

PROGRAM-1

Write a program in python to add two numbers.

CODE:

```
num1 = int(input("Enter 1st number:"))  
  
num2 = int(input("Enter 2nd number:"))  
  
sum_of_num = num1+num2  
  
print("Sum of the numbers is:",sum_of_num)
```

OUTPUT:

```
Enter 1st number:30  
Enter 2nd number:20  
Sum of the numbers is:50
```

PROGRAM-2

Write a program in python to find factorial of any number.

CODE:

```
num = int(input("Enter your number:"))
factorial = 1

if num == 1 or num == 0:
    print("Factorial of 1 and 0 is 1.")

elif num < 0:
    print("Sorry! Factorial of less than 0 doesn't exists.")

else:
    for i in range(1,num+1):
        factorial = factorial*i
    print("Factorial of",num,"is",factorial)
```

OUTPUT:

```
Enter your number:5
Factorial of 5 is 120
```

PROGRAM-3

Write a program in python to print table of any number.

CODE:

```
num = int(input("Enter the number whose table you want:"))

print("Here is the table of",num)

for i in range(1,11):
    table = num*i
    print(num,"x",i,"=",table)
```

OUTPUT:

```
Enter the number whose table you want:2
Here is the table of 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

PROGRAM-4

Write a program in python to find even and odd number.

CODE:

```
num = int(input("Enter your number:"))

if num%2 == 0:
    print(num, "is even number.")
else:
    print(num, "is odd number.")
```

OUTPUT:

For Even

```
Enter your number:24
24 is even number.
```

For Odd

```
Enter your number:25
25 is odd number.
```

PROGRAM-5

Write a program in python to create Numpy array.

CODE:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

print(type(arr))
```

OUTPUT:

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

PROGRAM-6

Write a program in python to Numpy array slicing.

CODE:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

OUTPUT:

[2,3,4,5]

PROGRAM-7

Write a program in python for Scipy constants.

CODE:

```
from scipy import constants
print(constants.mega)
print(constants.kilo)
print(constants.hecto)
print(constants.deka)
print(constants.deci)
```

OUTPUT:

```
1000000.0
1000.0
100.0
10.0
0.1
```

PROGRAM-8

Find root of equation $x + \cos(x)$ using Scipy optimizers.

CODE:

```
from scipy.optimize import root
from math import cos

def eqn(x):
    return x + cos(x)

myroot = root(eqn, 0)
print(myroot.x)
```

OUTPUT:

```
[-0.73908513]
```


PROGRAM-9

Write a program in python for creating Dataframe using pandas.

CODE:

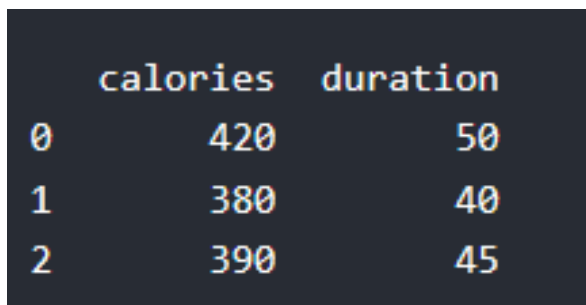
```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

df = pd.DataFrame(data)

print(df)
```

OUTPUT:



	calories	duration
0	420	50
1	380	40
2	390	45

PROGRAM-10

Write a program in python for creating Series using pandas.

CODE:

```
import pandas as pd
```

```
a = [1, 7, 2]
```

```
myvar = pd.Series(a)
```

```
print(myvar)
```

OUTPUT:

```
0    1
1    7
2    2
dtype: int64
```

PROGRAM-11

Write a program in python for loading IRIS dataset.

CODE:

```
from sklearn.datasets import load_iris

iris = load_iris()

X = iris.data

y = iris.target

feature_names = iris.feature_names

target_names = iris.target_names

print("Feature names:", feature_names)

print("Target names:", target_names)

print("\nFirst 5 rows of X:\n", X[:5])
```

OUTPUT:

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', '
petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']

First 5 rows of X:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

PROGRAM-12

Write a program in python for exploring Iris dataset using machine learning.

CODE:

```
from sklearn.datasets import load_iris

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

iris = load_iris()

X = iris.data

y = iris.target

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

classifier_knn = KNeighborsClassifier(n_neighbors=3)

classifier_knn.fit(X_train, y_train)

y_pred = classifier_knn.predict(X_test)

# Finding accuracy by comparing actual response values(y_test)with predicted # response
value(y_pred)

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Providing sample data and the model will make prediction out of that data

sample = [[5, 5, 3, 2], [2, 4, 3, 5]]

preds = classifier_knn.predict(sample)

pred_species = [iris.target_names[p] for p in preds]

print("Predictions:",pred_species)
```

OUTPUT:

```
Accuracy: 0.9833333333333333  
Predictions: ['versicolor', 'virginica']
```

PROGRAM-13

Write a program in python to create basic neural network using Tensorflow and keras.

CODE:

```
# MLP with manual validation set

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.model_selection import train_test_split

import numpy

# fix random seed for reproducibility

seed = 7

numpy.random.seed(seed)

# load pima indians dataset

dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables

X = dataset[:,0:8]

Y = dataset[:,8]

# split into 67% for train and 33% for test

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=seed)

# create model

model = Sequential()

model.add(Dense(12, input_dim=8, activation='relu'))
```

```

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# Compile model

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model

model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=150, batch_size=10)

```

OUTPUT:

```

Epoch 1/150
52/52 [=====] - 2s 15ms/step - loss: 5.6673 - accuracy: 0.6128 - val_loss: 2.1919 - val_accuracy: 0.5197
Epoch 2/150
52/52 [=====] - 0s 5ms/step - loss: 1.5381 - accuracy: 0.6304 - val_loss: 1.3713 - val_accuracy: 0.6102
Epoch 3/150
52/52 [=====] - 0s 4ms/step - loss: 1.2705 - accuracy: 0.6206 - val_loss: 1.1048 - val_accuracy: 0.6457
Epoch 4/150
52/52 [=====] - 0s 4ms/step - loss: 1.1138 - accuracy: 0.6420 - val_loss: 1.0053 - val_accuracy: 0.6614
Epoch 5/150
52/52 [=====] - 0s 4ms/step - loss: 1.0738 - accuracy: 0.6342 - val_loss: 0.9303 - val_accuracy: 0.6496
Epoch 6/150
52/52 [=====] - 0s 4ms/step - loss: 0.9940 - accuracy: 0.6440 - val_loss: 0.8808 - val_accuracy: 0.6654

```

PROGRAM-14

Write a program in python to implement “Tokenization” using NLTK.

CODE:

```
from nltk.tokenize import sent_tokenize, word_tokenize

text=" SRMIST aspires to be one of the best private universities in the world by the year 2025"

print(sent_tokenize(text))

print(word_tokenize(text))
```

OUTPUT:

```
['SRMIST aspires to be one of the best private universities in the world by the year 2025']
['SRMIST', 'aspires', 'to', 'be', 'one', 'of', 'the', 'best', 'private', 'universities', 'in', 'the', 'world', 'by', 'the', 'year', '2025']
```


PROGRAM-15

Write a program in python to implement “stopwords” using NLTK.

CODE:

```
# program filters stop words from the data.

from nltk.tokenize import sent_tokenize, word_tokenize

from nltk.corpus import stopwords

data = "All work and no play makes jack dull boy."

stopWords = set(stopwords.words('english'))

words = word_tokenize(data)

wordsFiltered = []

for w in words:

    if w not in stopWords:

        wordsFiltered.append(w)

print(wordsFiltered)
```

OUTPUT:

```
['All', 'work', 'play', 'makes', 'jack', 'dull', 'boy', '.']
```

PROGRAM-16

Write a program in python to implement “Lemmatization” using NLTK.

CODE:

```
import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

lemmatizer.lemmatize("books")
```

OUTPUT:

```
'book'
```

PROGRAM-17

Write a program in python to implement “Stemming” using NLTK.

CODE:

```
from nltk.stem import PorterStemmer

ps = PorterStemmer()

words = ["program", "programs", "programmer", "programming", "programmers"]

for w in words:

    print(w, " : ", ps.stem(w))
```

OUTPUT:

```
program : program
programs : program
programmer : programm
programming : program
programmers : program
```

PROGRAM-18

Write a program in python to implement “POS tagging” using NLTK.

CODE:

```
import nltk

from nltk import word_tokenize

sentence="Hello World! This is NLTK Program."

print(nltk.pos_tag(word_tokenize(sentence)))
```

OUTPUT:

```
[('Hello', 'NNP'), ('World', 'NNP'), ('!', '.'), ('This', 'DT'), ('is', 'VBZ'), ('NLTK', 'JJ'), ('Program', 'NNP'), ('.', '.')]
```

PROGRAM-19

Write a program in python to create scatterplot using Matplotlib.

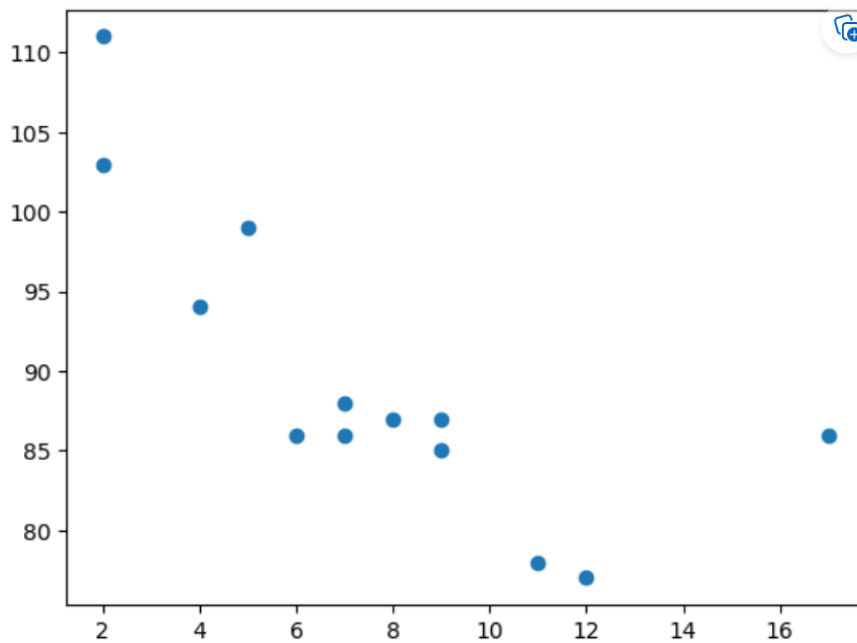
CODE:

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

OUTPUT:



PROGRAM-20

Write a program in python to create histogram using Matplotlib.

CODE:

```
import matplotlib.pyplot as plt

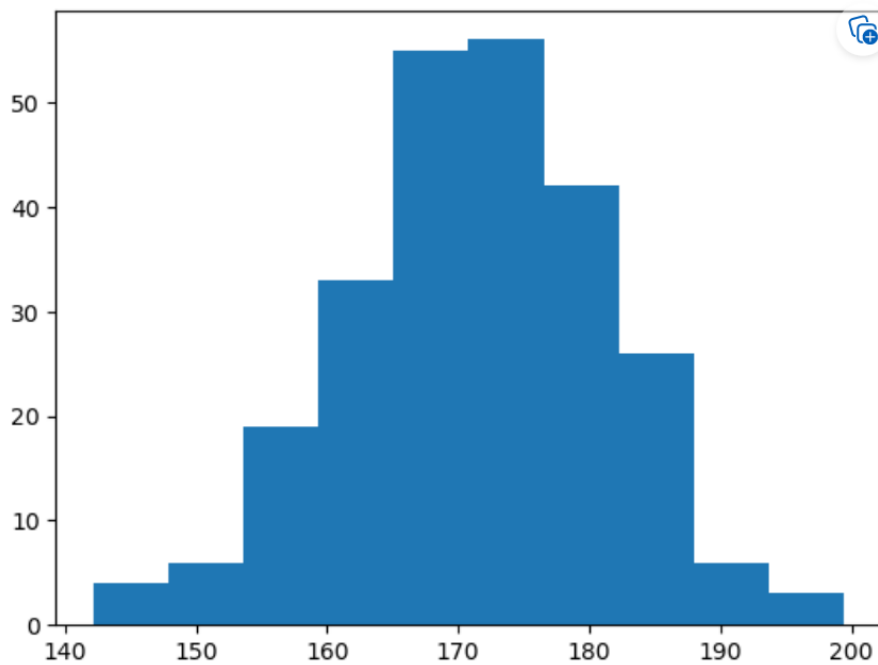
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x)

plt.show()
```

OUTPUT:



PROGRAM-21

Write a program in python to create pie-chart using Matplotlib.

CODE:

```
import matplotlib.pyplot as plt  
  
import numpy as np  
  
y = np.array([35, 25, 25, 15])  
  
plt.pie(y)  
  
plt.show()
```

OUTPUT:

