EXPERIMENT-1

<u>AIM</u>: Write a program to perform the following operations on Google Colab:

(i) Upload a file to colab.

(ii) Download a file from colab.

(iii) Change the colab runtime.

(iv) Install packages in colab.

(v) Unzip a file in colab.

(vi) Using matplotlib library for visualisation.

(vii) Exploring the numpy library in python to perform fundamental operations on arrays.

<u>CODE and OUTPUT:</u>

(i) Upload a file to colab

```
from google.colab import files

uploaded = files.upload()
```

```
Choose Files No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving cleaned_dataset.csv to cleaned_dataset.csv
```

(ii) Download a file from colab.

```
[ ] files.download('cleaned_dataset.csv')
```

(iii) Change the colab runtime.

First execute the following commands.

```
[ ] import torch
    torch.cuda.is_available()
```

```
False
```

Now, Go to Runtime -> Change Runtime Type -> Select GPU as Hardware accelerator and again run the following code cell to see GPU is enabled or not.

```
[ ] import torch
    torch.cuda.is_available()
```

```
True
```

(iv) Install packages in colab.

```
!pip install polyglot
```

```
Collecting polyglot
  Downloading https://files.pythonhosted.org/packages/e7/98/e24e2489114c5112b083714277204d92d372f5bbe00d5507acf40370edb9/polyglot-16.7.4.tar.gz (
     |████████████████████████████████| 133kB 7.9MB/s
Building wheels for collected packages: polyglot
  Building wheel for polyglot (setup.py) ... done
  Created wheel for polyglot: filename=polyglot-16.7.4-py2.py3-none-any.whl size=52557 sha256=9f5a21066c1f931a74ddd36e9ea2786867b5859c81f6a403d0b
  Stored in directory: /root/.cache/pip/wheels/5e/91/ef/f1369fdc1203b0a9347d4b24f149b83a305f39ab047986d9da
Successfully built polyglot
Installing collected packages: polyglot
Successfully installed polyglot-16.7.4
```

(v) Unzip a file in colab.

```
!unzip english_dataset.zip
```

```
Archive:  english_dataset.zip
   creating: english_dataset/
  inflating: __MACOSX/._english_dataset
  inflating: english_dataset/english_dataset.tsv
  inflating: __MACOSX/english_dataset/._english_dataset.tsv
  inflating: english_dataset/hasoc2019_en_test-2919.tsv
  inflating: __MACOSX/english_dataset/._hasoc2019_en_test-2919.tsv
```

(vi) Using matplotlib library for visualisation.

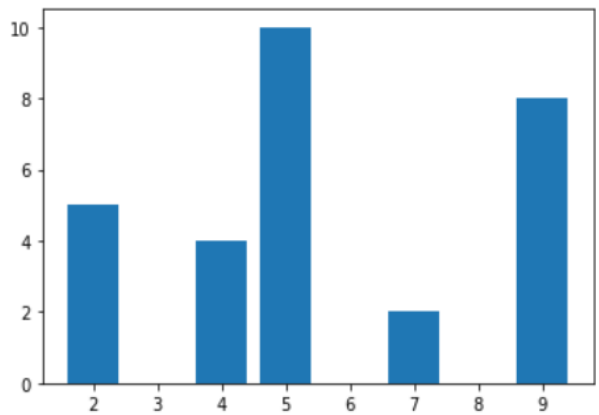
```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot the bar
plt.bar(x,y)

# function to show the plot
plt.show()
```



(vii) Exploring the numpy library in python to perform fundamental operations on arrays.

(a) Print array size:

```
#WAP to implement numpy library in python
import numpy as np
arr = np.array( [[ 1, 2, 3], [ 4, 5, 6]] )
#print array size
print ("Size of array:", arr.size)
```

```
Size of array: 6
```

(b) Iterate array using nditer:

```
#iterate array using nditer
print("2.Iteration over array")
for x in np.nditer(arr):
  print(x)
```

```
2.Iteration over array
1
2
3
4
5
6
```

(c) Create an array from tuple

```
#create array from tuple
arrx = np.array((1,2,3))
print("\n3.Array from tupple:\n",arrx)
```

```
3.Array from tupple:
 [1 2 3]
```

(d) Create 3x4 array with all zeros using np.zeroes

```
#create 3x4 array with all zeros using np.zeroes
arr1 = np.zeros((3,4))
print("\n4.Array with zeros of size 3X4\n",arr1)
```

```
4.Array with zeros of size 3X4
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

(e) Create sequence of integers from 0 to 30 with step size 5

```
#create sequence of integers from 0 to 30 with step size 5
f = np.arange(0, 30, 5)
print ("\n5.A sequential array with steps of 5:\n", f)
```

```
5.A sequential array with steps of 5:
 [ 0  5 10 15 20 25]
```

(f) reshape an array 2x4 to 2x2x3

```
#reshape an array 2x4 to 2x2x3
newarr = arr1.reshape(2, 2, 3)
print ("\n6.Original array:\n", arr1)
print ("\nReshaped array:\n", newarr)
```

```
6.Original array:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

Reshaped array:
 [[[0. 0. 0.]
  [0. 0. 0.]]

 [[0. 0. 0.]
  [0. 0. 0.]]]
```

(g) Flatten an array

```
#flatten an array
arr = np.array([[1, 2, 3], [4, 5, 6]])
flarr = arr.flatten()
print ("\n7.Original array:\n", arr)
print ("\nFattened array:\n", flarr)
```

```
7.Original array:
 [[1 2 3]
 [4 5 6]]

Fattened array:
 [1 2 3 4 5 6]
```

(h) Slice array with 2 rows and 2 columns

```
#slice array with 2 rows and 2 columns
temp = arr[:2,::2]
print("\n8.Array with first two rows and alternate columns(0 and 2):\n",temp)
```

```
8.Array with first two rows and alternate columns(0 and 2):
 [[1 3]
 [4 6]]
```