ॐ श्री गणेशाय नमः

In [1]:
```
from time import time
import math
def time_random():
    return time() - float(str(time()).split('.')[0])

def gen_random_range(min, max):
    return int(time_random() * (max - min) + min)
```

## Question-1

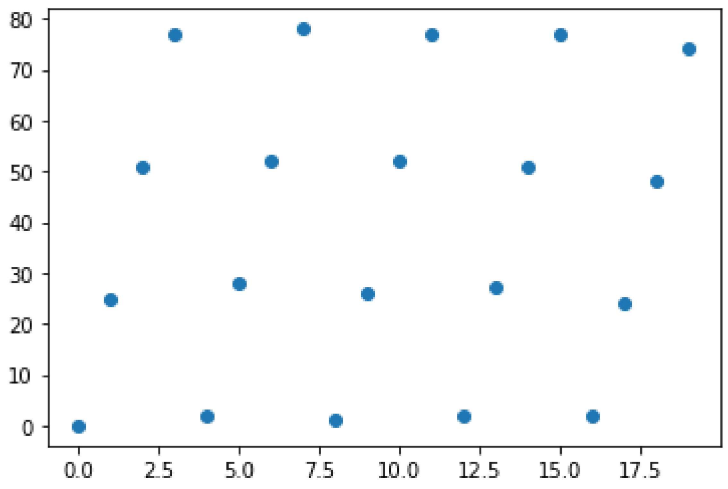
In [47]:
```
x=[]
y=[]
def randomNumber(Min, Max, N):
    def time_random():
        return time() - float(str(time()).split('.')[0])

    def gen_random_range(min, max):
        return int(time_random() * (max - min) + min)

    for i in range(N):
        for j in range(100000):
            time_random()
        x.append(i)
        y.append(gen_random_range(min,max))
    return f"x :{x} \ny:{y}"

min = int(input("Enter the first number:"))
max= int(input("Enter the second number:"))
N = int(input("Random number required: "))
print(randomNumber(min, max, N))
```

```
Enter the first number:0
Enter the second number:100
Random number required: 20
x :[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
y:[0, 25, 51, 77, 2, 28, 52, 78, 1, 26, 52, 77, 2, 27, 51, 77, 2, 24, 48, 74]
```

## Question-2

In [48]:
```
import matplotlib.pyplot as plt
plt.scatter(x, y)
```

Out[48]:
```
<matplotlib.collections.PathCollection at 0x1991d790970>
```



## Question-3

In [4]:
```
import math
from math import acos
for i in range(10):
    for j in range(int(math.exp(10))):
        time_random()
    pi = round(2 * acos(0.0), 2)
    print(pi)
```

```
3.14
3.14
3.14
3.14
3.14
3.14
3.14
3.14
3.14
3.14
```

In [ ]:

In [5]:
```
import numpy as np
import pandas as pd
```

In [6]:
```
data = pd.read_csv("text_data.csv", encoding= 'unicode_escape', index_col = [0])
```

In [7]:
```
#data = pd.read_clipboard(index_col=[0])
data
```

Out[7]:

| S.No. | Message_body | Label |
|---|---|---|
| 1 | UpgrdCentre Orange customer, you may now claim... | Spam |
| 2 | Loan for any purpose £500 - £75,000. Homeowner... | Spam |
| 3 | Congrats! Nokia 3650 video camera phone is you... | Spam |
| 4 | URGENT! Your Mobile number has been awarded wi... | Spam |
| 5 | Someone has contacted our dating service and e... | Spam |
| ... | ... | ... |
| 121 | 7 wonders in My WORLD 7th You 6th Ur style 5th... | Non-Spam |
| 122 | Try to do something dear. You read something f... | Non-Spam |
| 123 | Sun ah... Thk mayb can if dun have anythin on.... | Non-Spam |
| 124 | SYMPTOMS when U are in love: "1.U like listeni... | Non-Spam |
| 125 | Great. Have a safe trip. Dont panic surrender ... | Non-Spam |

125 rows × 2 columns

In [8]:
```
import nltk
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
#from gensim.utils import lemmatize
```

In [9]:
```
#pip install gensim
```

In [10]:
```
#nltk.download('stopwords') # comment out if already downloaded
nltk.download('punkt')     # comment out if already downloaded
```

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

Out[10]:
```
True
```

In [11]:
```
# convert to lower case
df = data.apply(lambda x: x.str.lower())
```

In [12]:
```
# replace special characters (preserving only space)
df = df.apply(lambda x: [re.sub('[^a-z0-9]', ' ', i) for i in x])
```

In [13]:
```
# tokenize columns
df = df.apply(lambda x:[word_tokenize(i) for i in x])
```

In [14]:
```
# remove stop words from token list in each column
df = df.apply(
    lambda x: [
                 [ w for w in tokenlist if w not in stopwords.words('english')]
                 for tokenlist in x])
```

In [15]:
```
# lemmatize columns
# the lemmatize method may fail during the first 3 to 4 iterations,
# so try running it several times
for attempt in range(1, 11):
    try:
        print(f'Lemmatize attempt: {attempt}')
        df = df.apply(
            lambda x: [ [  l.decode('utf-8').split('/', 1)[0]
                             for word in tokenlist for l in lemmatize(word) ]
                           for tokenlist in x])
        print(f'Attempt {attempt} success!')
        break
    except:
        pass
```

```
Lemmatize attempt: 1
Lemmatize attempt: 2
Lemmatize attempt: 3
Lemmatize attempt: 4
Lemmatize attempt: 5
Lemmatize attempt: 6
Lemmatize attempt: 7
Lemmatize attempt: 8
Lemmatize attempt: 9
Lemmatize attempt: 10
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 125 entries, 1 to 125
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Message_body  125 non-null    object
 1   Label         125 non-null    object
dtypes: object(2)
memory usage: 2.9+ KB
```

In [17]:
```
df['Message_body'][125][0]
```

Out[17]:
```
'great'
```

In [18]:
```
x=df['Message_body'].apply(lambda x: ", ".join(x))
y=df['Label'].apply(lambda x: " ".join(x))
```

In [19]:
```
x.head(-5)
```

Out[19]:
```
S.No.
1      upgrdcentre, orange, customer, may, claim, fre...
2      loan, purpose, 500, 75, 000, homeowners, tenan...
3      congrats, nokia, 3650, video, camera, phone, c...
4      urgent, mobile, number, awarded, 2000, prize, ...
5      someone, contacted, dating, service, entered, ...
                             ...                        
116                                           awake, oh
117                                      think, da, wil
118    piss, talking, someone, realise, u, point, rea...
119                  hospital, da, return, home, evening
120                              gettin, rdy, ship, comp
Name: Message_body, Length: 120, dtype: object
```

In [20]:
```
y.head()
```
Out[20]:
```
S.No.
1    spam
2    spam
3    spam
4    spam
5    spam
Name: Label, dtype: object
```

In [21]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [22]:
```
cv = CountVectorizer()
cv.fit(x)
cv_transform= cv.transform(x)
```

In [23]:
```
cv_transform
```
Out[23]:
```
<125x984 sparse matrix of type '<class 'numpy.int64'>'
	with 1685 stored elements in Compressed Sparse Row format>
```

In [24]:
```
cv_transform.toarray()
```
Out[24]:
```
array([[0, 1, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [25]:
```
pd.DataFrame(cv_transform.toarray(),
             columns=cv.get_feature_names_out()).head()
```
Out[25]:

| | 000 | 0207 | 021 | 03 | 07046744435 | 07123456789 | 07732584351 | 07742676969 | 0800 | 0800083940 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **1** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

5 rows × 984 columns

In [26]:
```
from sklearn.feature_extraction.text import TfidfTransformer
```

In [27]:
```
tf= TfidfTransformer()
tf_fit = tf.fit(cv_transform)
x_tf_transform = tf_fit.transform(cv_transform)
```

In [28]:
```
pd.DataFrame(x_tf_transform.toarray(),columns = cv.get_feature_names_out()).head()
```

Out[28]:

| | 000 | 0207 | 021 | 03 | 07046744435 | 07123456789 | 07732584351 | 07742676969 | 0800 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000000 | 0.254955 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **1** | 0.201213 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.22553 | |
| **2** | 0.000000 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **3** | 0.000000 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **4** | 0.000000 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |

5 rows × 984 columns

In [29]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [30]:
```
clf = MultinomialNB()
Model=clf.fit(x_tf_transform, y)
```

In [31]:
```
Model.predict(cv.transform(
    ["Wow didn't think it was that common. I take it all back ur not a freak! Unless u
```

Out[31]:
```
array(['non spam'], dtype='<U8')
```

In [32]:
```
Model.predict(cv.transform(
    ["You have lost 1 Millian $"]))
```

Out[32]:
```
array(['spam'], dtype='<U8')
```

In [33]:
```
Model.predict(cv.transform(
    ["Your salary is debited to your account"]))
```

Out[33]:
```
array(['spam'], dtype='<U8')
```

In [34]:
```
Model.predict(cv.transform(
    ["This is cat not dog"]))
```

Out[34]:
```
array(['spam'], dtype='<U8')
```

In [35]:
```
Model.predict(cv.transform(
    ["This model is not good"]))
```

Out[35]:
```
array(['non spam'], dtype='<U8')
```

In [36]:
```
Model.predict(cv.transform(
    ["Isq ne sathiya, mera Hala kya kar diya"]))
```

Out[36]:
```
array(['spam'], dtype='<U8')
```

In [37]:
```
Model.predict(cv.transform(
    ["Describe the issue linked to the documentation.It is becoming increasingly diffi
```

Out[37]:
```
array(['non spam'], dtype='<U8')
```

In [38]:
```
Model.predict(cv.transform(
    ["Isq ne sathiya, mera Hala kya kar diya"]))
```

Out[38]:
```
array(['spam'], dtype='<U8')
```

In [39]:
```
Model.predict(cv.transform(
    ["Deprecated: Read and write audio files in AIFF or AIFC format."]))
```

Out[39]:
```
array(['non spam'], dtype='<U8')
```

In [40]:
```
Model.predict(cv.transform(
    ["Command line option and argument parsing library"]))
```

Out[40]:
```
array(['spam'], dtype='<U8')
```

In [41]:
```
Model.predict(cv.transform(
    ["piss, talking, someone, realise, u, point,"]))
```

Out[41]:
```
array(['non spam'], dtype='<U8')
```

In [42]:
```
Model.predict(cv.transform(
    ["Dog,cat, bat",'Cat']))
```

Out[42]:
```
array(['spam', 'spam'], dtype='<U8')
```

In [43]:
```
import pickle
pickle.dump(cv, open('cv.pkl', 'wb'))
pickle.dump(Model, open('Model.pkl', 'wb'))
```

In [44]:
```
pwd()
```

Out[44]:
```
'E:\\Angel AI'
```

In [ ]:
```
from flask import Flask, render_template, request
import pickle

app = Flask(__name__, template_folder=r"E:\Angel AI\templates")
clf = pickle.load(open('Model.pkl', "rb"))
cv = pickle.load(open("cv.pkl", "rb"))


@app.route('/')
def Model():
    return render_template('Model.html')


@app.route('/Label', methods=['POST', 'GET'])
def Label():
    if request.method == 'POST':
        #HTML ->.py
        result = request.form['Data']
        result_pred = clf.predict(cv.transform([result]))
    #.py -> HTML
    return render_template("Label.html", result=result_pred)
```

```
if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
127.0.0.1 - - [11/Apr/2022 10:42:38] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Apr/2022 10:43:08] "POST /Label HTTP/1.1" 200 -
```

In [3]:
```
quit()
```