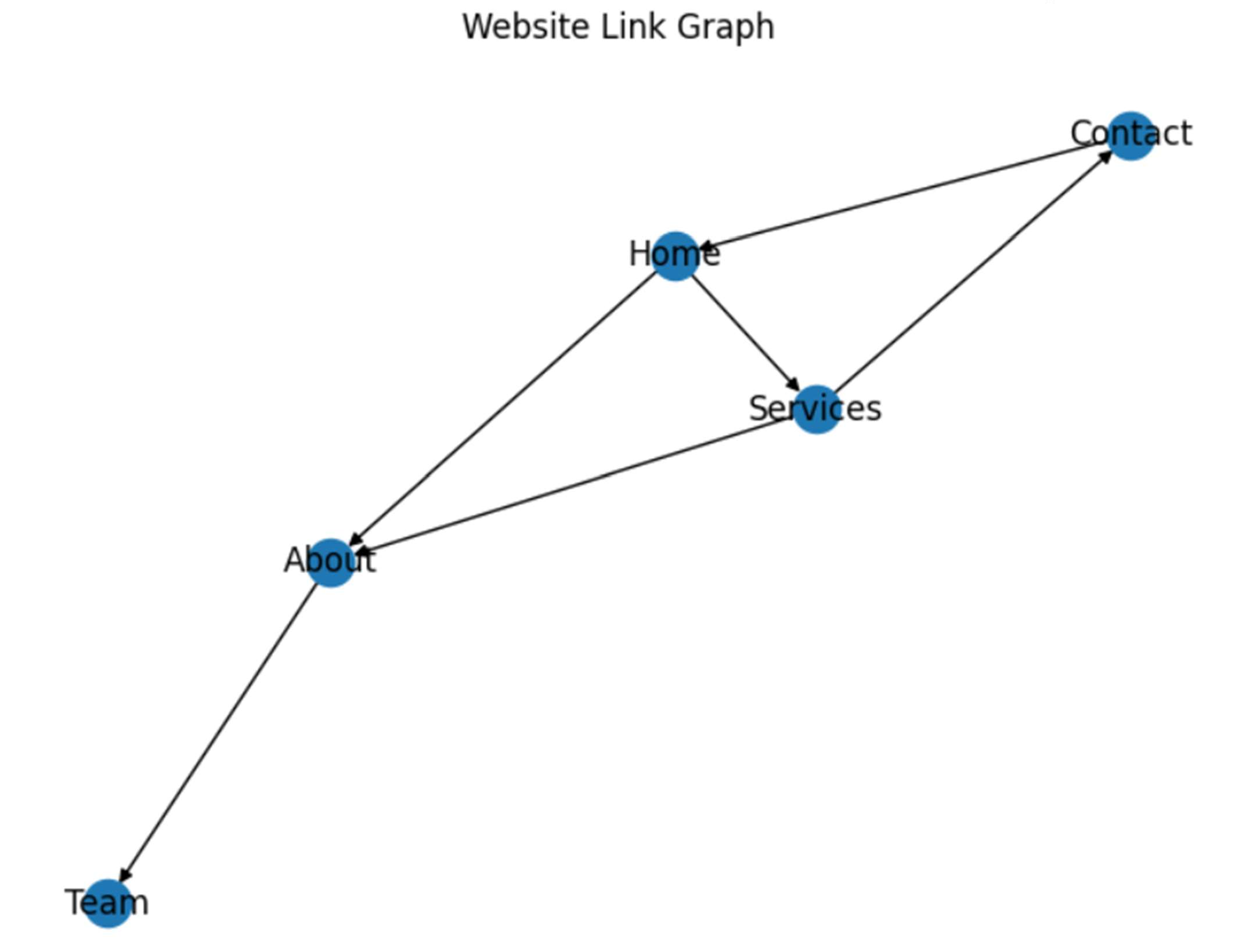
G.add\_edges\_from(edges)

### #Draw the graph

pos = nx.spring\_layout(G) **#positions from all nodes** nx.draw(G,pos,with\_labels = True, arrows=True) plt.title("Website Link Graph")

plt.show()

**Output:-**



# PRACTICAL 3

## Aim:- Write a python program to perform simple crawling from Wikipedia using keywords

**Description:-**

A web crawler in web mining is an automated program that browses the web systematically to collect and index data from web pages. It follows hyperlinks to discover new content and is used for search engines, data mining, and information retrieval.

## Code:-

### # Crawling through Wikipedia

import requests

from bs4 import BeautifulSoup

def simple\_crawler(keyword):

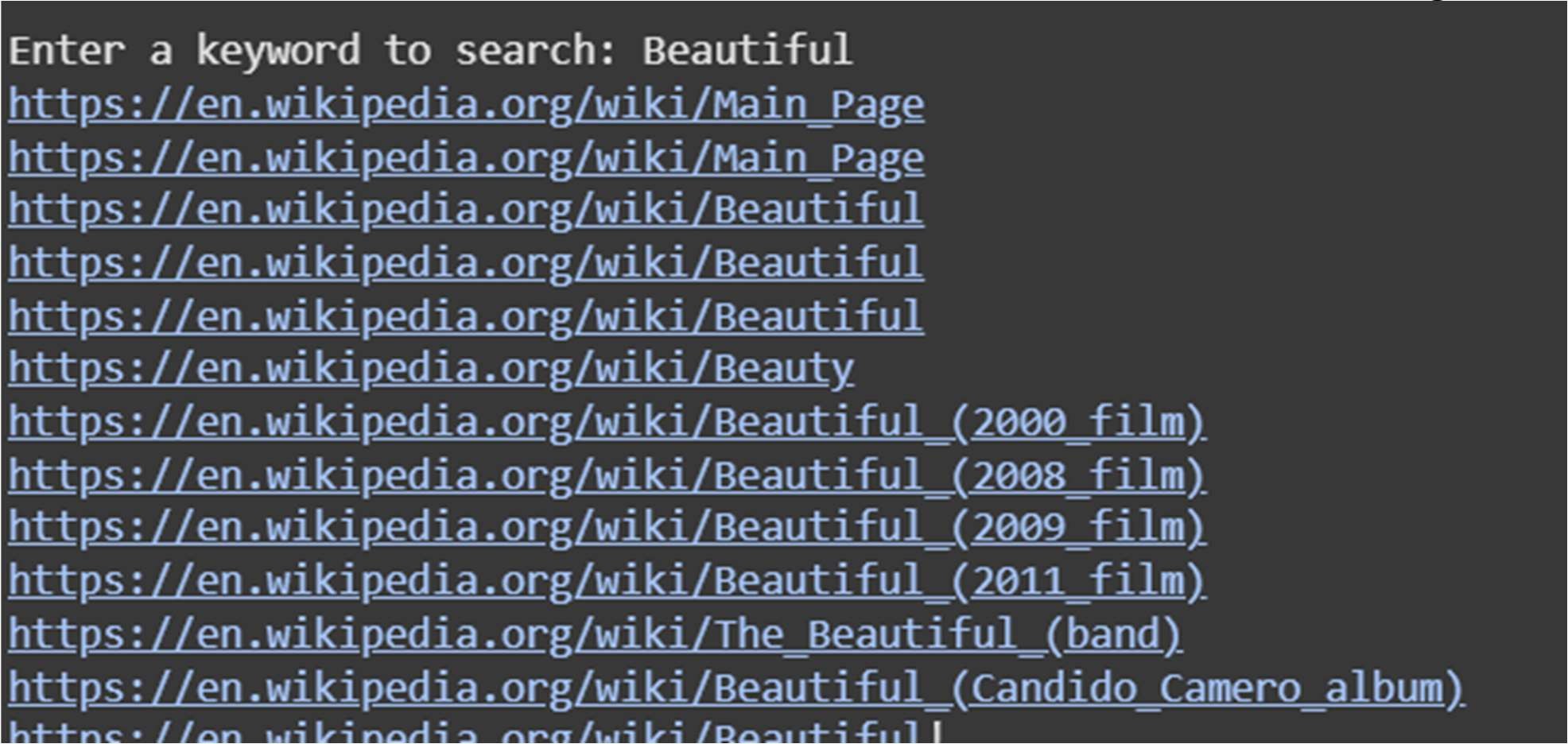
url = f"https://en.wikipedia.org/wiki/Special:Search?search={keyword}" response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser') for link in soup.find\_all('a',href=True):

if link['href'].startswith('/wiki/') and ':' not in link['href']: print(f"https://en.wikipedia.org{link['href']}")

keyword = input("Enter a keyword to search: ") simple\_crawler(keyword)

**Output:-**



# PRACTICAL 4

## Aim:- Write a python program to fetch page content Description:-

Fetching page content in web mining involves retrieving the HTML source of a webpage using web crawlers or scrapers. This process extracts useful data like text, links, and metadata for analysis, indexing, or information retrieval.

## Code:-

### # Fetch page

import requests

def fetch\_page(url):

headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'} try:

response = requests.get(url, headers=headers) response.raise\_for\_status() **# Raise an HTTPError for bad responses** return response.text

except requests.exceptions.HTTPError as e: print(f"HTTP error occurred: {e}")

except Exception as e:

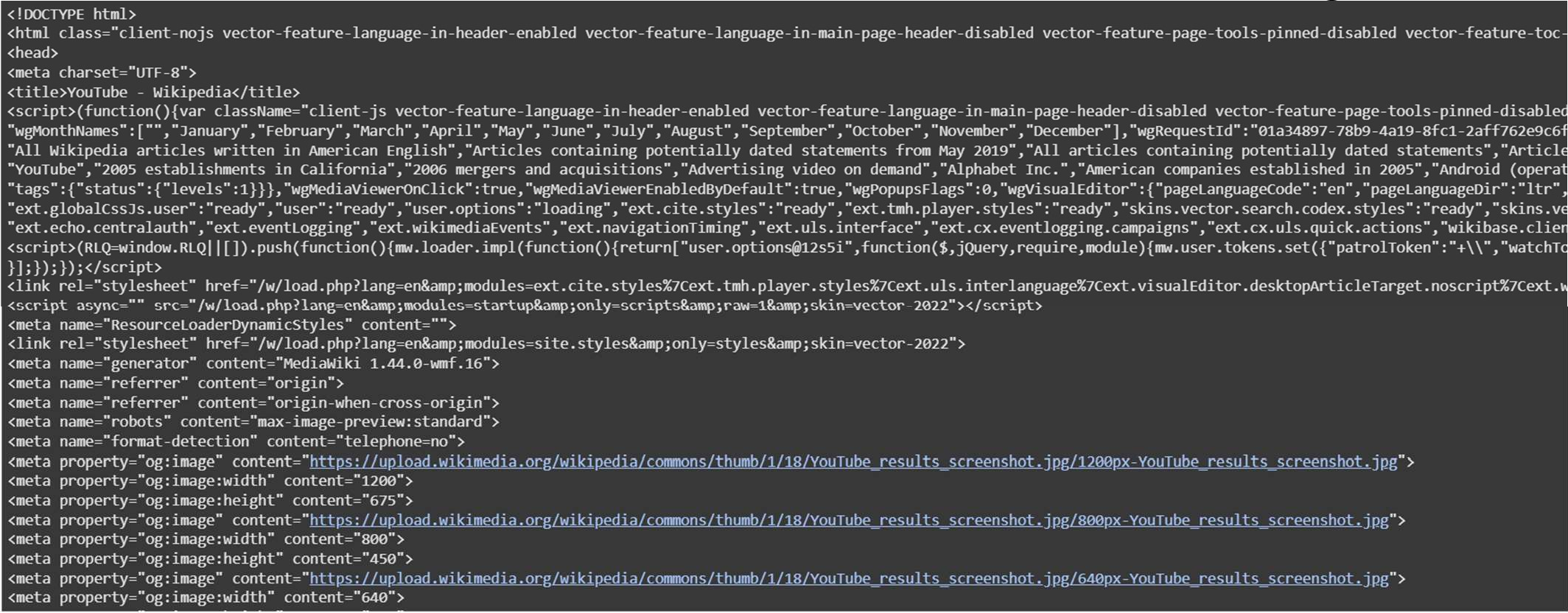
print(f"An error occurred: {e}")

url=" https://en.wikipedia.org/wiki/YouTube **# Replace with the target URL**

page\_content = fetch\_page(url) if page\_content:

print(page\_content)

**Output:-**



# PRACTICAL 5

## Aim:- Write a python program for Hits Algorithm Description:-

HITS (Hyperlink-Induced Topic Search) Algorithm in web mining ranks web pages by classifying them as hubs (linking to important pages) and authorities (highly linked pages). It improves search relevance by analyzing link structures.

## Code:-

### # HITS

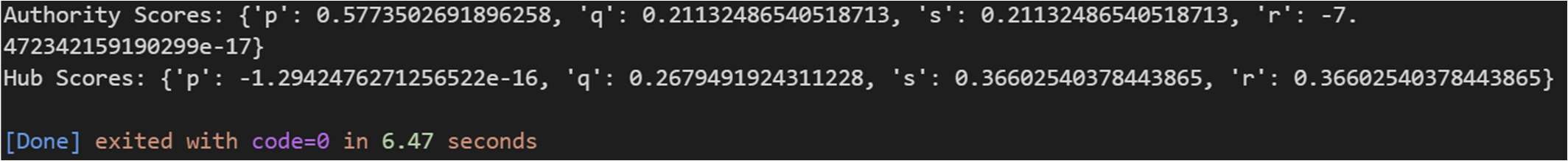
import networkx as nx G = nx.DiGraph()

### # G.add\_edges\_from([(1,2), (1,3), (2,4), (3,4), (4,5)])

G.add\_edges\_from([('p','q'), ('p','s'), ('p','r'), ('q','r'), ('r','p'), ('s','s')]) authority\_score, hub\_score = nx.hits(G)

print(f"Authority Scores: {authority\_score}") print(f"Hub Scores: {hub\_score}")

**Output:-**

****

# PRACTICAL 6

## Aim:- Write a python program for Page Rank Description:-

PageRank in web mining is an algorithm that ranks web pages based on the number and quality of incoming links. It helps search engines determine the importance and relevance of a webpage.

## Code:-

### # Page rank using NetworkX

import networkx as nx

import matplotlib.pyplot as plt G = nx.DiGraph() G.add\_edges\_from([

('A','B'),('A','C'),('B','A'),('B','D'),('B','E'),('B','F'), ('C','A'),('C','F')])

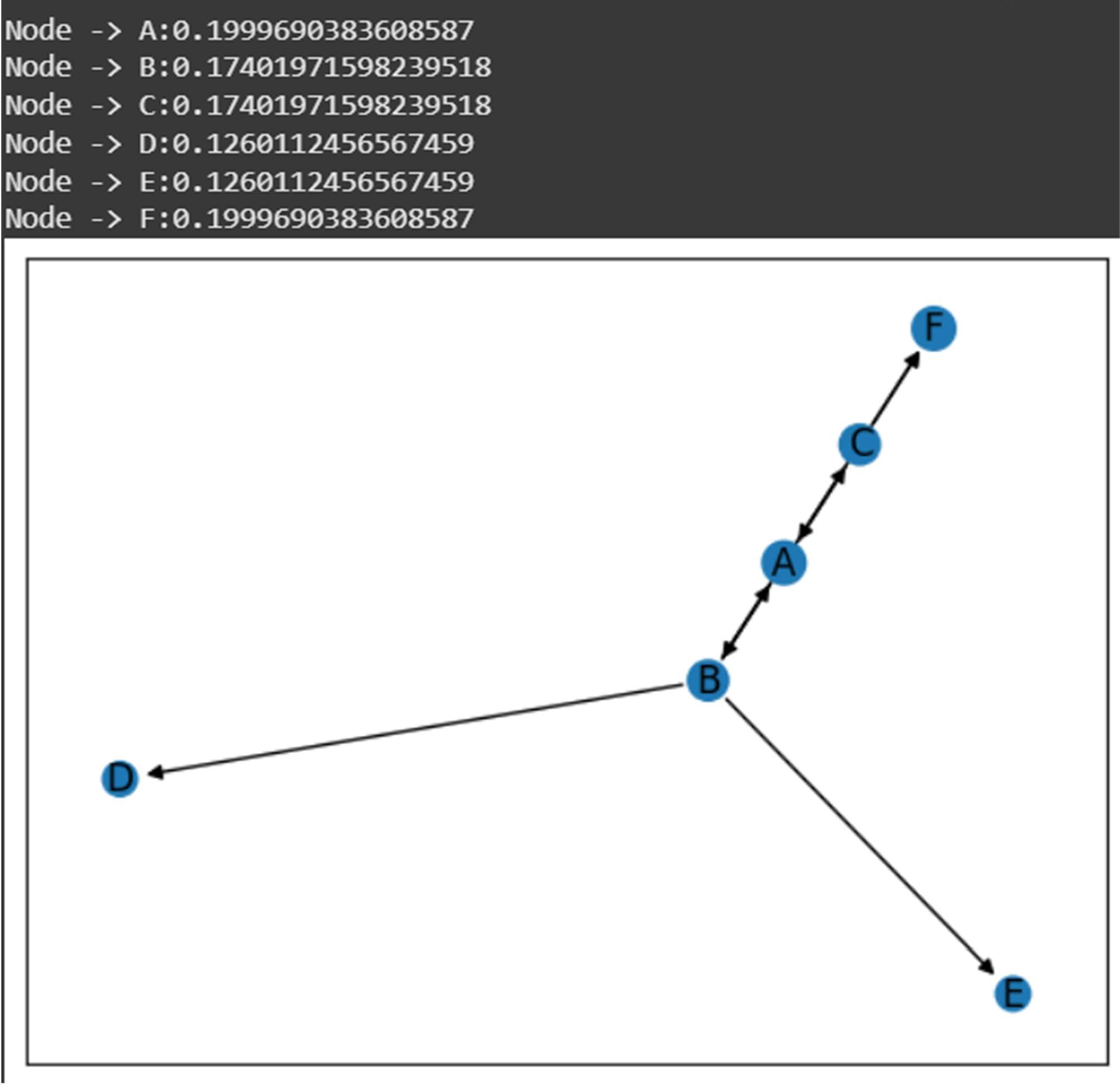
pr = nx.pagerank(G,alpha=0.85) for node, rank in pr.items():

print(f"Node -> {node}:{rank}") pos = nx.spring\_layout(G)

nx.draw\_networkx\_nodes(G, pos,node\_size=[pr[node]\*1000 for node in G.nodes()]) nx.draw\_networkx\_edges(G, pos)

nx.draw\_networkx\_labels(G, pos) plt.show()

**Output:-**



# PRACTICAL 7

## Aim:- Write a python program to summarize Text Description:-

Summarizing text in web mining involves extracting key information from large web documents using NLP techniques. It helps in quick understanding, content indexing, and efficient data retrieval.

## Code:-

### !pip install sumy

import nltk nltk.download('punkt\_tab')

from sumy.parsers.plaintext import PlaintextParser from sumy.nlp.tokenizers import Tokenizer

from sumy.summarizers.lex\_rank import LexRankSummarizer

def summarize\_text(text, num\_sentences=3):

### # Initialize the parser with the provided text

parser = PlaintextParser.from\_string(text, Tokenizer("english"))

### # Initialize the LexRankSummarizer

summarizer = LexRankSummarizer()

summary = summarizer(parser.document, num\_sentences) summarized\_text = ""

for sentence in summary: summarized\_text += str(sentence) + " "

return summarized\_text text = """

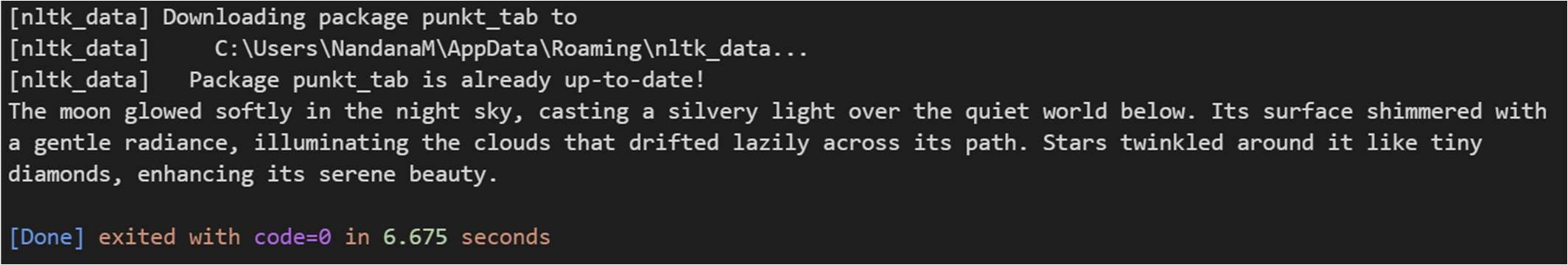
The narrator explains that Kamban's poem begins by describing the land of Kosala, where the story takes place.He describes the people at work on the land, and the animals that live among them. The capital city, Ayodhya, is a fabulous city ruled

over by King Dasaratha. Though Dasaratha is a compassionate and well-loved king, he laments that he's childless. One day, he mentions to his mentor that he has no sons

to succeed him,

and asks his mentor for help.""" summary = summarize\_text(text) print(summary)

**Output:-**

****

# PRACTICAL 8

## Aim:- Write a python program to implement a Recommender System. Description:-

A Recommender System in web mining analyzes user behavior and preferences to suggest relevant content, products, or services. It enhances user experience in e-commerce, streaming platforms, and online services. **Code:-**

### !pip install pandas scikit-surprise

import pandas as pd

from surprise import Dataset,Reader from surprise import SVD

from surprise import accuracy

from surprise.model\_selection import train\_test\_split

data = {

'user\_id': [1,1,1,2,2,3,3,3,4,4],

'item\_id': ['A','B','C','A','C','A','B','D','B','C'], 'rating' : [5,3,4,4,8,6,1,2,4,1]

}

df = pd.DataFrame(data)

reader = Reader(rating\_scale=(1,5))

dataset = Dataset.load\_from\_df(df[['user\_id','item\_id','rating']],reader) trainset, testset = train\_test\_split(dataset, test\_size=0.2) **# test\_size=0.2 means** testing only 20% data

model = SVD() model.fit(trainset)

predictions = model.test(testset) rmse = accuracy.rmse(predictions) print(f"RMSE: {rmse}")

pred = model.predict(3,'B')

print(f"Prediction for user 1, item A: {pred.est}")

**Output:-**

