HINDUSTHAN COLLEGE OF ARTS & SCIENCE

(Autonomous)

An Autonomous Institution – Affiliated to Bharathiar University

(ISO 9001 – 2001 Certificate Instituation)

Behind Nava India, Coimbatore – 641028.

DEPARTMENT OF COMPUTER APPLICATIONS (PG)

MASTER OF COMPUTER APPLICATIONS

PRACTICAL RECORD

23MCP23 – PRACTICAL: AI & MACHINE LEARNING USING PYTHON

NAME	: ______________________
REGISTER NO	: ______________________
CLASS	: ______________________
SEMESTER	: ______________________
YEAR	: ______________________

HINDUSTHAN COLLEGE OF ARTS & SCIENCE

(Autonomous)

An Autonomous Institution – Affiliated to Bharathiar University

(ISO 9001 – 2001 Certificate Instituation)

Behind Nava India, Coimbatore – 641028.

DEPARTMENT OF COMPUTER APPLICATIONS (PG)

CERTIFICATE

Certificate that this is a bonafide record of **AI & Machine Learning Using Python (23MCP23)** done by_________________________Register No:____________________during the academic year of 2024-2025.

STAFF-IN CHARGE **DIRECTOR**

Submitted for the Bharathiar University Practical Examination held on ____________________at Hindusthan College of Arts & Science, Coimbatore – 641028.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Date:

Place: Coimbatore

CONTENTS

PROGRAM NO: 01 **DATE:**	**Tensor Flow Library**	**PAGE NO:**

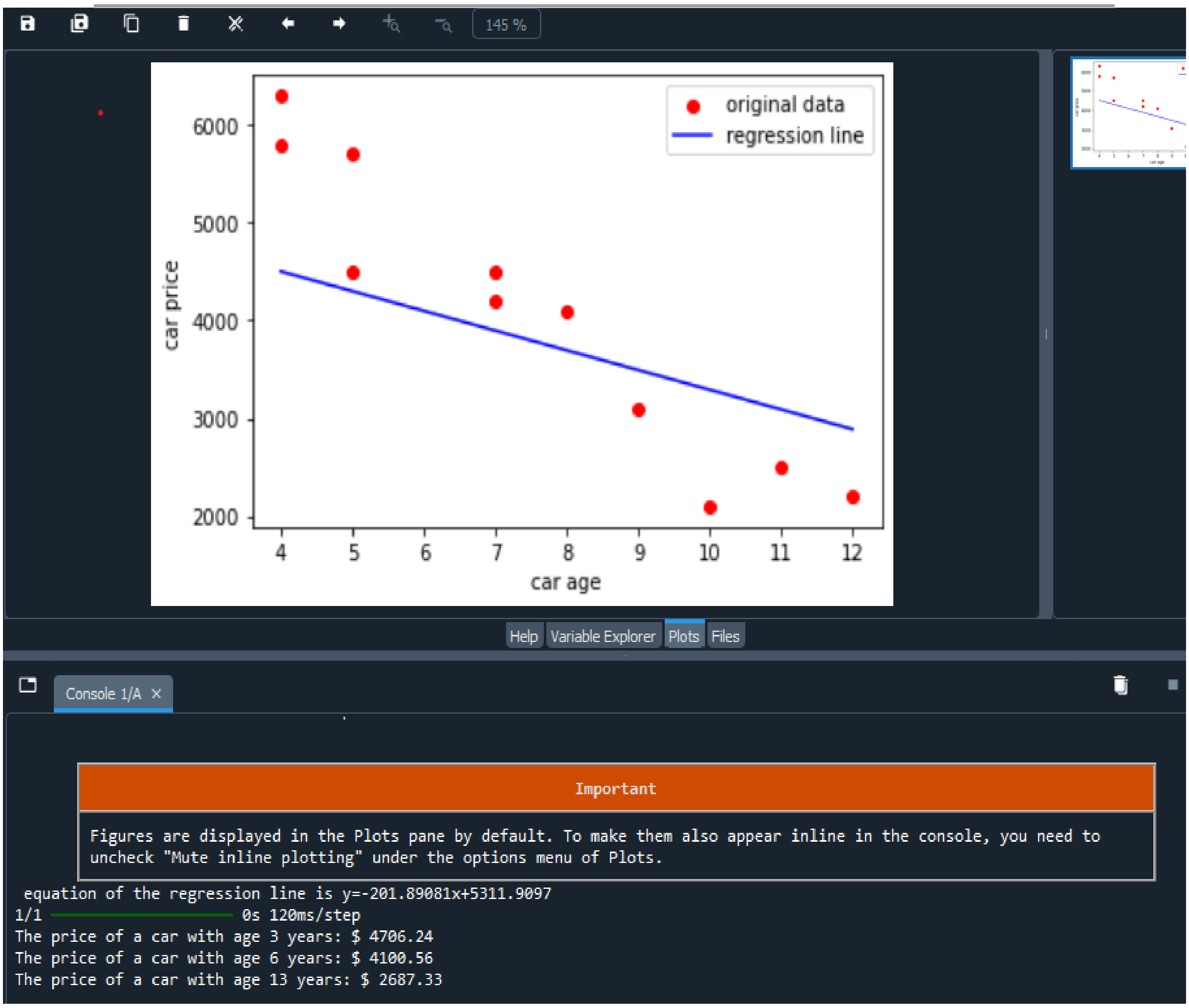
AIM:

ALGORITHM:

SOURCE CODE:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

car_age = np.array([4, 4, 5, 5, 7, 7, 8, 9, 10, 11, 12])
car_price = np.array([6300, 5800, 5700, 4500, 4500, 4200, 4100, 3100, 2100, 2500, 2200])
model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd',  loss='mean_squared_error')
model.fit(car_age, car_price, epochs=500)
predictions = model.predict(car_age)
plt.scatter(car_age, car_price, color='red', label='original data')
plt.plot(car_age, predictions, color='blue', label='regression line')
plt.xlabel('car age')
plt.ylabel('car price')
plt.legend()
plt.show()
slope = model.layers[0].kernel.numpy()[0][0]
intercept = model.layers[0].bias.numpy()[0]
print('equation of the regression line is y=' + str(slope) + 'x+' + str(intercept))
new_car_age = np.array([3, 6, 13])
new_predictions = model.predict(new_car_age)
for i, age in enumerate(new_car_age):
print(f"The price of a car with age {age} years: $ {new_predictions[i][0]:.2f}")
```

OUTPUT:



```
Important

Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to
uncheck "Mute inline plotting" under the options menu of Plots.
 equation of the regression line is y=-201.89081x+5311.9097
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
The price of a car with age 3 years: $ 4706.24
The price of a car with age 6 years: $ 4100.56
The price of a car with age 13 years: $ 2687.33
```

RESULT:

PROGRAM NO: 02 **DATE:**	**Maximum and Minimum element - NumPy**	**PAGE NO:**

AIM:

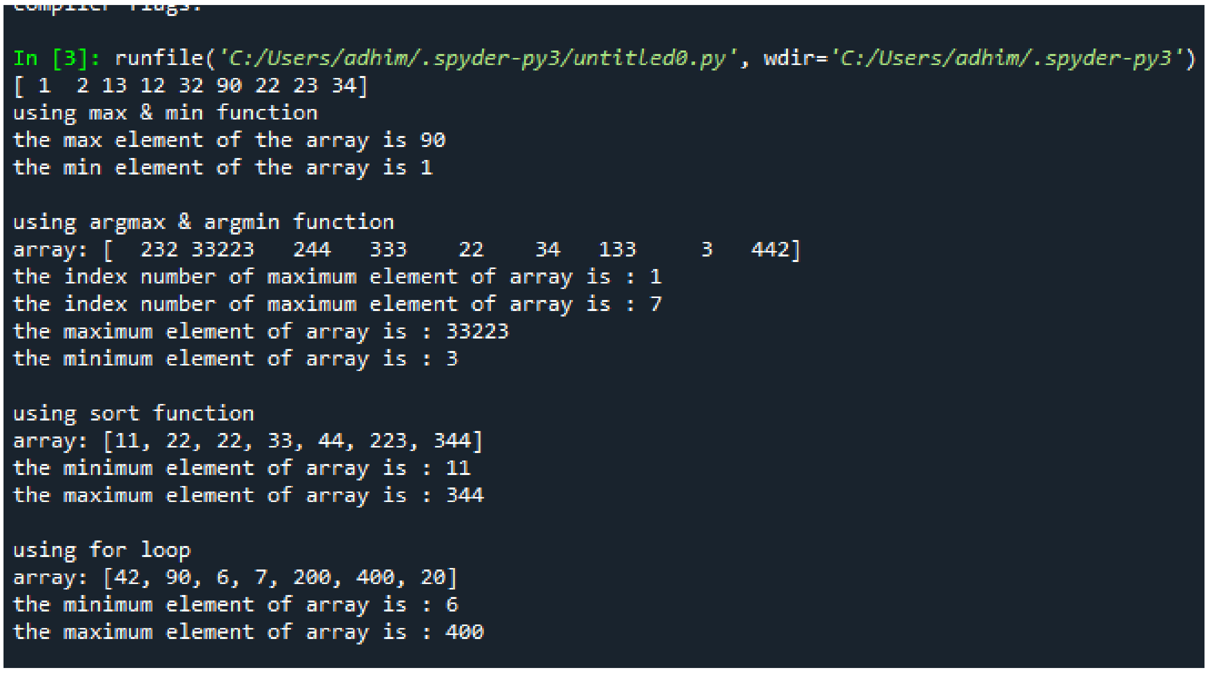
ALGORITHM:

SOURCE CODE:

```
import numpy as np
array = np.array([1,2,13,12,32,90,22,23,34])
print(array)
maximum = np.max(array)
minimum = np.min(array)
print("using max & min function")
print("the max element of the array is",maximum)
print("the min element of the array is",minimum)

arry=np.array([232,33223,244,333,22,34,133,3,442])
max_index=np.argmax(arry)
min_index=np.argmin(arry)
Max_elnt=arry[max_index]
Min_elnt=arry[min_index]
print("\nusing argmax & argmin function")
print("array:",arry)
print("the index number of maximum element of array is :",max_index)
print("the index number of maximum element of array is :",min_index)
print("the maximum element of array is :",Max_elnt)
print("the minimum element of array is :",Min_elnt)

ary=np.array=([223,344,22,33,44,22,11])
ary.sort()
print("\nusing sort function")
min_ele=ary[0]
max_ele=ary[-1]
print("array:",ary)
print("the minimum element of array is :",min_ele)
print("the maximum element of array is :",max_ele)

ary1=np.array=([42,90,6,7,200,400,20])
max=ary1[0]
min=ary1[0]
for i in ary1:
if i>max:
max=i
if i<min:
min=i
```

```
print("\nusing for loop")
print("array:",ary1)
print("the minimum element of array is :",min)
print("the maximum element of array is :",max)
```

OUTPUT:



```
In [3]: runfile('C:/Users/adhim/.spyder-py3/untitled0.py', wdir='C:/Users/adhim/.spyder-py3')
[ 1  2 13 12 32 90 22 23 34]
using max & min function
the max element of the array is 90
the min element of the array is 1

using argmax & argmin function
array: [  232 33223   244   333    22    34   133     3   442]
the index number of maximum element of array is : 1
the index number of maximum element of array is : 7
the maximum element of array is : 33223
the minimum element of array is : 3

using sort function
array: [11, 22, 22, 33, 44, 223, 344]
the minimum element of array is : 11
the maximum element of array is : 344

using for loop
array: [42, 90, 6, 7, 200, 400, 20]
the minimum element of array is : 6
the maximum element of array is : 400
```

RESULT:

PROGRAM NO: 03 **DATE:**	**Natural Language Processing**	**PAGE NO:**

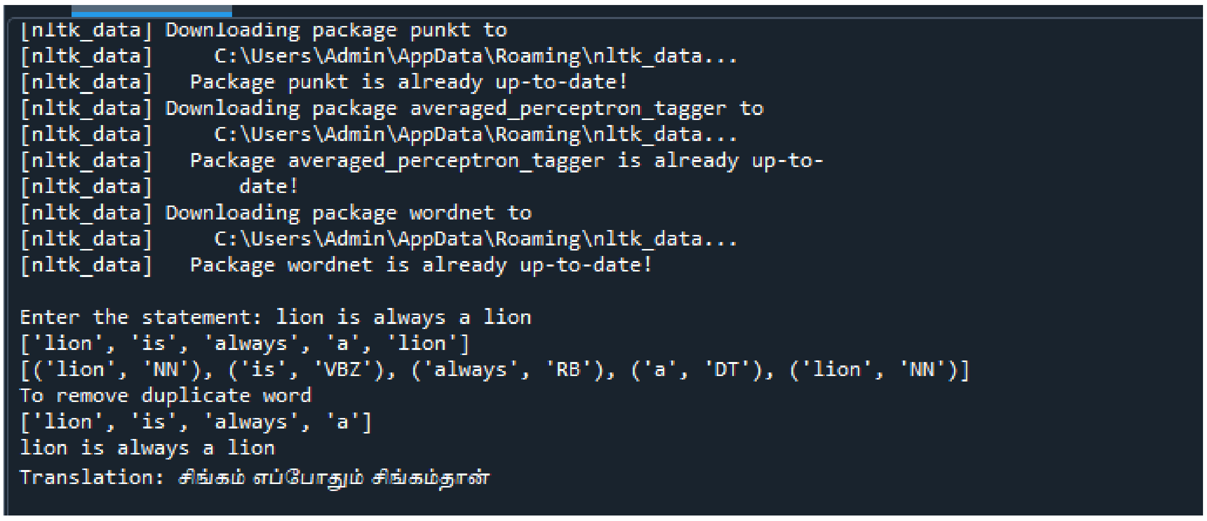
AIM:

ALGORITHM:

SOURCE CODE:

```
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
text=input("Enter the statement:")
words=word_tokenize(text)
print(words)
pos=pos_tag(words)
print(pos)
print("To remove duplicate word")
coll=set()
result=[]
for word in words:
    if word not in coll:
        coll.add(word)
        result.append(word)
print(result)
content=word_tokenize (text)
wnl=WordNetLemmatizer()
lemmatized_string=' '.join([wnl.lemmatize(words)for words in content])
print(lemmatized_string)

from translate import Translator
tr=Translator(to_lang="ta")
translated_text=tr.translate(text)
print("Translation:",translated_text)
```

OUTPUT:



```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!

Enter the statement: lion is always a lion
['lion', 'is', 'always', 'a', 'lion']
[('lion', 'NN'), ('is', 'VBZ'), ('always', 'RB'), ('a', 'DT'), ('lion', 'NN')]
To remove duplicate word
['lion', 'is', 'always', 'a']
lion is always a lion
Translation: சிங்கம் எப்போதும் சிங்கம்தான்
```

RESULT:

PROGRAM NO: 04 **DATE:**	**Series to Python List – Pandas**	**PAGE NO:**

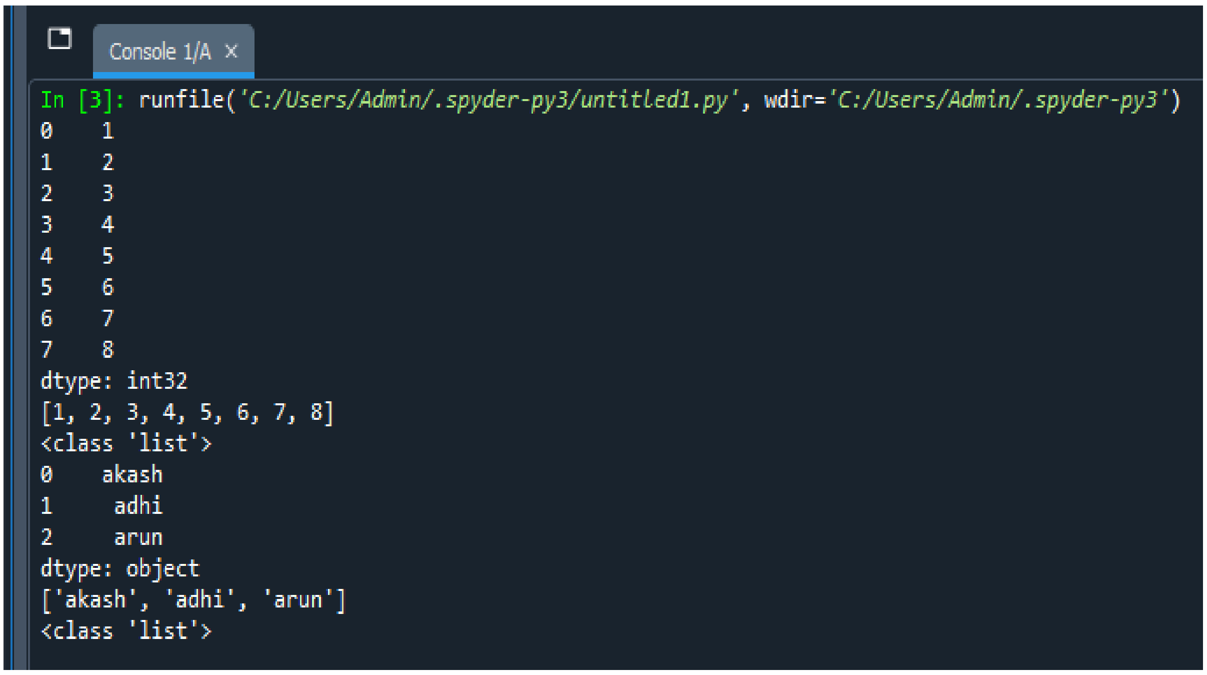
AIM:

ALGORITHM:

SOURCE CODE:

```
import numpy as np
import pandas as pd
array=np.array([1,2,3,4,5,6,7,8])
ser=pd.Series(array)
print(ser)

list=ser.tolist()
print(list)
print(type(list))

array1=np.array(["akash","adhi","arun"])
ser1=pd.Series(array1)
print(ser1)

list1=ser1.tolist();
print(list1)
print(type(list1))
```

OUTPUT:



```
In [3]: runfile('C:/Users/Admin/.spyder-py3/untitled1.py', wdir='C:/Users/Admin/.spyder-py3')
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int32
[1, 2, 3, 4, 5, 6, 7, 8]
<class 'list'>
0    akash
1     adhi
2     arun
dtype: object
['akash', 'adhi', 'arun']
<class 'list'>
```

RESULT:

PROGRAM NO: 05 **DATE:**	**Time Series Analysis**	**PAGE NO:**

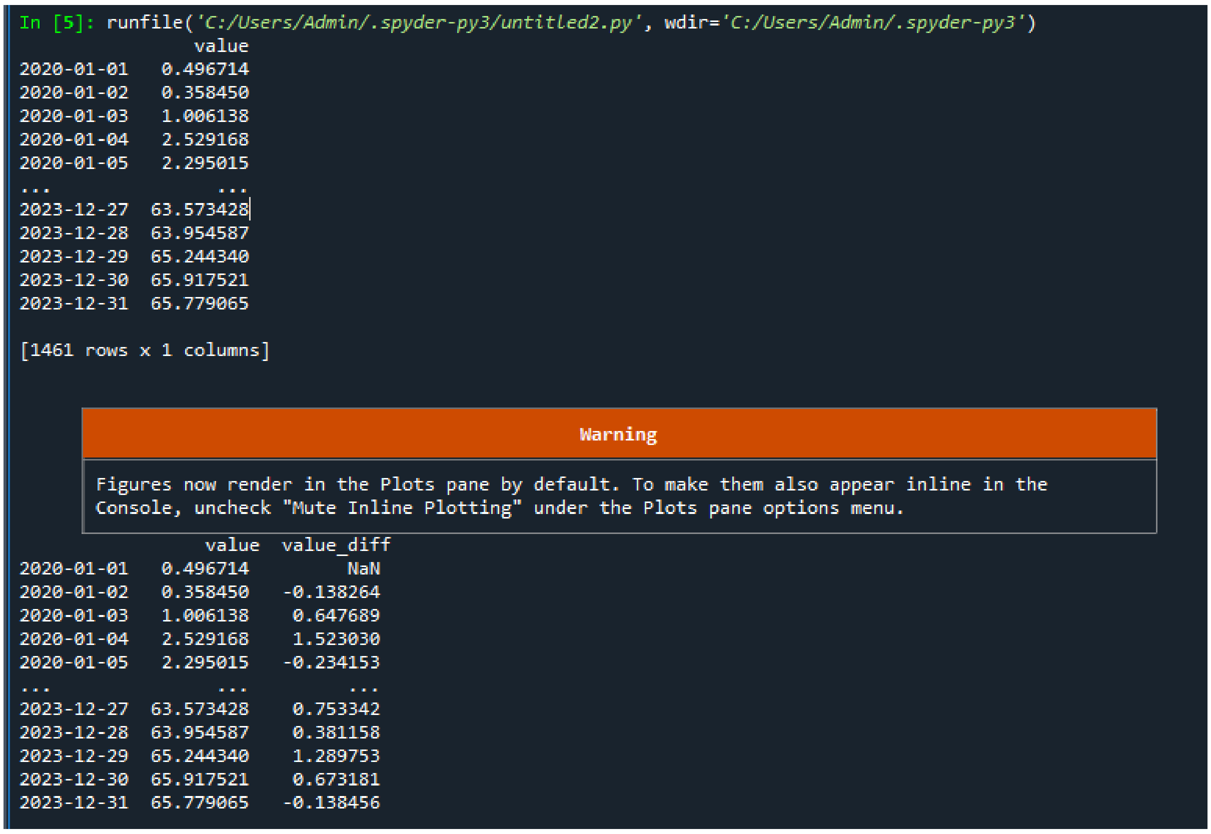
AIM:

ALGORITHM:

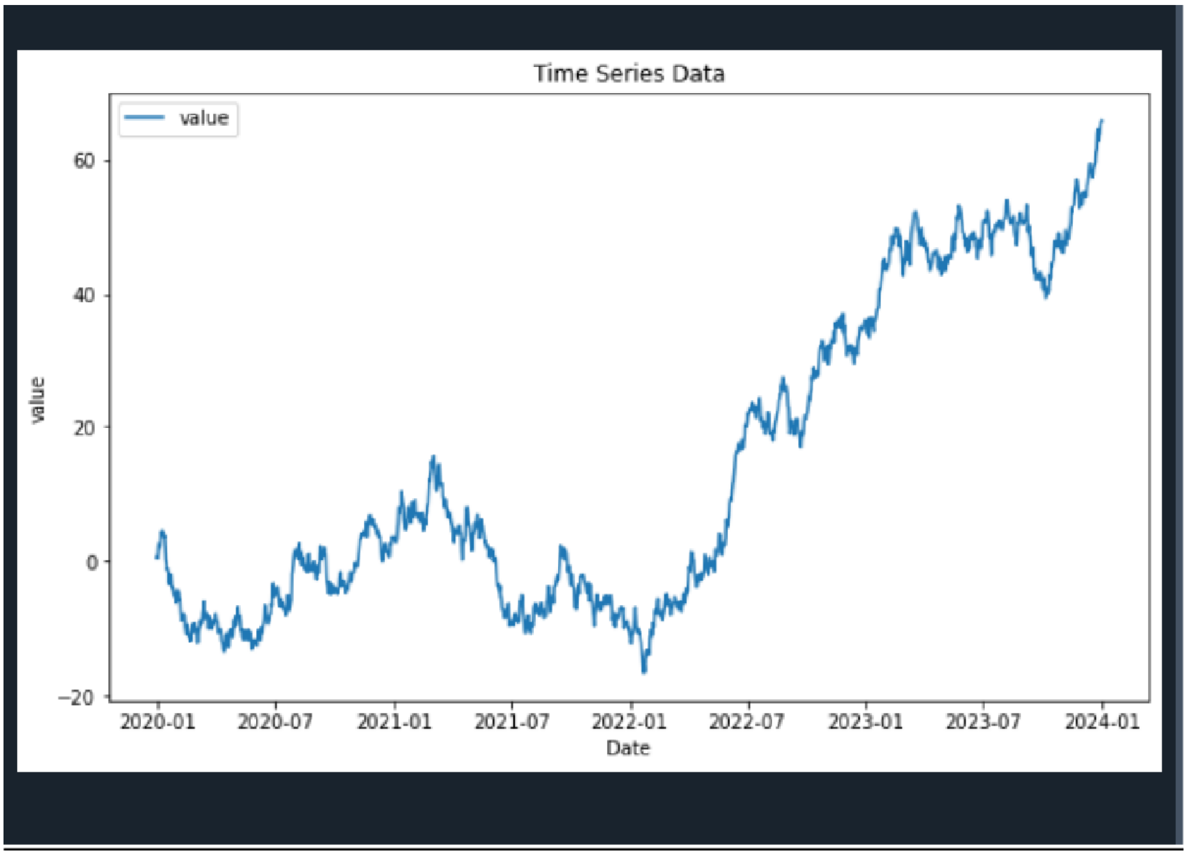
SOURCE CODE:

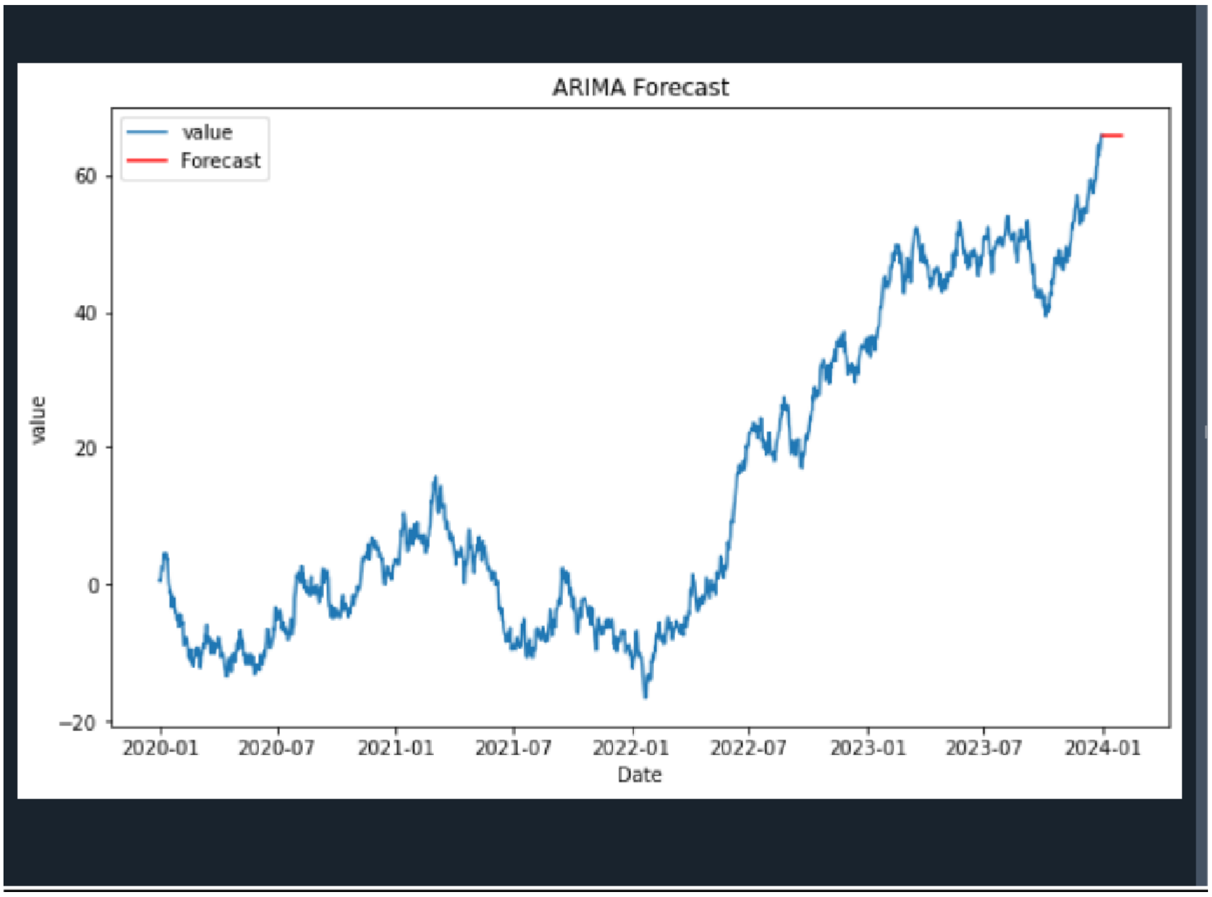
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

np.random.seed(42)
date_rng=pd.date_range(start='2020-01-01',end='2023-12-31',freq='D')
data=np.random.randn(len(date_rng)).cumsum()
df=pd.DataFrame(data,index=date_rng,columns=['value'])
print(df)
plt.figure(figsize=(10,6))
plt.plot(df.index,df['value'],label='value')
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('value')
plt.legend()
plt.show()

df['value_diff']=df['value'].diff().dropna()
print(df)
model=ARIMA(df['value'],order=(1,1,1))
model_fit=model.fit()
Forecast=model_fit.forecast(steps=30)
Forecast_dates=pd.date_range(start=df.index[-1]+pd.Timedelta(days=1),periods=30,freq='D')
plt.figure(figsize=(10,6))
plt.plot(df.index,df['value'],label='value')
plt.plot(Forecast_dates,Forecast,label='Forecast',color='red')
plt.title('ARIMA Forecast')
plt.xlabel('Date')
plt.ylabel('value')
plt.legend()
plt.show()
```

OUTPUT:



```
In [5]: runfile('C:/Users/Admin/.spyder-py3/untitled2.py', wdir='C:/Users/Admin/.spyder-py3')
                value
2020-01-01   0.496714
2020-01-02   0.358450
2020-01-03   1.006138
2020-01-04   2.529168
2020-01-05   2.295015
...               ...
2023-12-27  63.573428
2023-12-28  63.954587
2023-12-29  65.244340
2023-12-30  65.917521
2023-12-31  65.779065

[1461 rows x 1 columns]

                              Warning

        Figures now render in the Plots pane by default. To make them also appear inline in the
        Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.
                value  value_diff
2020-01-01   0.496714         NaN
2020-01-02   0.358450   -0.138264
2020-01-03   1.006138    0.647689
2020-01-04   2.529168    1.523030
2020-01-05   2.295015   -0.234153
...               ...         ...
2023-12-27  63.573428    0.753342
2023-12-28  63.954587    0.381158
2023-12-29  65.244340    1.289753
2023-12-30  65.917521    0.673181
2023-12-31  65.779065   -0.138456
```




ARIMA Forecast
value
Forecast
60
40
20
0
−20
value
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
Date

RESULT:

PROGRAM NO: 06 **DATE:**	**Clustering Algorithm**	**PAGE NO:**

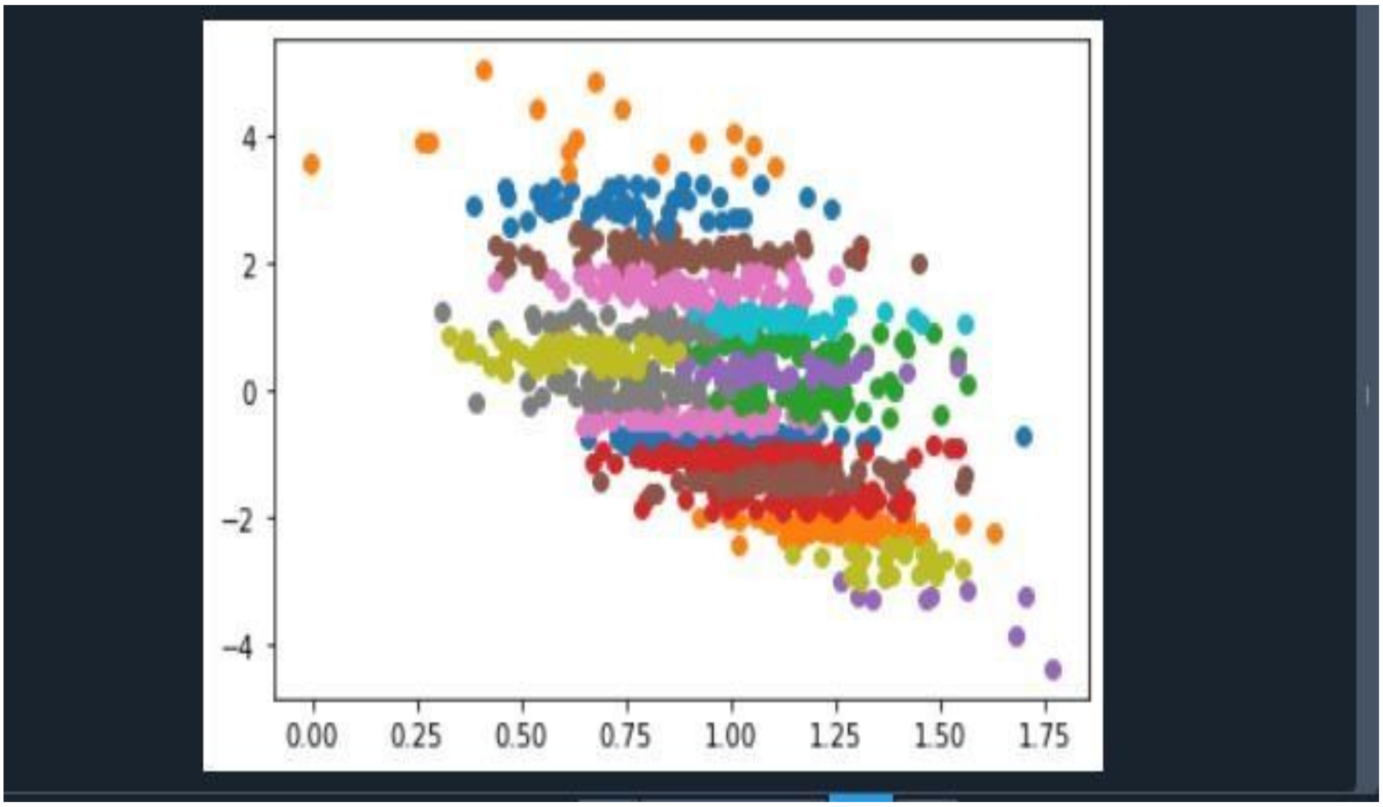
AIM:

ALGORITHM:

SOURCE CODE:

```
# affinity propagation clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import AffinityPropagation
from matplotlib import pyplot
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=None)
# define the model
model = AffinityPropagation(damping=0.9)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
        # get row indexes for samples with this cluster
        row_ix = where(yhat == cluster)
        # create scatter of these samples
        pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

OUTPUT:



RESULT:

PROGRAM NO: 07 **DATE:**	**Reinforcement Learning**	**PAGE NO:**

AIM:

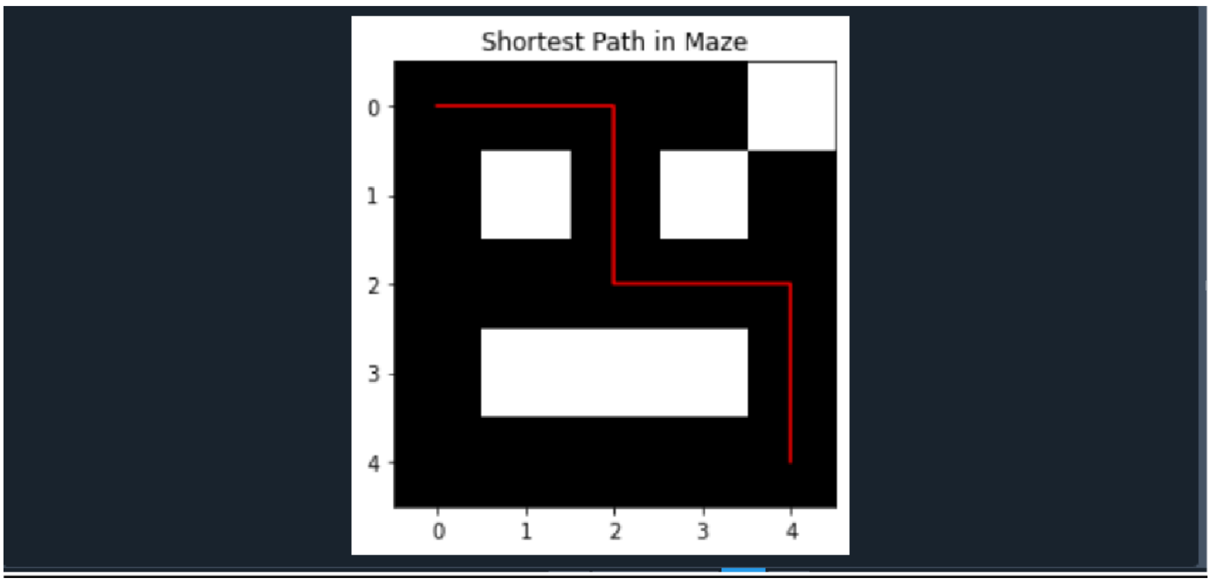
ALGORITHM:

SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as plt

# Maze environment
maze = np.array([[0, 0, 0, 0, 1], [0, 1, 0, 1, 0], [0, 0, 0, 0, 0], [0, 1, 1, 1, 0], [0, 0, 0, 0, 0]])
start_state = (0, 0)
goal_state = (4, 4)

# Q-learning parameters
actions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
q_table = np.zeros((maze.shape[0], maze.shape[1], len(actions)))
alpha, gamma, epsilon = 0.1, 0.9, 0.1
# Training loop
for episode in range(1000):
    state = start_state
    while state != goal_state:
        # Choose action
        if np.random.rand() < epsilon:
            action = np.random.choice(range(len(actions)))
        else:
            action = np.argmax(q_table[state[0], state[1], :])
        # Take action and update Q-table
        dx, dy = actions[action]
        next_state = (state[0] + dx, state[1] + dy)
        if 0 <= next_state[0] < maze.shape[0] and 0 <= next_state[1] < maze.shape[1] and
maze[next_state[0], next_state[1]] == 0:
            reward = 1 if next_state == goal_state else 0
            q_table[state[0], state[1], action] += alpha * (reward + gamma * np.max(q_table[next_state[0],
next_state[1], :]) - q_table[state[0], state[1], action])
            state = next_state
        else:
            reward = -1
            q_table[state[0], state[1], action] += alpha * (reward - q_table[state[0], state[1], action])
path = [start_state]
state = start_state
while state != goal_state:
    action = np.argmax(q_table[state[0], state[1], :])
    dx, dy = actions[action]
```

```
        state = (state[0] + dx, state[1] + dy)
        path.append(state)

    plt.imshow(maze, cmap="gray")
    plt.plot([p[1] for p in path], [p[0] for p in path], color="red")
    plt.title("Shortest Path in Maze")
    plt.show()
```

OUTPUT:



RESULT:

PROGRAM NO: 08 **DATE:**	**Keras Model**	**PAGE NO:**

AIM:

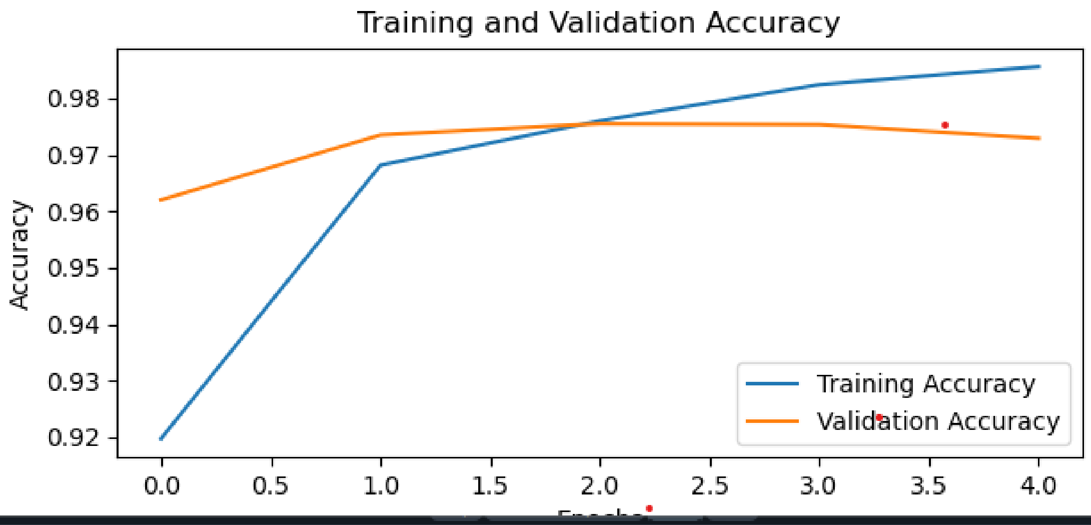
ALGORITHM:

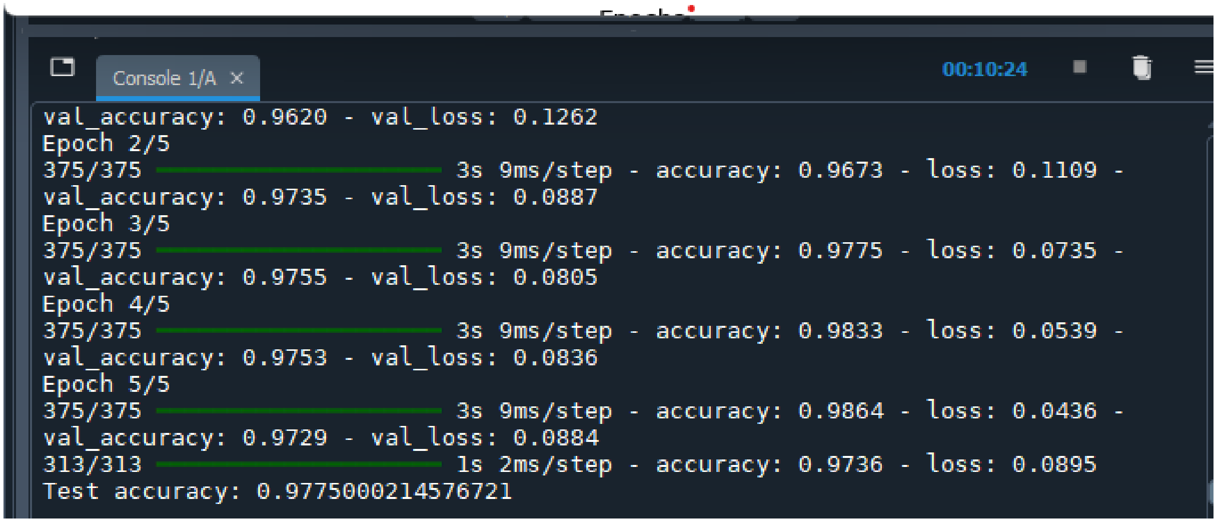
SOURCE CODE:

```
# Import necessary libraries
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

# Load and preprocess the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the pixel values (0 to 255) to the range 0 to 1
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# Reshape the data to fit the Keras model (28x28 pixels to 784 pixels in a flat array)
x_train = x_train.reshape((x_train.shape[0], 28 * 28))
x_test = x_test.reshape((x_test.shape[0], 28 * 28))

# One-hot encode the labels (0-9 digits)
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build the Keras model
model = models.Sequential()

# Input layer (28*28 = 784 nodes)
model.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))

# Dropout layer to prevent overfitting
model.add(layers.Dropout(0.2))

# Second hidden layer
model.add(layers.Dense(256, activation='relu'))

# Output layer (10 classes for 10 digits)
model.add(layers.Dense(10, activation='softmax'))
```

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model and store the history
history = model.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.2)

# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(x_test, y_test)

# Print test accuracy
print(f"Test accuracy: {test_acc}")

# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

OUTPUT:





```
val_accuracy: 0.9620 - val_loss: 0.1262
Epoch 2/5
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9673 - loss: 0.1109 -
val_accuracy: 0.9735 - val_loss: 0.0887
Epoch 3/5
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9775 - loss: 0.0735 -
val_accuracy: 0.9755 - val_loss: 0.0805
Epoch 4/5
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9833 - loss: 0.0539 -
val_accuracy: 0.9753 - val_loss: 0.0836
Epoch 5/5
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9864 - loss: 0.0436 -
val_accuracy: 0.9729 - val_loss: 0.0884
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9736 - loss: 0.0895
Test accuracy: 0.9775000214576721
```

RESULT:

PROGRAM NO: 09 **DATE:**	**Exponents , trigonometric Problem- Scipy**	**PAGE NO:**

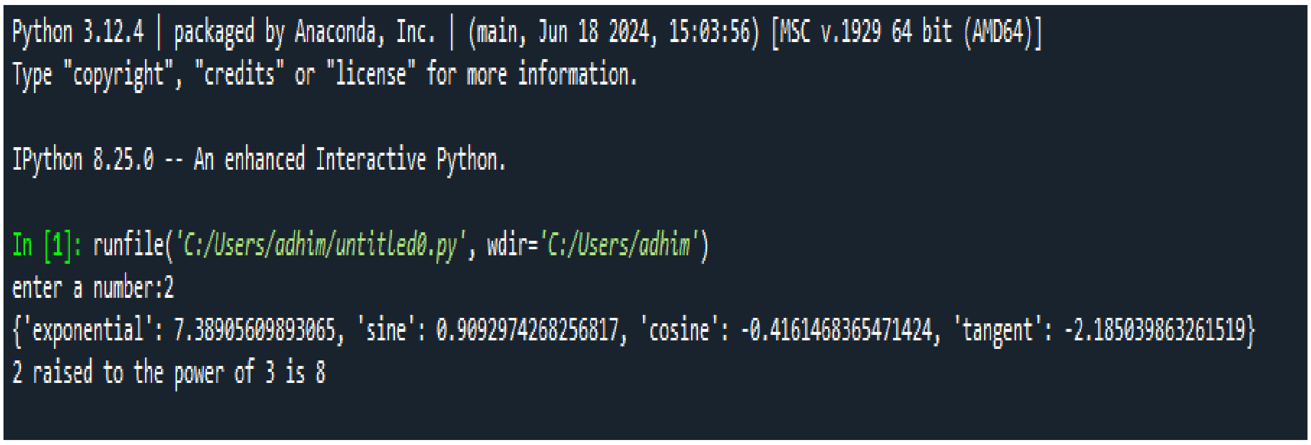
AIM:

ALGORITHM:

SOURCE CODE:

```
 import numpy as np
 from scipy import special
x = int(input("Enter a number: "))
# Calculate exponential and trigonometric values
 exp_value = np.exp(x)
 sin_value = np.sin(x)
cos_value = np.cos(x)
tan_value = np.tan(x)
results = {
    'exponential': exp_value,
    'sine':  sin_value,
    'cosine': cos_value,
    'tangent': tan_value,
 }
print(results)
 base = 2
 exp = 3
# Use the built-in pow function to calculate the exponent
 result_exp = pow(base, exp)
 print(f"{base} raised to the power of {exp} is {result_exp}")
```

OUTPUT:



```
Python 3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.25.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/adhim/untitled0.py', wdir='C:/Users/adhim')
enter a number:2
{'exponential': 7.38905609893065, 'sine': 0.9092974268256817, 'cosine': -0.4161468365471424, 'tangent': -2.185039863261519}
2 raised to the power of 3 is 8
```

RESULT: