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 & ML 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 & ML 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	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test /255.0

model = Sequential([
   Flatten(input_shape=(28, 28)),
   Dense(128, activation='relu'),
   Dense(10)
])

model.compile(optimizer='adam',
          loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
          metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))

test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'Test accuracy: {test_acc}')
```

OUTPUT:

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ **1s** 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
super().__init__(**kwargs)

Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ **14s** 7ms/step - accuracy: 0.8828 - loss: 0.4233 - val_accuracy: 0.9627 - val_loss: 0.1252
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ **10s** 5ms/step - accuracy: 0.9656 - loss: 0.1176 - val_accuracy: 0.9728 - val_loss: 0.0908
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ **10s** 5ms/step - accuracy: 0.9785 - loss: 0.0721 - val_accuracy: 0.9736 - val_loss: 0.0869
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ **10s** 5ms/step - accuracy: 0.9835 - loss: 0.0546 - val_accuracy: 0.9771 - val_loss: 0.0766
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ **8s** 4ms/step - accuracy: 0.9869 - loss: 0.0412 - val_accuracy: 0.9761 - val_loss: 0.0764
313/313 - 1s - 2ms/step - accuracy: 0.9761 - loss: 0.0764
Test accuracy: 0.9761000275611877

RESULT:

PROGRAM NO: 02 DATE:	Searching Maximum and Minimum element using NumPy	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
import numpy as np
array=np.array([69,96,99,14,3])
max_element=np.max(array)
min_element=np.min(array)
print("Maximum element in the array:",max_element)
print("Minimum element in the array:",min_element)
```

OUTPUT:

```
Maximum element in the array: 99
Minimum element in the array: 3
```

RESULT:

PROGRAM NO: 03 DATE:	Natural Language Processing	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
from nltk.chat.util import Chat, reflections
pairs =[
    ['(hi|hello|hey|holla|hola)', ['Hey there !', 'Hi there !', 'Hey !']],
    ['(what is your name ?)', ['I am Monkey D Luffy']],
    ['(what do you do ?)', ['I am Emperor of the sea']],
    ['(what is your power ?)', ['I am Sun god NIKA!']]
]

chat = Chat(pairs, reflections)
chat.converse()
```

OUTPUT:

```
>hi
Hi there !
>what is your name
I am Monkey D Luffy
>what do you do
I am Emperor of the sea
>what is your power
I am Sun god NIKA!
>
```

RESULT:

PROGRAM NO: 04 DATE:	Convert a Pandas Module Series to Python List	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
import pandas as pd
mk=pd.Series([1,2,3,4,5,6])
print(mk)
print(type(mk))
print("Pandas Series to python list:")
print(mk.tolist())
print(type(mk.tolist()))
```

OUTPUT:

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
<class 'pandas.core.series.Series'>
Pandas Series to python list:
[1, 2, 3, 4, 5, 6]
<class 'list'>
```

RESULT:

PROGRAM NO: 05 DATE:	Time Series Analysis	PAEG NO:

AIM:

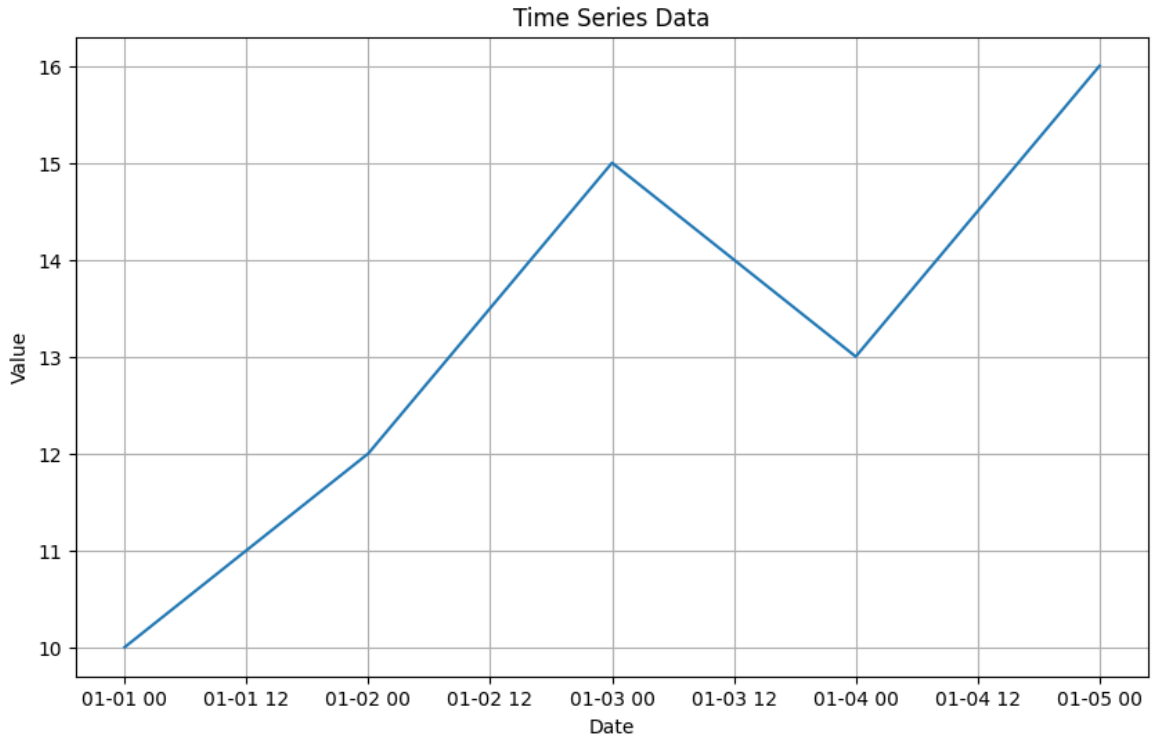
ALGORITHM:

SOURCE CODE:

```
import pandas as pd
import matplotlib.pyplot as plt

data={'Date': pd.to_datetime(['2023-01-01','2023-01-02','2023-01-03','2023-01-
04','2023-01-05']),'Value':[10,12,15,13,16]}

df=pd.DataFrame(data)
df.set_index('Date',inplace=True)

print(df)
print(df.describe())

plt.figure(figsize=(10,6))
plt.plot(df.index,df['Value'])
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.show()
```

OUTPUT:

```
            Value
Date
2023-01-01     10
2023-01-02     12
2023-01-03     15
2023-01-04     13
2023-01-05     16

            Value
count        5.000000
mean        13.200000
std          2.387467
min         10.000000
25%         12.000000
50%         13.000000
75%         15.000000
max         16.000000
```


Time Series Data
16
15
14
13
12
11
10
Value
01-01 00
01-01 12
01-02 00
01-02 12
01-03 00
01-03 12
01-04 00
01-04 12
01-05 00
Date

RESULT:

PROGRAM NO: 06 DATE:	Clustering Algorithm	PAEG NO:

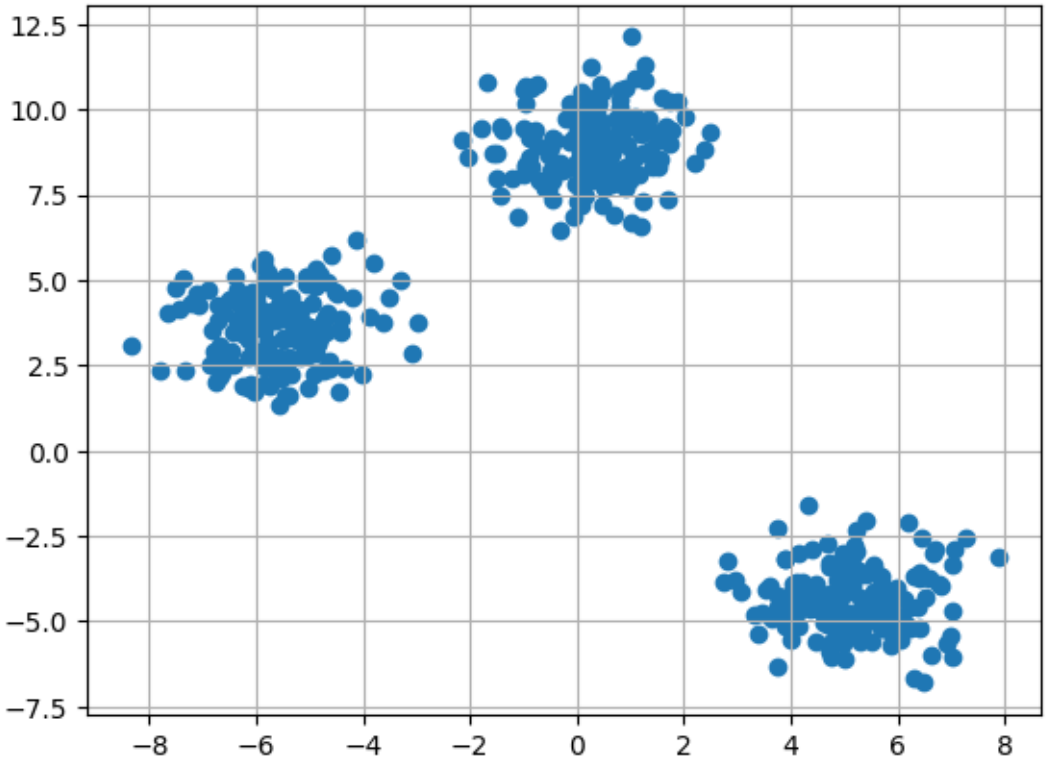
AIM:

ALGORITHM:

SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

x,y=make_blobs(n_samples=500,n_features=2,centers=3,random_state=23)

fig=plt.figure(0)
plt.grid(True)
plt.scatter(x[:,0],x[:,1])
plt.show()
```

OUTPUT:



RESULT:

PROGRAM NO: 07 DATE:	Reinforcement Learning	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
import random
import gym

env=gym.make('CartPole-v1',render_mode='human')

episodes=10
for episode in range(1,episodes+1):
    state=env.reset()
    done=False
    score=0

    while not done:
      action=random.choice([0,1])
      n_state,reward,done,info=env.step(action)
      score+=reward
      env.render()
    print('Episode:{} Score:{}'.format(episode,score))
env.close()
```

OUTPUT:

```
Episode:1 Score:39.0
Episode:2 Score:17.0
Episode:3 Score:18.0
Episode:4 Score:25.0
Episode:5 Score:29.0
Episode:6 Score:38.0
Episode:7 Score:17.0
Episode:8 Score:11.0
Episode:9 Score:38.0
Episode:10 Score:17.0
```

RESULT:

PROGRAM NO: 08 DATE:	Keras Model	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
  keras.layers.Dense(10, activation='relu', input_shape=(2,)),
  keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mse')

import numpy as np
x_train = np.random.rand(100, 2)
y_train = np.dot(x_train, [2., 3.]) + 1

model.fit(x_train, y_train, epochs=10)

x_test = np.random.rand(10, 2)
predictions = model.predict(x_test)

print(predictions)
```

OUTPUT:

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 11.0643
Epoch 2/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.9817
Epoch 3/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 10.9532
Epoch 4/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 10.8204
Epoch 5/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 10.4963
Epoch 6/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.5292
Epoch 7/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 10.4133
Epoch 8/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.1161
Epoch 9/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.9829
Epoch 10/10
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 10.0117
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step
[[0.30706084]
 [0.40062627]
 [0.24234425]
 [0.4796983 ]
 [0.22320671]
 [0.38175967]
 [0.3056984 ]
 [0.5155654 ]
 [0.42961207]
 [0.34717298]]
```

RESULT:

PROGRAM NO: 09 DATE:	Finding exponents and trigonometric Problems using Scipy	PAEG NO:

AIM:

ALGORITHM:

SOURCE CODE:

```
import numpy as np
import scipy.special as sp

sin_value=np.sin(np.pi/2)
cos_value=np.cos(np.pi/2)
tan_value=np.tan(np.pi/2)
print(f"sin({np.pi/2})={sin_value}")
print(f"cos({np.pi/2})={cos_value}")
print(f"tan({np.pi/2})={tan_value}")

asin_value=np.arcsin(0.5)
acos_value=np.arccos(0.5)
atan_value=np.arctan(0.5)

print(f"arcsin(0.5)={np.degrees(asin_value)} degrees")
print(f"arccos(0.5)={np.degrees(acos_value)} degrees")
print(f"arctan(0.5)={np.degrees(atan_value)} degrees")
```

OUTPUT:

```
sin(1.5707963267948966)=1.0
cos(1.5707963267948966)=6.123233995736766e-17
tan(1.5707963267948966)=1.633123935319537e+16
arcsin(0.5)=30.000000000000004 degrees
arccos(0.5)=60.00000000000001 degrees
arctan(0.5)=26.56505117707799 degrees
```

RESULT: