**CO3**

**K-MEANS(without csv)**

import numpy as np

import matplotlib.pyplot as plt

# Generate sample data

data = np.array([[2, 3], [3, 4], [5, 6], [9, 10], [10, 8]])

# Number of clusters (you can change this)

k = 3

# Initialize centroids randomly

centroids = data[np.random.choice(data.shape[0], k, replace=False)]

# Main K-means algorithm

num\_iterations = 100

for iteration in range(num\_iterations):

# Calculate distances between data points and centroids

distances = np.linalg.norm(data[:, np.newaxis] - centroids, axis=2)

# Assign data points to the nearest centroid

labels = np.argmin(distances, axis=1)

print(labels)

# Plot data points and current centroids

plt.scatter(data[:, 0], data[:, 1], c=labels)

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200)

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title(f"Iteration {iteration}")

plt.show()

# Update centroids based on the mean of assigned data points

new\_centroids = np.array([data[labels == i].mean(axis=0) for i in range(k)])

# Check for convergence

if np.array\_equal(centroids,new\_centroids):

break

centroids = new\_centroids

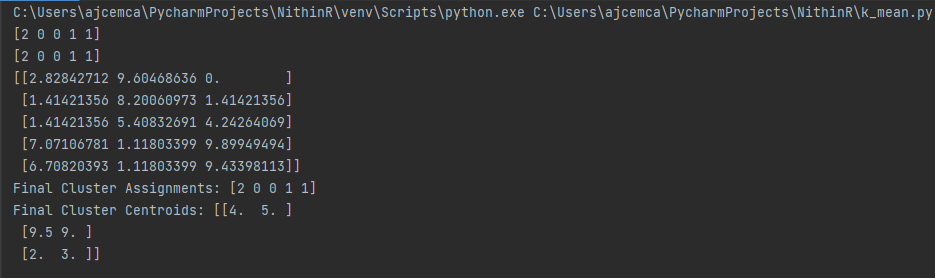
# Print final cluster assignments and centroids

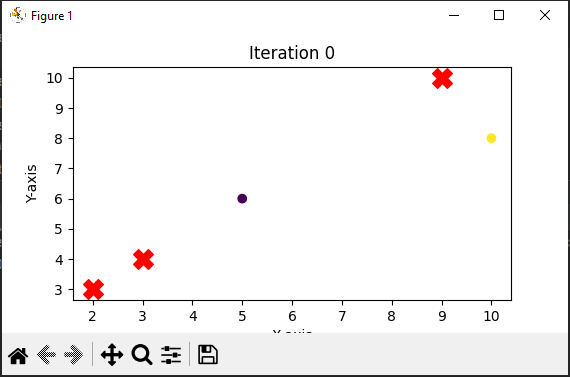
print(distances)

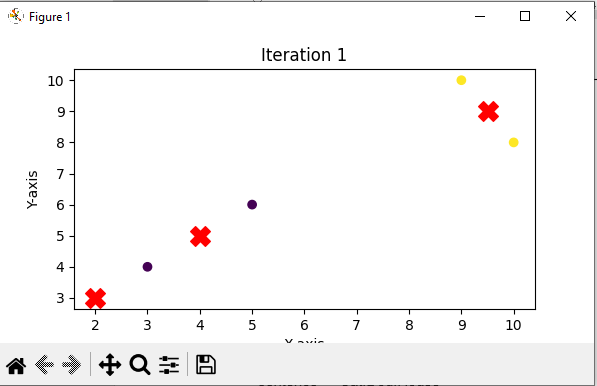
print("Final Cluster Assignments:", labels)

print("Final Cluster Centroids:", centroids)

OUTPUT







**POS**

import nltk

from nltk import pos\_tag

from nltk.tokenize import word\_tokenize

# Download necessary NLTK resources

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

# Sample sentence

sentence = "Amal,jyotthi,kanjirappally"

# Tokenize the sentence

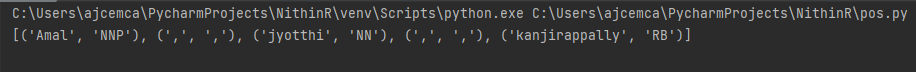
words = word\_tokenize(sentence)

# Perform POS tagging

pos\_tags = pos\_tag(words)

# Display the result

print(pos\_tags)

**Output**

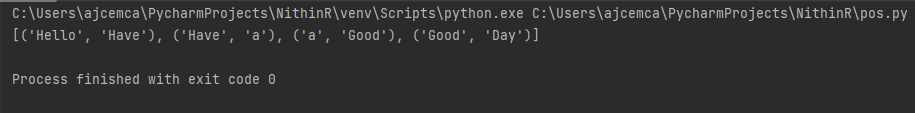
**N-gram Modeling**from nltk import bigrams,word\_tokenize

sentence="Hello Have a Good Day"

words=word\_tokenize(sentence)

bigrams\_list=list(bigrams(words))

print(bigrams\_list)

**Output**

**WebCrawler**

import requests

from bs4 import BeautifulSoup

def simple\_web\_crawler(url, max\_depth=2):

visited\_urls = set()

def crawl(url, depth):

if depth > max\_depth or url in visited\_urls:

return

print(f"Crawling: {url}")

try:

response = requests.get(url)

visited\_urls.add(url)

if response.status\_code == 200:

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

title = soup.title.string.strip() if soup.title else 'No title found'

print(f"Page Title: {title}")

for link in soup.find\_all('a', href=True):

next\_url = link['href']

crawl(next\_url, depth + 1)

except Exception as e:

print(f"Error crawling {url}: {e}")

crawl(url, depth=1)

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

start\_url = "https://aesajce.in"

simple\_web\_crawler(start\_url)

**Output**

