Data Toolkit Assignment

Question_1st:-Demonstrate three different methods for creating identical 2D arrays in NumPy. Provide the code for eachmethod and the final output after each method.

```
# Method 1: Using np.array():-You can create a 2D array directly by
passing a list of lists to the

import numpy as np

array1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Array created using np.array():\n", array1)

print("========================")

# Method 2: Using np.full():-This method creates a 2D array filled
with a specified value. You specify the shape and the value to fill
the array.

array2 = np.full((3, 3), [[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Array created using np.full():\n", array2)

print("=========================")

# Method 3: Using np.zeros() with slicing:-First, create a 2D array of
zeros with the desired shape, and then fill it with values.

array3 = np.zeros((3, 3), dtype=int)
array3[:] = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print("Array created using np.zeros() with slicing:\n", array3)
```

```
Array created using np.array():
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
========================
Array created using np.full():
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
=========================
Array created using np.zeros() with slicing:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Question_2nd:-❨ Using the Numpy function, generate an array of 100 evenly spaced numPers Petween 1 and 10 and Reshape that wD array into a 2D array

```
import numpy as np

# Step 1: Generate 100 evenly spaced numbers between 1 and 10
array_1d = np.linspace(1, 10, 100)

# Step 2: Reshape the 1D array into a 2D array (10x10)
array_2d = array_1d.reshape(10, 10)

print("1D Array of 100 evenly spaced numbers between 1 and 10:\n",
array_1d)
print("\nReshaped 2D Array (10x10):\n", array_2d)

1D Array of 100 evenly spaced numbers between 1 and 10:
 [ 1.          1.09090909  1.18181818  1.27272727  1.36363636
1.45454545
  1.54545455  1.63636364  1.72727273  1.81818182  1.90909091  2.
  2.09090909  2.18181818  2.27272727  2.36363636  2.45454545
2.54545455
  2.63636364  2.72727273  2.81818182  2.90909091  3.
3.09090909
  3.18181818  3.27272727  3.36363636  3.45454545  3.54545455
3.63636364
  3.72727273  3.81818182  3.90909091  4.          4.09090909
4.18181818
  4.27272727  4.36363636  4.45454545  4.54545455  4.63636364
4.72727273
  4.81818182  4.90909091  5.          5.09090909  5.18181818
5.27272727
  5.36363636  5.45454545  5.54545455  5.63636364  5.72727273
5.81818182
  5.90909091  6.          6.09090909  6.18181818  6.27272727
6.36363636
  6.45454545  6.54545455  6.63636364  6.72727273  6.81818182
6.90909091
  7.          7.09090909  7.18181818  7.27272727  7.36363636
7.45454545
  7.54545455  7.63636364  7.72727273  7.81818182  7.90909091  8.
  8.09090909  8.18181818  8.27272727  8.36363636  8.45454545
8.54545455
  8.63636364  8.72727273  8.81818182  8.90909091  9.
9.09090909
  9.18181818  9.27272727  9.36363636  9.45454545  9.54545455
9.63636364
  9.72727273  9.81818182  9.90909091 10.        ]

Reshaped 2D Array (10x10):
 [[ 1.          1.09090909  1.18181818  1.27272727  1.36363636
1.45454545
   1.54545455  1.63636364  1.72727273  1.81818182]
 [ 1.90909091  2.          2.09090909  2.18181818  2.27272727
2.36363636
   2.45454545  2.54545455  2.63636364  2.72727273]
 [ 2.81818182  2.90909091  3.          3.09090909  3.18181818
3.27272727
   3.36363636  3.45454545  3.54545455  3.63636364]
 [ 3.72727273  3.81818182  3.90909091  4.          4.09090909
4.18181818
   4.27272727  4.36363636  4.45454545  4.54545455]
 [ 4.63636364  4.72727273  4.81818182  4.90909091  5.
5.09090909
   5.18181818  5.27272727  5.36363636  5.45454545]
 [ 5.54545455  5.63636364  5.72727273  5.81818182  5.90909091  6.
   6.09090909  6.18181818  6.27272727  6.36363636]
 [ 6.45454545  6.54545455  6.63636364  6.72727273  6.81818182
6.90909091
   7.          7.09090909  7.18181818  7.27272727]
 [ 7.36363636  7.45454545  7.54545455  7.63636364  7.72727273
7.81818182
   7.90909091  8.          8.09090909  8.18181818]
 [ 8.27272727  8.36363636  8.45454545  8.54545455  8.63636364
8.72727273
   8.81818182  8.90909091  9.          9.09090909]
 [ 9.18181818  9.27272727  9.36363636  9.45454545  9.54545455
9.63636364
   9.72727273  9.81818182  9.90909091 10.        ]]
```

Question_3rd:- Explain the following terms:

The difference in nparray, npasarray and npasanyarrayX

The difference between Deep copy and shallow copyX

```
# 1st:- np.array():-np.array(): This function always creates a new array. If you pass an existing array to it, it will make a copy of the array

import numpy as np

list_data = [1, 2, 3]
array1 = np.array(list_data)
print("Using np.array():", array1)

print("==========================")

# 2nd:-npasarray():-This function converts the input to an array, but it does not create a copy if the input is already an array

array2 = np.asarray(array1)
print("Using np.asarray():", array2)

print("==========================")

# 3rd:-np.asanarray():- This function is similar to np.asarray(), but it preserves subclasses. For example, if the input is a matrix (a subclass of ndarray), np.asanyarray() will keep it as a matrix

matrix_data = np.matrix([[1, 2], [3, 4]])
array3 = np.asanyarray(matrix_data)
print("Using np.asanyarray():\n", array3)
```

```
Using np.array(): [1 2 3]
==========================
Using np.asarray(): [1 2 3]
==========================
Using np.asanyarray():
 [[1 2]
 [3 4]]
```

Question_4th:-Generate a 3x3 array with random floating-point numPers Petween 5 and 20 9hen, round each numPer inthe array to 2 decimal places

```
import numpy as np

# Step 1: Generate a 3x3 array with random floating-point numbers
between 5 and 20
random_array = np.random.uniform(5, 20, (3, 3))

# Step 2: Round each number in the array to 2 decimal places
rounded_array = np.round(random_array, 2)

print("3x3 Array with random floating-point numbers between 5 and 20:\
n", random_array)
print("\nRounded 3x3 Array to 2 decimal places:\n", rounded_array)
```

```
3x3 Array with random floating-point numbers between 5 and 20:
 [[ 6.64242178 15.00523942  5.48201415]
 [ 5.60472063 17.454048   17.30607245]
 [ 5.92602993 18.60470873  7.27726084]]

Rounded 3x3 Array to 2 decimal places:
 [[ 6.64 15.01  5.48]
 [ 5.6  17.45 17.31]
 [ 5.93 18.6   7.28]]
```

Question_5th:- Create a NumPy array with random integers Petween 1 and 10 of shape (5,6))) After creating the array perform the following operations:

# a)Extract all even integers from array.

# b)Extract all odd integers from array

```
import numpy as np

# Create a random integer array with sahpe 5 row and 6 col
array=np.random.randint(1,10,size=(5,6))
print(array)
print("==================================")

even_integers = array[array % 2 == 0]
print("Even integers:", even_integers)

print("===================================")

odd_integers = array[array % 2 != 0]
print("odd integers:", odd_integers)

[[7 3 1 9 9 9]
 [9 2 2 8 1 1]
 [3 9 8 1 1 1]
 [7 5 2 3 4 3]
 [5 9 1 4 6 2]]
==================================
Even integers: [2 2 8 8 2 4 4 6 2]
===================================
odd integers: [7 3 1 9 9 9 9 1 1 3 9 1 1 1 7 5 3 3 5 9 1]
```

Question_6th:- Create a D NumPy array of shape (3, 3, 3) containing random integers Petween 1 and 10 Perform the following operations:

# a) Find the indices of the maximum values along each depth level (third axis).

# b) Perform element wise multiplication of between both arrayX

```
import numpy as np

# Create a 3D NumPy array with random integers between 1 and 10
array = np.random.randint(1, 10, size=(3, 3, 3))
print("3D Array:\n", array)
print("====================")



# Find indices of the maximum values along the third axis
max_indices = np.argmax(array, axis=2)
print("Indices of maximum values along each depth level:\n",
max_indices)

print("Another 3D array")
```

```python
# Another 3D array
array2=np.random.randint(1,10,size=(3,3,3))
print(array2)

print(" multhiplication of both array")
# multhiplication of both array

array_multiplication=array1*array2
print(array_multiplication)
```

```
3D Array:
 [[[7 9 9]
  [3 5 2]
  [4 2 6]]

 [[3 7 9]
  [8 3 7]
  [9 3 1]]

 [[3 3 1]
  [2 7 7]
  [3 4 5]]]
====================
Indices of maximum values along each depth level:
 [[1 1 2]
 [2 0 0]
 [0 1 2]]
Another 3D array
[[[5 5 3]
  [9 5 9]
  [8 3 5]]

 [[8 3 7]
  [9 9 3]
  [2 3 3]]

 [[4 3 2]
  [9 9 8]
  [7 7 3]]]
 multhiplication of both array
[[[5 5 3]
  [0 5 9]
  [0 0 5]]

 [[8 3 7]
  [0 9 3]
  [0 0 3]]

 [[4 3 2]
```

```
  [0 9 8]
  [0 0 3]]]
```

Question_7th:-(Clean and transform the 'Phone' column in the sample dataset to remove non-numeric characters and convert it to a numeric data type) Also display the taPle attriPutes and data types of each column(

```
import pandas as pd

# Display the original DataFrame
df=pd.read_csv("People Data.csv")
df
```

```
     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male   
1        2  3d5AD30A4cD38ed         Jo    Rivers  Female   
2        3  810Ce0F276Badec     Sheryl    Lowery  Female   
3        4  BF2a889C00f0cE1    Whitney    Hooper    Male   
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female   
..     ...              ...        ...       ...     ...   
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female   
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female   
997    998  2adde51d8B8979E      Cathy  Mckinney  Female   
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male   
999   1000  8b756f6231DDC6e        Lee      Tran  Female   

                             Email                   Phone Date of 
birth  \
0              pwarner@example.org           857.139.8239    27-01-
2014   
1    fergusonkatherine@example.net                    NaN    26-07-
1931   
2              fhoward@example.org          (599)782-0605    25-11-
2013   
3            zjohnston@example.com                    NaN    17-11-
2012   
4                 elin@example.net     (390)417-1635x3010    15-04-
1923   
..                             ...                    ...           ..
.   
995         lyonsdaisy@example.net           021.775.2933    05-01-
```

```
1959
996        dariusbryan@example.com   001-149-710-7799x721    06-10-
2001
997         georgechan@example.org  +1-750-774-4128x33265    13-05-
1918
998            wanda04@example.net          (915)292-2254    31-08-
1971
999        deannablack@example.org     079.752.5424x67259    24-01-
1947

                           Job Title  Salary
0                  Probation officer   90000
1                             Dancer   80000
2                               Copy   50000
3           Counselling psychologist   65000
4                Biomedical engineer  100000
..                               ...     ...
995                Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                 Ambulance person  100000
999       Nurse, learning disability   90000

[1000 rows x 10 columns]
```

```
# Clean the 'Phone' column by removing non-numeric characters
df['Phone'] = df['Phone'].str.replace(r'\D', '', regex=True)
df
```

```
     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1        2  3d5AD30A4cD38ed         Jo    Rivers  Female
2        3  810Ce0F276Badec     Sheryl    Lowery  Female
3        4  BF2a889C00f0cE1    Whitney    Hooper    Male
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female
..     ...              ...        ...       ...     ...
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female
997    998  2adde51d8B8979E      Cathy  Mckinney  Female
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999   1000  8b756f6231DDC6e        Lee      Tran  Female

                             Email           Phone Date of birth  \
0              pwarner@example.org      8571398239    27-01-2014
1    fergusonkatherine@example.net             NaN    26-07-1931
2              fhoward@example.org      5997820605    25-11-2013
3            zjohnston@example.com             NaN    17-11-2012
4                 elin@example.net  39041716353010    15-04-1923
..                             ...             ...           ...
995         lyonsdaisy@example.net      0217752933    05-01-1959
```

```
996        dariusbryan@example.com  0011497107799721    06-10-2001
997         georgechan@example.org  1750774412833265    13-05-1918
998            wanda04@example.net        9152922254    31-08-1971
999        deannablack@example.org   079752542467259    24-01-1947

                           Job Title  Salary
0                  Probation officer   90000
1                             Dancer   80000
2                               Copy   50000
3           Counselling psychologist   65000
4                Biomedical engineer  100000
..                               ...     ...
995                Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                 Ambulance person  100000
999       Nurse, learning disability   90000

[1000 rows x 10 columns]
```

```
# Convert the 'Phone' column to a numeric data type
df['Phone'] = pd.to_numeric(df['Phone'])
df
```

```
     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1        2  3d5AD30A4cD38ed         Jo    Rivers  Female
2        3  810Ce0F276Badec     Sheryl    Lowery  Female
3        4  BF2a889C00f0cE1    Whitney    Hooper    Male
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female
..     ...              ...        ...       ...     ...
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female
997    998  2adde51d8B8979E      Cathy  Mckinney  Female
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999   1000  8b756f6231DDC6e        Lee      Tran  Female

                             Email         Phone Date of birth  \
0              pwarner@example.org  8.571398e+09    27-01-2014
1    fergusonkatherine@example.net           NaN    26-07-1931
2              fhoward@example.org  5.997821e+09    25-11-2013
3            zjohnston@example.com           NaN    17-11-2012
4                 elin@example.net  3.904172e+13    15-04-1923
..                             ...           ...           ...
995         lyonsdaisy@example.net  2.177529e+08    05-01-1959
996        dariusbryan@example.com  1.149711e+13    06-10-2001
997         georgechan@example.org  1.750774e+15    13-05-1918
998            wanda04@example.net  9.152922e+09    31-08-1971
999        deannablack@example.org  7.975254e+13    24-01-1947
```

```
                           Job Title  Salary
0                  Probation officer   90000
1                             Dancer   80000
2                               Copy   50000
3            Counselling psychologist   65000
4                 Biomedical engineer  100000
..                               ...     ...
995                Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                 Ambulance person  100000
999       Nurse, learning disability   90000

[1000 rows x 10 columns]

# Display table attributes and data types of each column
print("\nTable attributes and data types:")
print(df.info())


Table attributes and data types:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Index          1000 non-null   int64  
 1   User Id        1000 non-null   object 
 2   First Name     1000 non-null   object 
 3   Last Name      1000 non-null   object 
 4   Gender         1000 non-null   object 
 5   Email          1000 non-null   object 
 6   Phone          979 non-null    float64
 7   Date of birth  1000 non-null   object 
 8   Job Title      1000 non-null   object 
 9   Salary         1000 non-null   int64  
dtypes: float64(1), int64(2), object(7)
memory usage: 78.2+ KB
None
```

Question_8th:- Perform the following terms using people dataset:

## a) Read the 'dataYcsv' file using pandas, skipping the first 50 rows. ## b) Only read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and ‘Salary’ from the file. ## c) Display the first 10 rows of the filtered dataset. ## d) Extract the ‘Salary’' column as a Series and display its last 5 valuesX

```
# a) Read the 'dataYcsv' file using pandas, skipping the first 50
rows.

import pandas as pd

# Read the CSV file, skipping the first 50 rows
df1 = pd.read_csv("People Data.csv",skiprows=50)
df1

       50  afF3018e9cdd1dA     George    Mercer  Female  \
0      51  CccE5DAb6E288e5         Jo    Zavala    Male
1      52  DfBDc3621D4bcec     Joshua     Carey  Female
2      53  f55b0A249f5E44D     Rickey     Hobbs  Female
3      54  Ed71DcfaBFd0beE      Robyn    Reilly    Male
4      55  FDaFD0c3f5387EC  Christina    Conrad    Male
..    ...              ...        ...       ...     ...
945   996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
946   997  ECddaFEDdEc4FAB      Donna     Barry  Female
947   998  2adde51d8B8979E      Cathy  Mckinney  Female
948   999  Fb2FE369D1E171A   Jermaine    Phelps    Male
949  1000  8b756f6231DDC6e        Lee      Tran  Female

    douglascontreras@example.net    +1-326-669-0118x4341  11-09-
1941  \
0           pamela64@example.net  001-859-448-9935x54536  23-11-1992

1      dianashepherd@example.net    001-274-739-8470x814  07-01-1915

2      ingramtiffany@example.org        241.179.9509x498  01-07-1910

3     carriecrawford@example.org       207.797.8345x6177  27-07-1982

4     fuentesclaudia@example.net    001-599-042-7428x143  06-01-1998

..                           ...                     ...         ...

945       lyonsdaisy@example.net            021.775.2933  05-01-1959

946      dariusbryan@example.com    001-149-710-7799x721  06-10-2001

947       georgechan@example.org   +1-750-774-4128x33265  13-05-1918

948          wanda04@example.net           (915)292-2254  31-08-1971

949      deannablack@example.org      079.752.5424x67259  24-01-1947


             Human resources officer   70000
0                       Nurse, adult   80000
1                Seismic interpreter   70000
2                          Barrister   60000
```

```
3                Engineer, structural  100000
4                     Producer, radio   50000
..                                ...     ...
945                 Personnel officer   90000
946           Education administrator   50000
947   Commercial/residential surveyor   60000
948                  Ambulance person  100000
949        Nurse, learning disability   90000

[950 rows x 10 columns]

# b. Only read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and
‘Salary’ from the file.

df1=pd.read_csv("People Data.csv",usecols=["Last
Name","Gender","Email","Phone","Salary"])
df1

    Last Name  Gender                          Email
Phone  \
0     Mahoney    Male            pwarner@example.org
857.139.8239
1      Rivers  Female  fergusonkatherine@example.net
NaN
2      Lowery  Female            fhoward@example.org
(599)782-0605
3      Hooper    Male          zjohnston@example.com
NaN
4        Rice  Female               elin@example.net     (390)417-
1635x3010
..        ...     ...                            ...
...
995    Bryant  Female         lyonsdaisy@example.net
021.775.2933
996     Barry  Female        dariusbryan@example.com   001-149-710-
7799x721
997  Mckinney  Female         georgechan@example.org  +1-750-774-
4128x33265
998    Phelps    Male            wanda04@example.net
(915)292-2254
999      Tran  Female        deannablack@example.org
079.752.5424x67259

     Salary
0     90000
1     80000
2     50000
3     65000
4    100000
..      ...
```

```
995   90000
996   50000
997   60000
998  100000
999   90000

[1000 rows x 5 columns]
```

```python
# c) Display the first 10 rows of the filtered dataset.
df=pd.read_csv("People Data.csv")

df2=df[(df['Salary'] > 50000)]

df2.head()
```

```
   Index          User Id First Name Last Name  Gender  \
0      1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1      2  3d5AD30A4cD38ed         Jo    Rivers  Female
3      4  BF2a889C00f0cE1    Whitney    Hooper    Male
4      5  9afFEafAe1CBBB9    Lindsey      Rice  Female
6      7  efeb05c7Cc94EA3     Ernest   Hoffman    Male

                           Email                 Phone Date of birth  \
0            pwarner@example.org          857.139.8239    27-01-2014
1  fergusonkatherine@example.net                   NaN    26-07-1931
3          zjohnston@example.com                   NaN    17-11-2012
4              elin@example.net  (390)417-1635x3010    15-04-1923
6        jeffharvey@example.com   093.655.7480x7895    22-12-1984

                Job Title  Salary
0       Probation officer   90000
1                  Dancer   80000
3  Counselling psychologist   65000
4     Biomedical engineer  100000
6          Health visitor   60000
```

```python
# d. Extract the ‘Salary’' column as a Series and display its last 5
valuesX
import pandas as pd

# Read the CSV file
df = pd.read_csv("People Data.csv")

# Extract the 'Salary' column as a Series
salary_series = df['Salary']

# Display the last 5 values of the 'Salary' Series
print(salary_series.tail(5))
```

```
995     90000
996     50000
```

```
997      60000
998     100000
999      90000
Name: Salary, dtype: int64
```

Question_9th:- Filter and select rows from the People_Dataset, where the “Last Name' column contains the name 'Duke','Gender' column contains the word Female and ‘salary’ should Pe less than 85000

```
import pandas as pd

# Read the CSV file
df = pd.read_csv("People Data.csv")

# Filter rows based on the conditions
filtered_df = df[(df['Last Name']=='Duke') & (df['Gender']=='Female')
& (df['Salary'] < 80000)]

# Display the filtered DataFrame
filtered_df
```

```
     Index          User Id First Name Last Name  Gender  \
45      46  99A502C175C4EBd     Olivia      Duke  Female  
210    211  DF17975CC0a0373    Katrina      Duke  Female  
457    458  dcE1B7DE83c1076      Traci      Duke  Female  
729    730  c9b482D7aa3e682     Lonnie      Duke  Female  

                        Email                   Phone Date of birth  \
45        diana26@example.net  001-366-475-8607x04350    13-10-1934   
210       robin78@example.com            740.434.0212    21-09-1935   
457  perryhoffman@example.org     +1-903-596-0995x489    11-02-1997   
729   kevinkramer@example.net            982.692.6257    12-05-2015   

           Job Title  Salary  
45           Dentist   60000  
210  Producer, radio   50000  
457        Herbalist   50000  
729     Nurse, adult   70000  
```

Question_10th:-Create a 7*5. Dataframe in Pandas using a series generated from 35. random integers Petween 1 to 6)?

```
import pandas as pd
import numpy as np

# Create a 2D numpy array with random integers between 1 and 6, shape
7x5
array = np.random.randint(1, 6, size=(7, 5))
print("Array:\n", array)

# Create a DataFrame from the 2D numpy array
df = pd.DataFrame(array, columns=['Column1', 'Column2', 'Column3',
'Column4', 'Column5'])
print("\nDataFrame:\n", df)

# Flatten the DataFrame to a 1D array and then create a Series
flattened_series = pd.Series(df.values.flatten())
print("\nFlattened Series:\n", flattened_series)
```

```
Array:
 [[4 5 1 2 4]
 [1 3 3 2 4]
 [2 5 4 5 3]
 [2 4 2 4 2]
 [1 2 3 2 5]
 [1 4 5 3 4]
 [4 3 3 2 2]]

DataFrame:
    Column1  Column2  Column3  Column4  Column5
0        4        5        1        2        4
1        1        3        3        2        4
2        2        5        4        5        3
3        2        4        2        4        2
4        1        2        3        2        5
5        1        4        5        3        4
6        4        3        3        2        2

Flattened Series:
 0     4
1     5
2     1
3     2
4     4
5     1
6     3
```

```
7     3
8     2
9     4
10    2
11    5
12    4
13    5
14    3
15    2
16    4
17    2
18    4
19    2
20    1
21    2
22    3
23    2
24    5
25    1
26    4
27    5
28    3
29    4
30    4
31    3
32    3
33    2
34    2
dtype: int32
```

Question_11th:-Create two different Series, each of length 50, with the following criteria:

a) The first Series should contain random numbers ranging from 10 to 50.

b) The second Series should contain random numbers ranging from 100 to 1000.

c) Create a DataFrame by 'joining these Series by column, and, change the names of the columns to 'col1', 'col2',etc

```python
# a) The first Series should contain random numbers ranging from 10 to 50.

import pandas as pd
import numpy as np

# Create a Series with random integers between 10 and 50
random_series_1st = pd.Series(np.random.randint(10, 50, size=10))

# Display the Series
print("Random Series_1st:\n", random_series_1st)
```

```
Random Series_1st:
 0    32
1    30
2    33
3    37
4    47
5    40
6    18
7    20
8    11
9    37
dtype: int32
```

```
# b. The second Series should contain random numbers ranging from 100
to 1000.

import pandas as pd
import numpy as np

# Create a Series with random integers between 10 and 50
random_series_2nd = pd.Series(np.random.randint(100, 1000, size=10))

# Display the Series
print("Random Series_2nd:\n", random_series_2nd)

Random Series_2nd:
 0    778
1    259
2    348
3    773
4    636
5    661
6    526
7    303
8    164
9    996
dtype: int32

# Create a DataFrame by 'joining these Series by column, and, change
the names of the columns to 'col1', 'col2',etc

# Combine the Series into a DataFrame
df = pd.DataFrame({
    'col1': random_series_1st,
    'col2': random_series_2nd,
})



# Display the DataFrame
print("DataFrame:\n", df)

DataFrame:
    col1  col2
0    32   778
1    30   259
2    33   348
3    37   773
4    47   636
5    40   661
6    18   526
7    20   303
8    11   164
9    37   996
```

Question_12th:-g Perform the following operations using people data set:

a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.

b) Delete the rows containing any missing values.

d) Print the final output also.

```
# a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the
dataset.

# a) Delete the 'Email', 'Phone', and 'Date of birth' columns
df=pd.read_csv("People Data.csv")
df = df.drop(columns=['Email', 'Phone', 'Date of birth'])
df

     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1        2  3d5AD30A4cD38ed         Jo    Rivers  Female
2        3  810Ce0F276Badec     Sheryl    Lowery  Female
3        4  BF2a889C00f0cE1    Whitney    Hooper    Male
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female
..     ...              ...        ...       ...     ...
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female
997    998  2adde51d8B8979E      Cathy  Mckinney  Female
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999   1000  8b756f6231DDC6e        Lee      Tran  Female

                    Job Title  Salary
0           Probation officer   90000
1                      Dancer   80000
2                        Copy   50000
3     Counselling psychologist   65000
4         Biomedical engineer  100000
..                        ...     ...
995         Personnel officer   90000
996   Education administrator   50000
```

```
997  Commercial/residential surveyor   60000
998                 Ambulance person  100000
999       Nurse, learning disability   90000

[1000 rows x 7 columns]
```

```
# b) Delete the rows containing any missing values.

import pandas as pd

# Load the dataset
df = pd.read_csv("People Data.csv")


# Delete the rows containing any missing values
df = df.dropna()
df
```

```
     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male
2        3  810Ce0F276Badec     Sheryl    Lowery  Female
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female
5        6  aF75e6dDEBC5b66     Sherry  Caldwell    Male
6        7  efeb05c7Cc94EA3     Ernest   Hoffman    Male
..     ...              ...        ...       ...     ...
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female
997    998  2adde51d8B8979E      Cathy  Mckinney  Female
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999   1000  8b756f6231DDC6e        Lee      Tran  Female

                       Email                  Phone Date of birth  \
0        pwarner@example.org           857.139.8239    27-01-2014
2        fhoward@example.org          (599)782-0605    25-11-2013
4           elin@example.net     (390)417-1635x3010    15-04-1923
5       kaitlin13@example.net            8537800927    06-08-1917
6      jeffharvey@example.com     093.655.7480x7895    22-12-1984
..                       ...                    ...           ...
995   lyonsdaisy@example.net           021.775.2933    05-01-1959
996  dariusbryan@example.com   001-149-710-7799x721    06-10-2001
997   georgechan@example.org  +1-750-774-4128x33265    13-05-1918
998      wanda04@example.net          (915)292-2254    31-08-1971
999  deannablack@example.org     079.752.5424x67259    24-01-1947

                           Job Title  Salary
0                   Probation officer   90000
2                                Copy   50000
4                 Biomedical engineer  100000
5            Higher education lecturer   50000
6                      Health visitor   60000
```

```
..                                 ...     ...
995                  Personnel officer   90000
996            Education administrator   50000
997  Commercial/residential surveyor   60000
998                   Ambulance person  100000
999         Nurse, learning disability   90000

[979 rows x 10 columns]
```

```python
# d) Print the final output also.

import pandas as pd

# Load the dataset
df = pd.read_csv("People Data.csv")

# a) Delete the 'Email', 'Phone', and 'Date of birth' columns
df = df.drop(columns=['Email', 'Phone', 'Date of birth'])

# b) Delete the rows containing any missing values
df = df.dropna()

# d) Print the final output
df
```

```
     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1        2  3d5AD30A4cD38ed         Jo    Rivers  Female
2        3  810Ce0F276Badec     Sheryl    Lowery  Female
3        4  BF2a889C00f0cE1    Whitney    Hooper    Male
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female
..     ...              ...        ...       ...     ...
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female
997    998  2adde51d8B8979E      Cathy  Mckinney  Female
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999   1000  8b756f6231DDC6e        Lee      Tran  Female

                           Job Title  Salary
0                  Probation officer   90000
1                             Dancer   80000
2                               Copy   50000
3           Counselling psychologist   65000
4                Biomedical engineer  100000
..                               ...     ...
995                Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                 Ambulance person  100000
999       Nurse, learning disability   90000
```

```
[1000 rows x 7 columns]
```

Question_13th:-Create two NumPy arrays, x and y, each containing 100 random float values between 0 and 1. Perform the following tasks using Matplotlib and NumPy:

a) Create a scatter plot using x and y, setting the color of the points to red and the marker style to 'o'.

b) Add a horizontal line at y = 0.5 using a dashed line style and label it as 'y = 0.5'.

c) Add a vertical line at x = 0.5 using a dotted line style and label it as 'x = 0.5'.

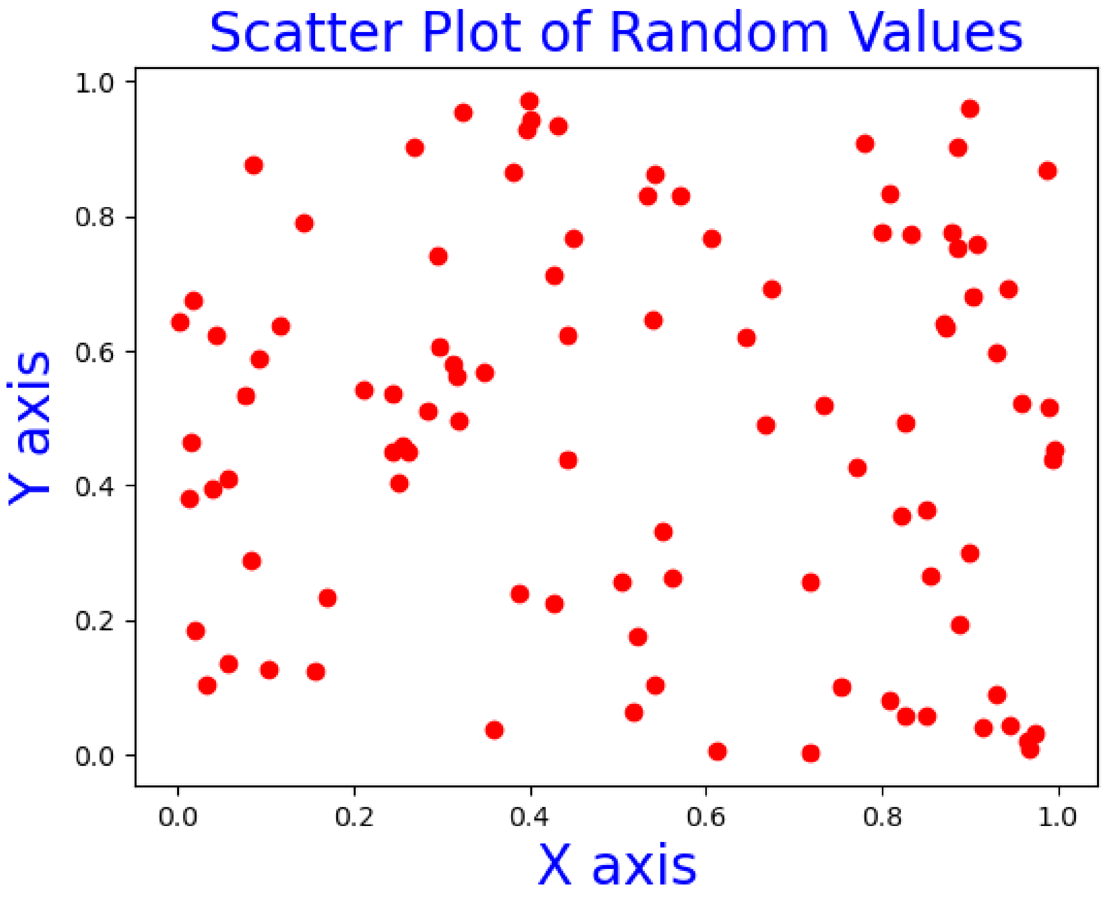
d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.

e) Set the title of the plot as 'Advanced Scatter Plot of Random Values'.

f) Display a legend for the scatter plot, the horizontal line, and the vertical line.

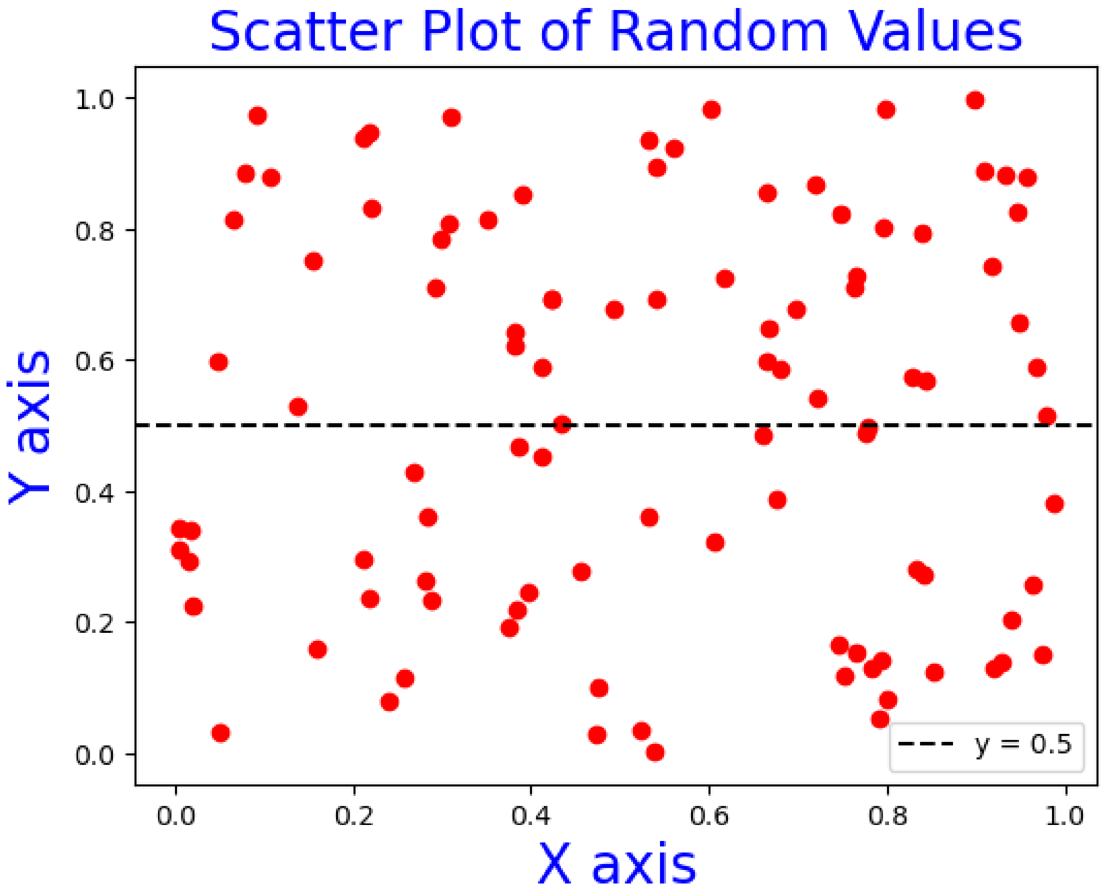
```
# a) Create a scatter plot using x and y, setting the color of the
points to red and the marker style to 'o'.
```

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

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y, color='red', marker='o')

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X axis', color='blue', fontsize=20)
plt.ylabel('Y axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

# Show the plot
plt.show()
```



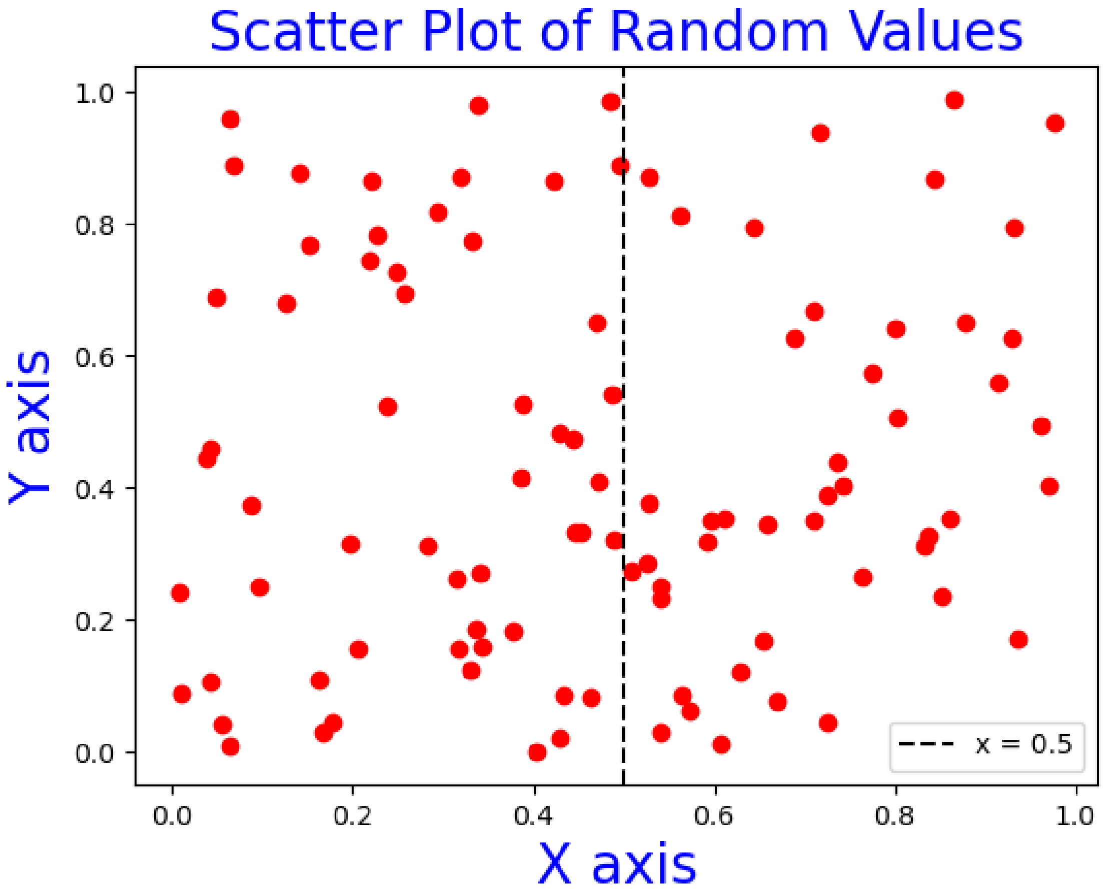
```python
# b) Add a horizontal line at y = 0.5 using a dashed line style and
label it as 'y = 0.5'.
```

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

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y, color='red', marker='o')

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X axis', color='blue', fontsize=20)
plt.ylabel('Y axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='black', linestyle='--', label='y = 0.5')

# Add a legend to show the label for the horizontal line
plt.legend()

# Show the plot
plt.show()
```



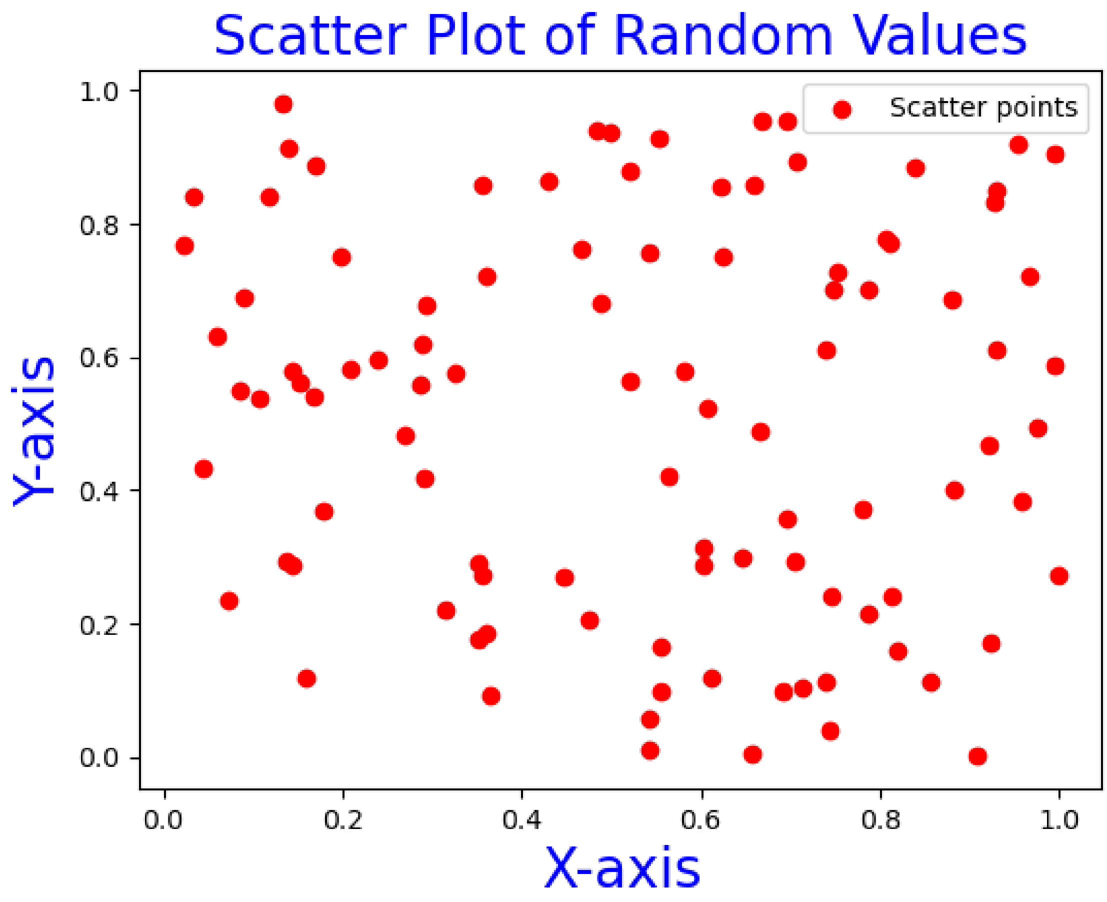
```
# c) Add a vertical line at x = 0.5 using a dotted line style and
label it as 'x = 0.5'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y, color='red', marker='o')

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X axis', color='blue', fontsize=20)
plt.ylabel('Y axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a horizontal line at y = 0.5 with dashed line style
plt.axvline(x=0.5, color='black', linestyle='--', label='x = 0.5')

# Add a legend to show the label for the horizontal line
```

```
plt.legend()

# Show the plot
plt.show()
```



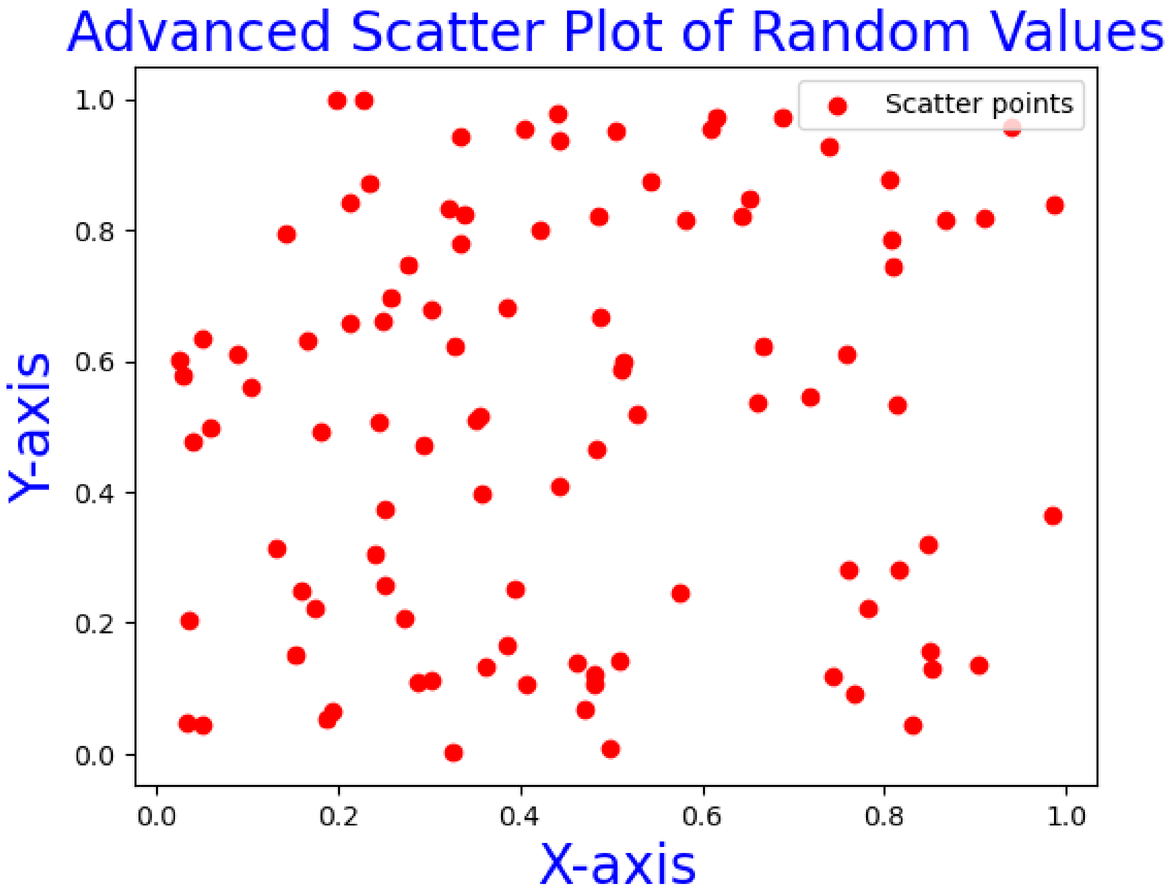
```
# d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot with a single label
plt.scatter(x, y, color='red', marker='o', label="Scatter points")

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X-axis', color='blue', fontsize=20)
plt.ylabel('Y-axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)
```

```
# Add a legend to show the label for the scatter points
plt.legend()

# Show the plot
plt.show()
```



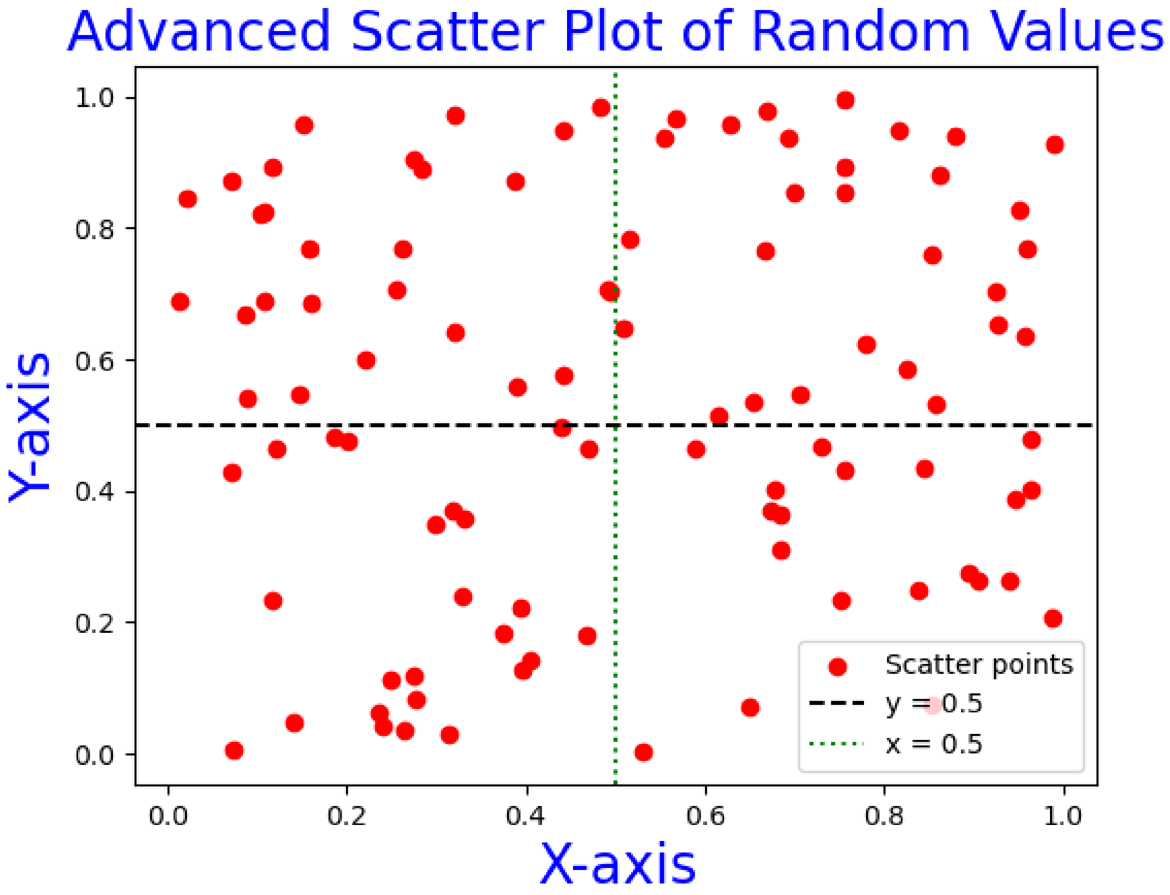
```
# e) Set the title of the plot as 'Advanced Scatter Plot of Random
Values'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot with a single label
plt.scatter(x, y, color='red', marker='o', label="Scatter points")

# Add labels and title for clarity with specified colors and sizes
```

```
plt.xlabel('X-axis', color='blue', fontsize=20)
plt.ylabel('Y-axis', color='blue', fontsize=20)

# Set the title of the plot
plt.title('Advanced Scatter Plot of Random Values', color='blue',
fontsize=20)

# Add a legend to show the label for the scatter points
plt.legend()

# Show the plot
plt.show()
```



```
# f) Display a legend for the scatter plot, the horizontal line, and
the vertical line.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
```

```python
y = np.random.rand(100)

# Create a scatter plot with a single label
plt.scatter(x, y, color='red', marker='o', label="Scatter points")

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X-axis', color='blue', fontsize=20)
plt.ylabel('Y-axis', color='blue', fontsize=20)

# Set the title of the plot
plt.title('Advanced Scatter Plot of Random Values', color='blue',
fontsize=20)

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='black', linestyle='--', label='y = 0.5')

# Add a vertical line at x = 0.5 with dotted line style
plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

# Add a legend to show the labels for the scatter plot, horizontal
line, and vertical line
plt.legend()

# Show the plot
plt.show()
```


Advanced Scatter Plot of Random Values
Y-axis
X-axis
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
● Scatter points
y = 0.5
x = 0.5

Question_14th:-Create a time-series dataset in a Pandas DataFrame with columns: 'Date', 'Temperature', 'Humidity' and Perform the following tasks using Matplotlib:

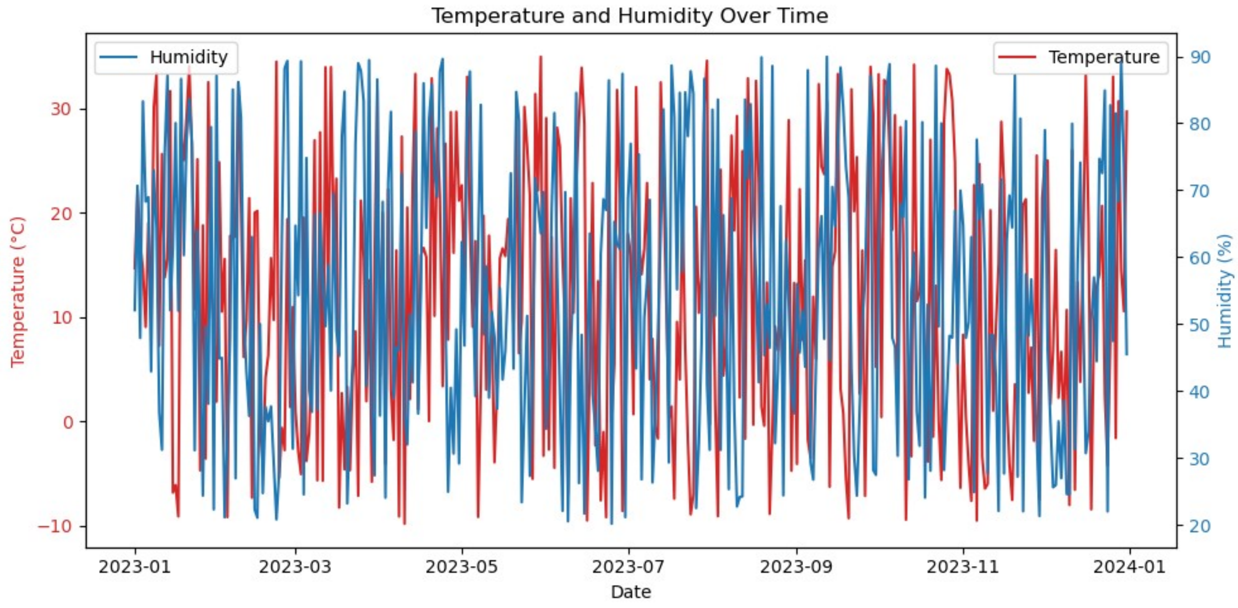
a) Plot the 'Temperature' and 'Humidity' on the same plot with different y-axes (left y-axis for 'Temperature' and right y-axis for 'Humidity').

b) Label the x-axis as 'Date'.

c) Set the title of the plot as 'Temperature and Humidity Over Time'.

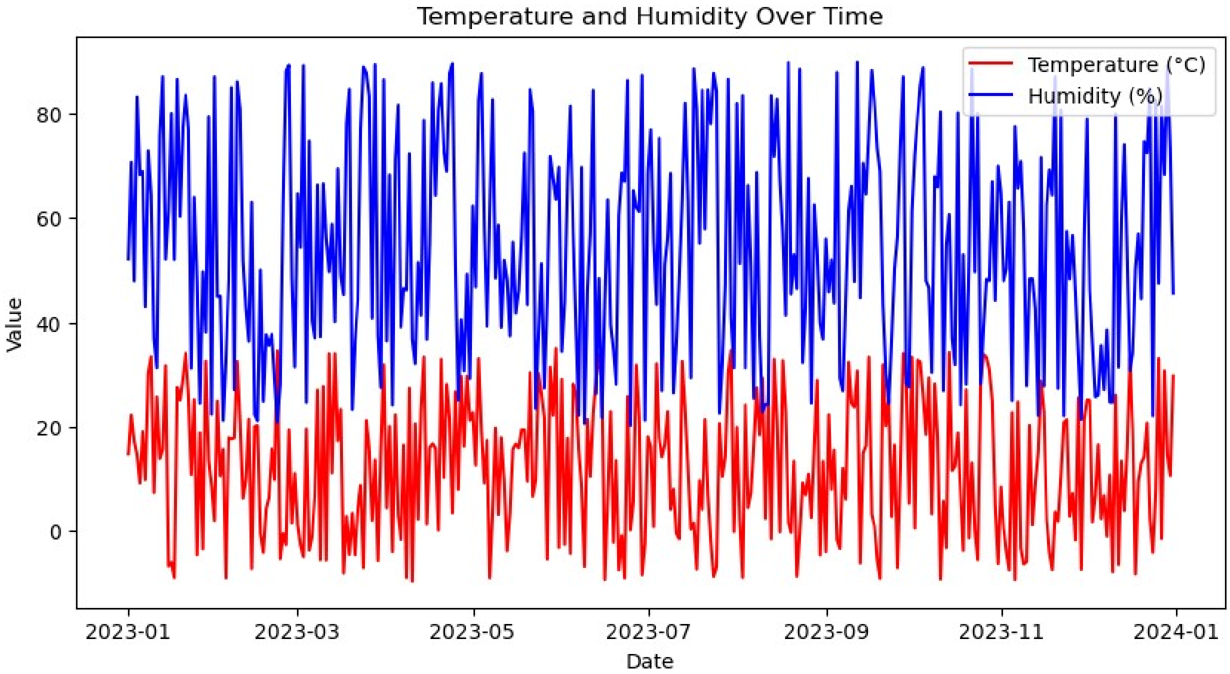
```python
# a) Plot the 'Temperature' and 'Humidity' on the same plot with
different y-axes (left y-axis for 'Temperature' and right y-axis for
'Humidity').

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a date range
date_range = pd.date_range(start='2023-01-01', end='2023-12-31',
freq='D')

# Generate random data for temperature and humidity
np.random.seed(0)  # for reproducibility
temperature = np.random.uniform(-10,35, size=len(date_range))
humidity = np.random.uniform(20,90, size=len(date_range))

# Create a DataFrame
df = pd.DataFrame({
    'Date': date_range,
    'Temperature': temperature,
    'Humidity': humidity
})
```

```
fig, ax1 = plt.subplots(figsize=(10, 5))

# Plot Temperature with the primary y-axis
ax1.plot(df['Date'], df['Temperature'], color='tab:red',
label='Temperature')
ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature (°C)', color='tab:red')
ax1.tick_params(axis='y', labelcolor='tab:red')
ax1.legend()

# Create a secondary y-axis for the Humidity
ax2 = ax1.twinx()
ax2.plot(df['Date'], df['Humidity'], color='tab:blue',
label='Humidity')
ax2.set_ylabel('Humidity (%)', color='tab:blue')
ax2.tick_params(axis='y', labelcolor='tab:blue')
ax2.legend()
# Add a title
plt.title('Temperature and Humidity Over Time')

# Show the plot
plt.tight_layout()
plt.show()
```



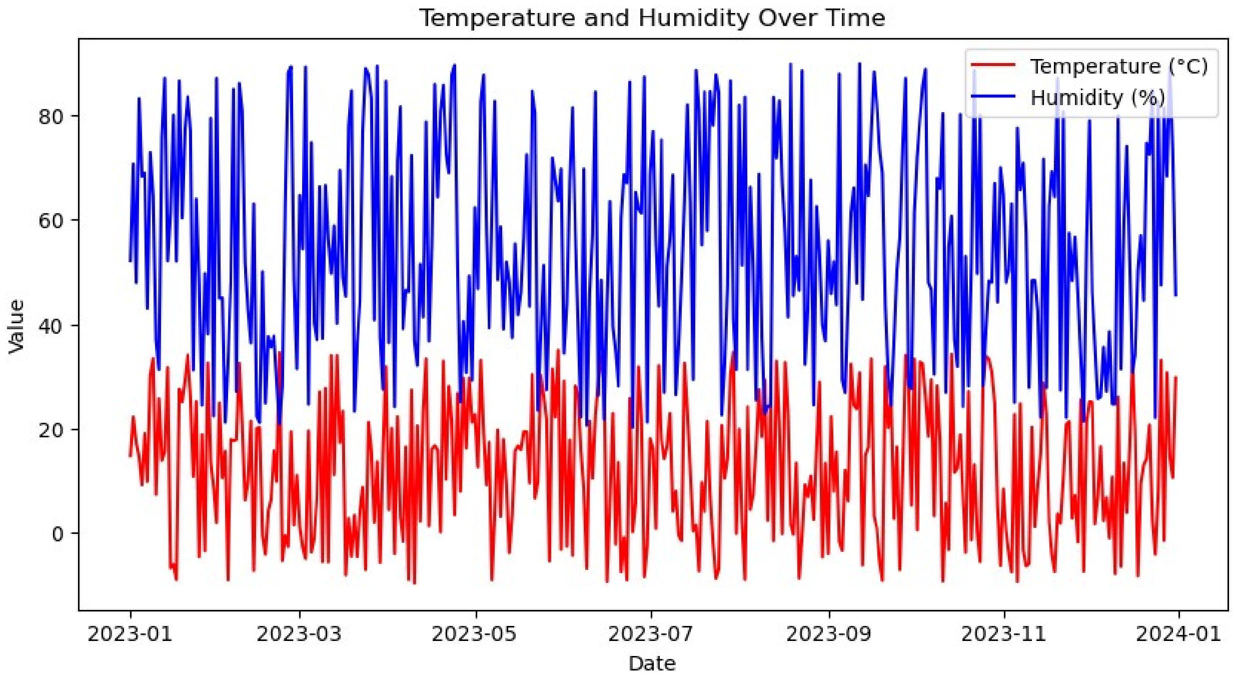
```
# b) Label the x-axis as 'Date'.


import matplotlib.pyplot as plt

# Plot Temperature and Humidity over time
```

```python
plt.figure(figsize=(10, 5))
plt.plot(df['Date'], df['Temperature'], label='Temperature (°C)',
color='red')
plt.plot(df['Date'], df['Humidity'], label='Humidity (%)',
color='blue')

# Label the axes
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Temperature and Humidity Over Time')

# Show legend
plt.legend()

# Show the plot
plt.show()
```



```python
# C. Set the title of the plot as 'Temperature and Humidity Over
Time'.

import matplotlib.pyplot as plt

# Plot Temperature and Humidity over time
plt.figure(figsize=(10, 5))
plt.plot(df['Date'], df['Temperature'], label='Temperature (°C)',
color='red')
plt.plot(df['Date'], df['Humidity'], label='Humidity (%)',
color='blue')
```

```
# Label the axes
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Temperature and Humidity Over Time')

# Show legend
plt.legend()

# Show the plot
plt.show()
```



Question_15th:- Create a NumPy array data containing 1000 samples from a normal distribution. Perform the following tasks using Matplotlib:

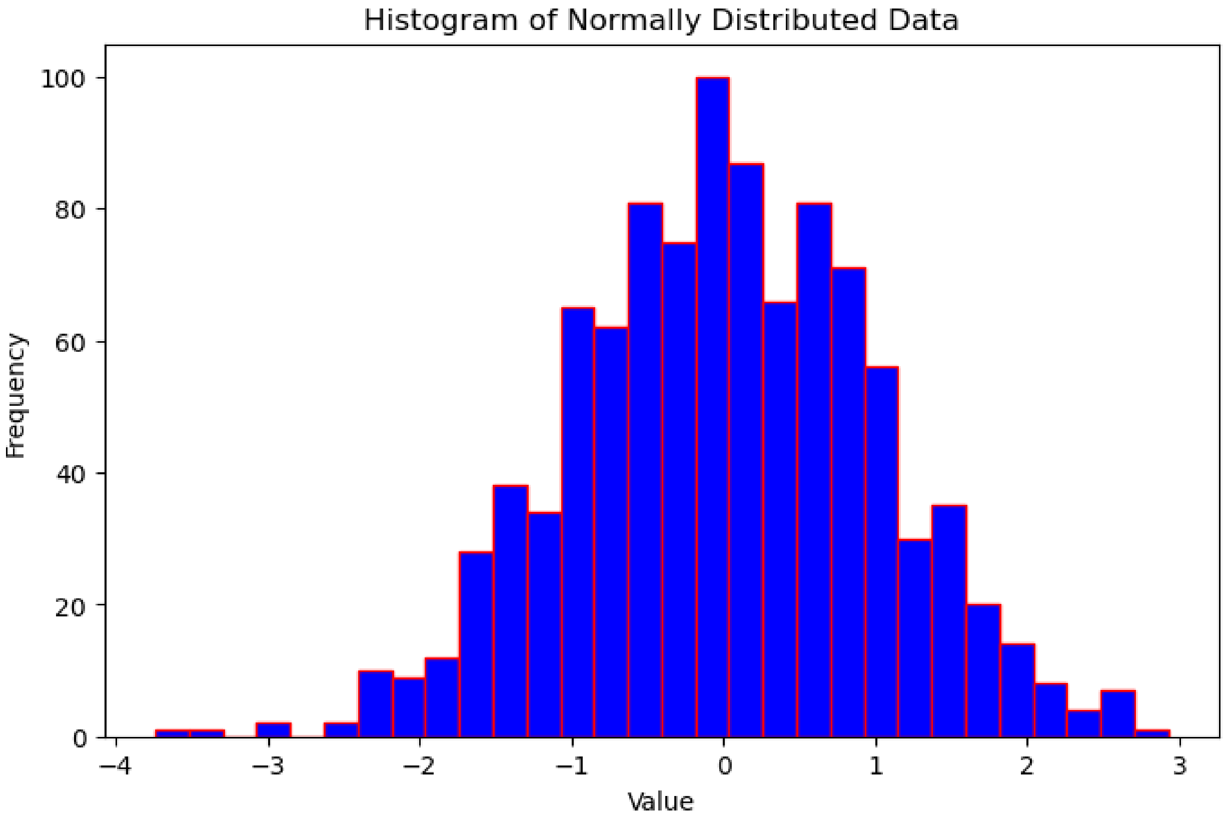
a) Plot a histogram of the data with 30 bins.

b) Overlay a line plot representing the normal distribution's probability density function (PDF).

c) Label the x-axis as 'Value' and the y-axis as 'Frequency/Probability'.

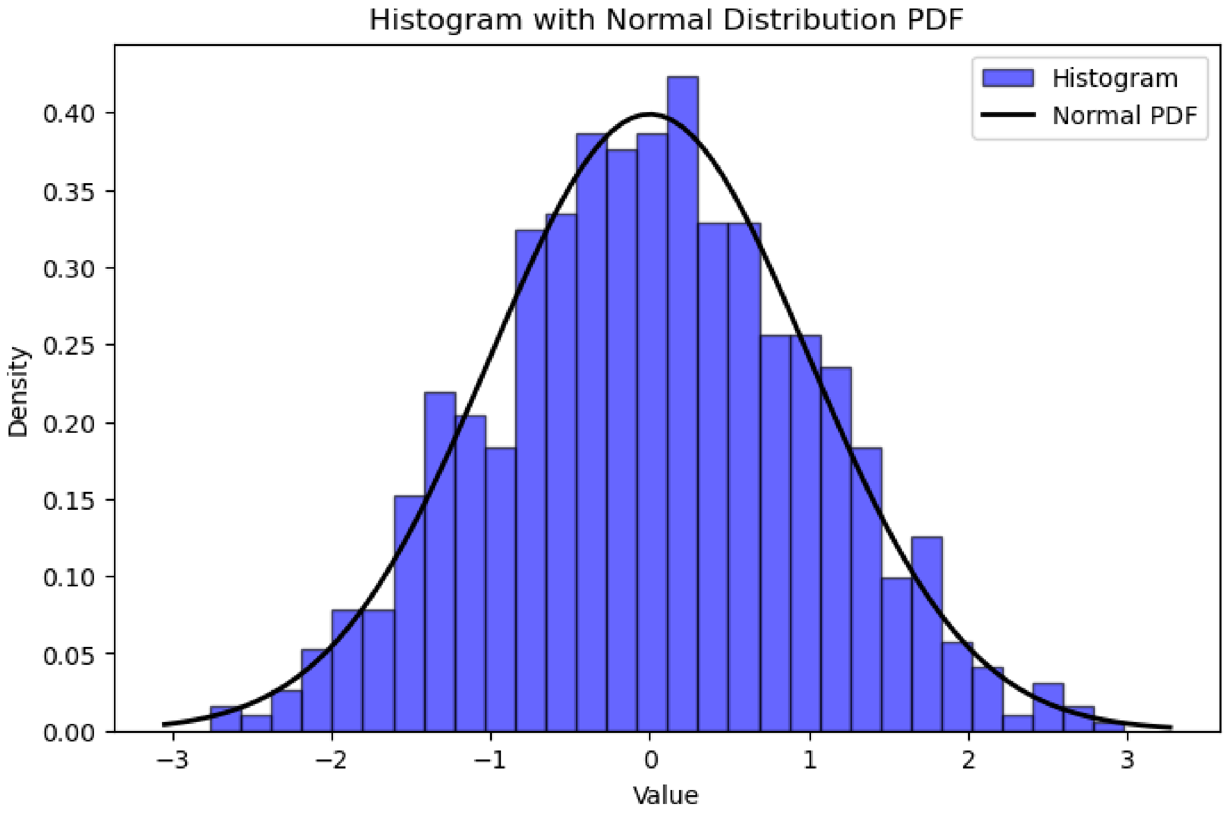
d) Set the title of the plot as 'Histogram with PDF Overlay'.

```python
# a) Plot a histogram of the data with 30 bins.

import numpy as np

# Generate 1000 samples from a normal distribution
data = np.random.normal(0,1, size=1000)

import matplotlib.pyplot as plt

# Plot the histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, color='blue', edgecolor='red')

# Label the axes
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Normally Distributed Data')

# Show the plot
plt.show()
```



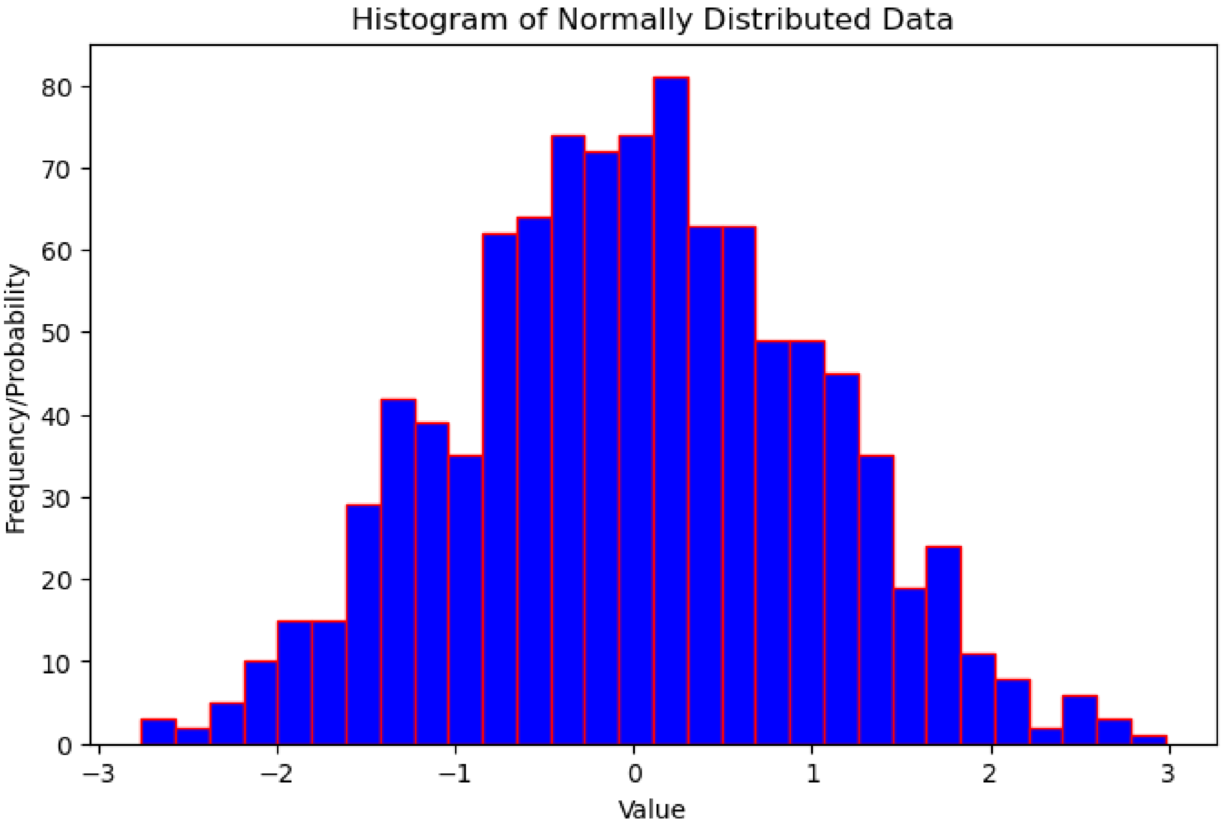
```
# b. Overlay a line plot representing the normal distribution's
probability density function (PDF).
import matplotlib.pyplot as plt
from scipy.stats import norm

# Create histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, density=True, color='blue', edgecolor='black',
alpha=0.6, label='Histogram')

# Overlay the PDF
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create 100 points between xmin and
xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF of the normal
distribution
plt.plot(x, p, 'k', linewidth=2, label='Normal PDF')  # Plot the PDF

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram with Normal Distribution PDF')

# Show legend
```

```
plt.legend()

# Display the plot
plt.show()
```

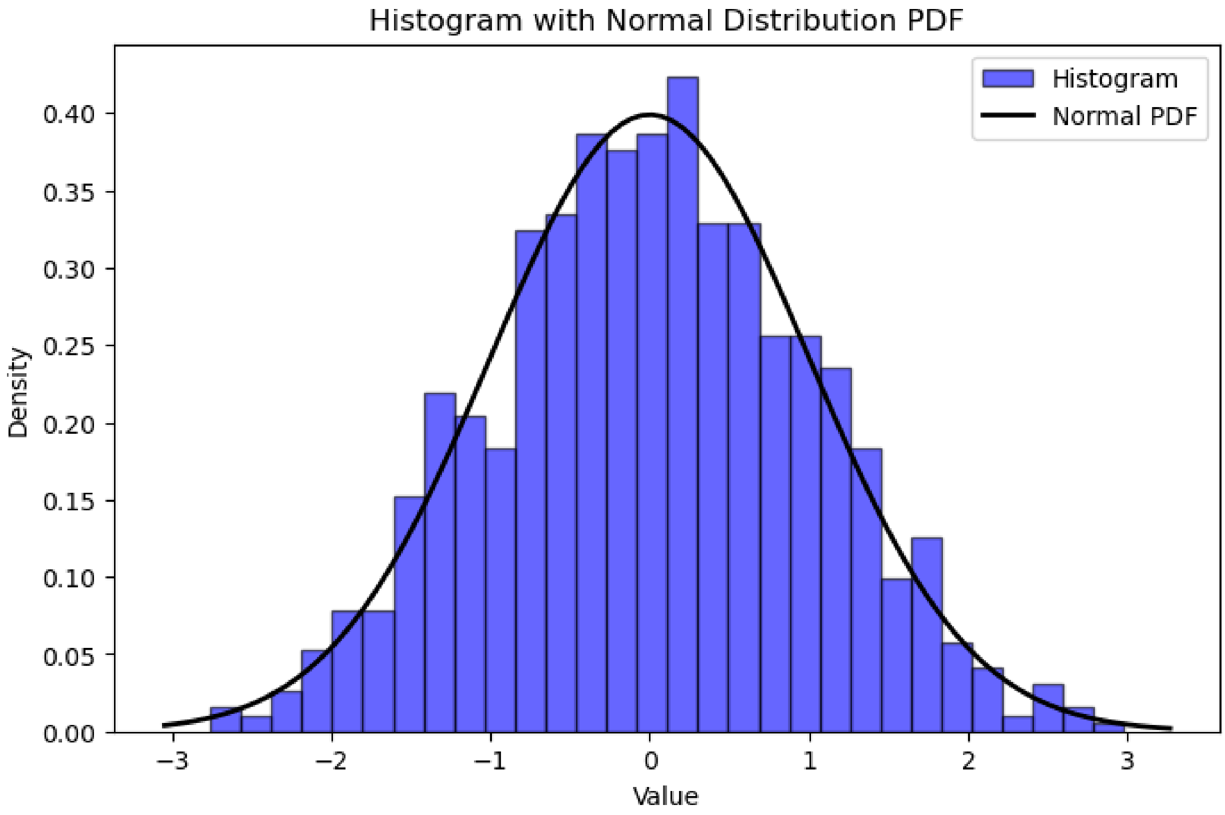


```
# c) Label the x-axis as 'Value' and the y-axis as
'Frequency/Probability'.

import numpy as np

# Generate 1000 samples from a normal distribution
data = np.random.normal(0,1, size=1000)

import matplotlib.pyplot as plt

# Plot the histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, color='blue', edgecolor='red')

# Label the axes
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram of Normally Distributed Data')
```

```
# Show the plot
plt.show()
```



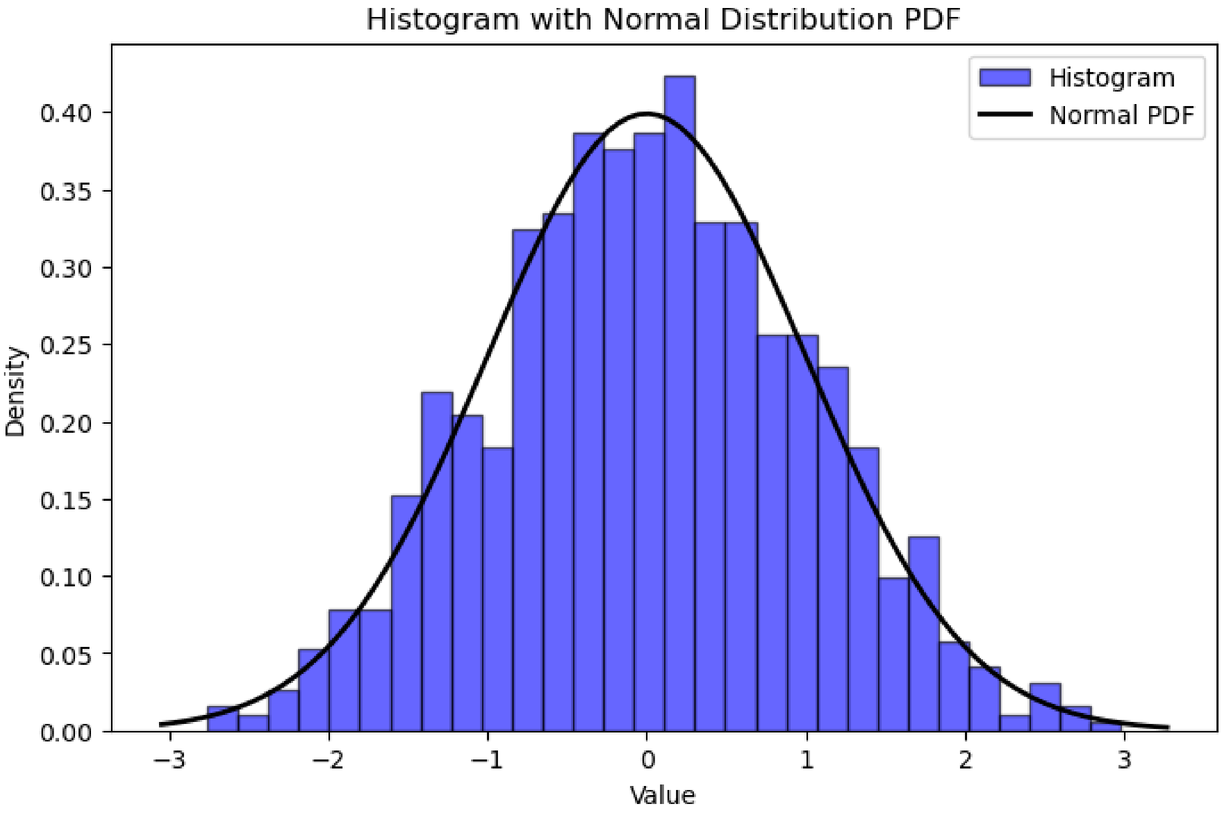
```
# d) Set the title of the plot as 'Histogram with PDF Overlay'.

import matplotlib.pyplot as plt
from scipy.stats import norm

# Create histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, density=True, color='blue', edgecolor='black', alpha=0.6, label='Histogram')

# Overlay the PDF
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create 100 points between xmin and xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF of the normal distribution
plt.plot(x, p, 'k', linewidth=2, label='Normal PDF')  # Plot the PDF

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram with Normal Distribution PDF')

# Show legend
plt.legend()

# Display the plot
plt.show()
```



Question_16th:- Set the title of the plot as 'Histogram with PDF Overlay'.

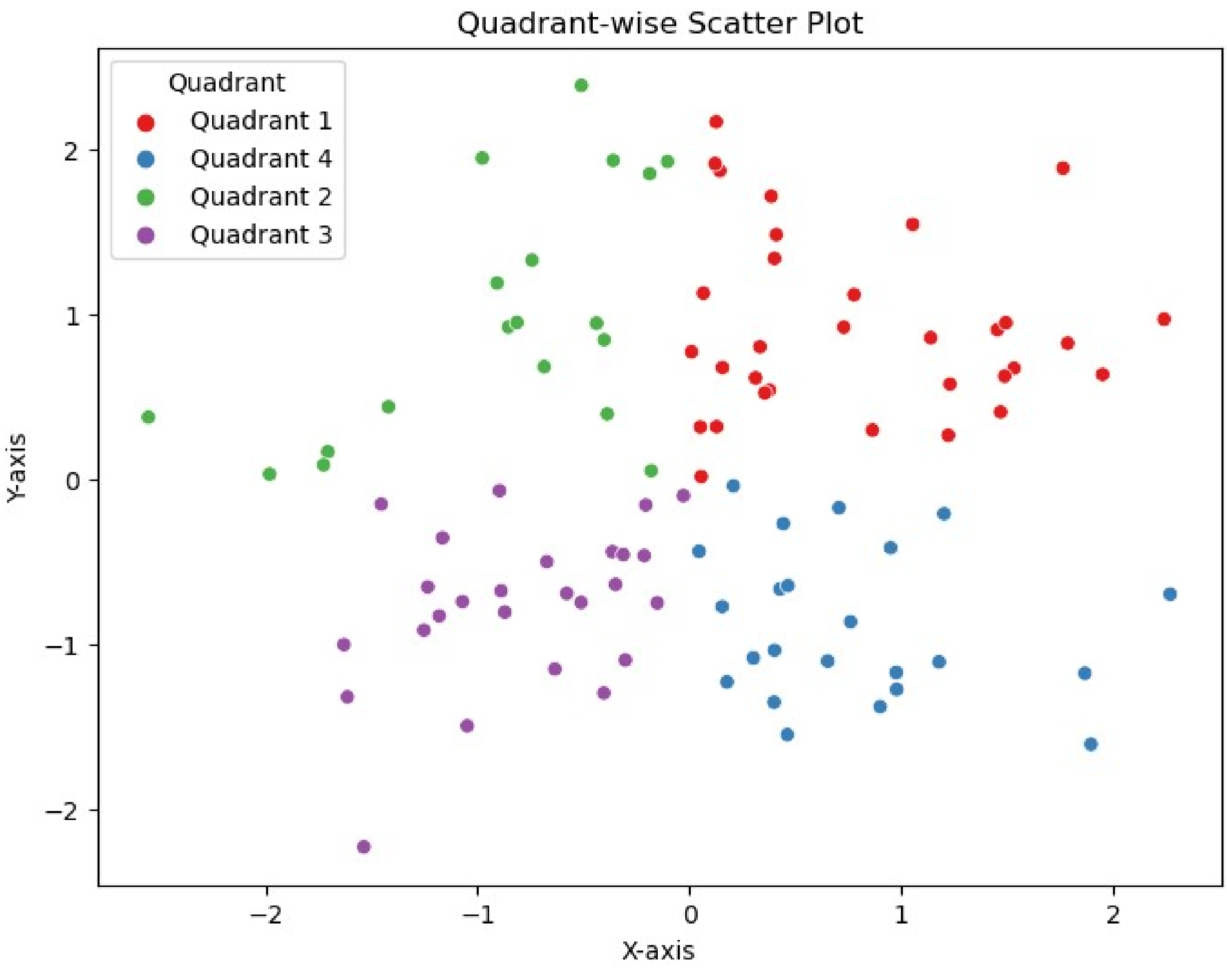
```
import matplotlib.pyplot as plt
from scipy.stats import norm

# Create histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, density=True, color='blue', edgecolor='black',
alpha=0.6, label='Histogram')

# Overlay the PDF
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
```

```
x = np.linspace(xmin, xmax, 100)  # Create 100 points between xmin and xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF of the normal distribution
plt.plot(x, p, 'k', linewidth=2, label='Normal PDF')  # Plot the PDF

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram with Normal Distribution PDF')

# Show legend
plt.legend()

# Display the plot
plt.show()
```



Question_17th:-Create a Seaborn scatter plot of two random arrays, color points based on their position relative to the origin (quadrants), add a legend, label the axes, and set the title as 'Quadrant-wise

```python
import numpy as np
import pandas as pd

# Generate random data for x and y
np.random.seed(0)  # For reproducibility
x = np.random.randn(100)
y = np.random.randn(100)

# Create a DataFrame
df = pd.DataFrame({'x': x, 'y': y})

# Define the quadrant based on the position relative to the origin
def determine_quadrant(row):
    if row['x'] >= 0 and row['y'] >= 0:
        return 'Quadrant 1'
    elif row['x'] < 0 and row['y'] >= 0:
        return 'Quadrant 2'
    elif row['x'] < 0 and row['y'] < 0:
        return 'Quadrant 3'
    else:
        return 'Quadrant 4'

df['Quadrant'] = df.apply(determine_quadrant, axis=1)

import seaborn as sns
import matplotlib.pyplot as plt

# Create the scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='x', y='y', hue='Quadrant', palette='Set1', 
legend='full')

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')

# Show the plot
plt.show()
```



Question_18th:- With Bokeh, plot a line chart of a sine wave function, add grid lines, label the axes, and set the title as 'Sine Wave Function.

```
from bokeh.plotting import figure, show, output_notebook
import numpy as np

# Prepare data
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# Create a Bokeh figure
p = figure(title="Sine Wave Function", x_axis_label='X-axis',
y_axis_label='Y-axis')

# Add a line renderer
p.line(x, y, line_width=2, color='blue', legend_label='Sine Wave')
```

```
# Add grid lines
p.grid.grid_line_color = 'gray'
p.grid.grid_line_alpha = 0.5

# Show the plot in a Jupyter notebook
output_notebook()
show(p)
```

```
"(function(root) {\n  function now() {\n    return new Date();\n  }\n\
n  const force = true;\n\n  if (typeof root._bokeh_onload_callbacks
=== \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded() {\n    const el =
document.getElementById(\"1053\");\n    if (el != null) {\n
el.textContent = \"BokehJS is loading...\";\n    }\n    if (root.Bokeh
!== undefined) {\n      if (el != null) {\n        el.textContent =
\"BokehJS \" + root.Bokeh.version + \" successfully loaded.\";\
n      }\n    } else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(display_loaded, 100)\n    }\n  }\n\n  function
run_callbacks() {\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-2.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n          for (let i = 0; i <
inline_js.length; i++) {\n      inline_js[i].call(root, root.Bokeh);\n
}\nif (force === true) {\n        display_loaded();\n      }} else if
(Date.now() < root._bokeh_timeout) {\n      setTimeout(run_inline_js,
100);\n    } else if (!root._bokeh_failed_load) {\n
console.log(\"Bokeh: BokehJS failed to load within specified
timeout.\");\n      root._bokeh_failed_load = true;\n    } else if
(force !== true) {\n      const cell = $
(document.getElementById(\"1053\")).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""
```

Question_19th:-Using Bokeh, generate a bar chart of randomly generated categorical data, color bars based on their values, add hover tooltips to display exact values, label the axes, and set the title as 'Random Categorical Bar Chart.

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool
import pandas as pd
import numpy as np

# Generate random categorical data
np.random.seed(0)  # For reproducibility
categories = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(1, 100, size=len(categories))

# Create a DataFrame
df = pd.DataFrame({'Category': categories, 'Value': values})

# Create a ColumnDataSource
source = ColumnDataSource(df)

# Create a Bokeh figure
p = figure(x_range=df['Category'], title="Random Categorical Bar
Chart",
           x_axis_label='Category', y_axis_label='Value',
           toolbar_location=None, tools='')

# Add bars with color based on values
p.vbar(x='Category', top='Value', width=0.5, source=source,
       legend_field='Category', color='blue', line_color='white')

# Add hover tooltips
hover = HoverTool()
hover.tooltips = [("Category", "@Category"), ("Value", "@Value")]
p.add_tools(hover)

# Customize grid lines and axis ticks
p.grid.grid_line_color = 'gray'
p.grid.grid_line_alpha = 0.5

# Show the plot in a Jupyter notebook
output_notebook()
show(p)
```

```
"(function(root) {\n  function now() {\n    return new Date();\n  }\n\
n  const force = true;\n\n  if (typeof root._bokeh_onload_callbacks
=== \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded() {\n    const el =
document.getElementById(\"1153\");\n    if (el != null) {\n
el.textContent = \"BokehJS is loading...\";\n    }\n    if (root.Bokeh
!== undefined) {\n      if (el != null) {\n        el.textContent =
\"BokehJS \" + root.Bokeh.version + \" successfully loaded.\";\
n      }\n    } else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(display_loaded, 100)\n    }\n  }\n\n  function
run_callbacks() {\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-2.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-2.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n          for (let i = 0; i <
inline_js.length; i++) {\n      inline_js[i].call(root, root.Bokeh);\n
}\nif (force === true) {\n        display_loaded();\n      }} else if
(Date.now() < root._bokeh_timeout) {\n      setTimeout(run_inline_js,
100);\n    } else if (!root._bokeh_failed_load) {\n
console.log(\"Bokeh: BokehJS failed to load within specified
timeout.\");\n      root._bokeh_failed_load = true;\n    } else if
(force !== true) {\n      const cell = $
(document.getElementById(\"1153\")).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""
```

Question_20:-Using Plotly, create a basic line plot of a randomly generated dataset, label the axes, and set the title as'Simple Line Plot.

```python
import plotly.graph_objects as go
import numpy as np

# Generate random data
np.random.seed(0)  # For reproducibility
x = np.linspace(0, 10, 100)
```

```
y = np.random.randn(100)

# Create a line plot
fig = go.Figure()

# Add a line trace
fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Random Data'))

# Update the layout with titles and axis labels
fig.update_layout(
    title='Simple Line Plot',
    xaxis_title='X-axis',
    yaxis_title='Y-axis'
)

# Show the plot
fig.show()
```

```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"mode":"lines","name":"Random Data","type":"scatter","x":
[0,0.10101010101010101,0.20202020202020202,0.30303030303030304,0.40404
040404040403,0.5050505050505051,0.6060606060606061,0.7070707070707071,
0.8080808080808081,0.9090909090909091,1.0101010101010102,1.11111111111
11112,1.2121212121212122,1.3131313131313131,1.4141414141414141,1.51515
15151515151,1.6161616161616161,1.7171717171717171,1.8181818181818181,1
.9191919191919191,2.0202020202020203,2.121212121212121,2.2222222222222
223,2.323232323232323,2.4242424242424243,2.525252525252525,2.626262626
2626263,2.727272727272727,2.8282828282828283,2.929292929292929,3.03030
30303030303,3.131313131313131,3.2323232323232323,3.3333333333333335,3.
4343434343434343,3.5353535353535355,3.6363636363636362,3.7373737373737
375,3.8383838383838382,3.9393939393939394,4.040404040404041,4.14141414
1414141,4.242424242424242,4.343434343434343,4.444444444444445,4.545454
545454545,4.646464646464646,4.747474747474747,4.848484848484849,4.9494
949494949495,5.05050505050505,5.151515151515151,5.252525252525253,5.3535
35353535354,5.454545454545454,5.555555555555555,5.656565656565657,5.75
7575757575758,5.858585858585858,5.959595959595959,6.0606060606060606,6
.161616161616162,6.262626262626262,6.363636363636363,6.464646464646464
5,6.565656565656566,6.666666666666667,6.767676767676767,6.868686868686
8685,6.96969696969697,7.070707070707071,7.171717171717171,7.2727272727
272725,7.373737373737374,7.474747474747475,7.575757575757575,7.6767676
767676765,7.777777777777778,7.878787878787879,7.979797979797979,8.0808
08080808081,8.181818181818182,8.282828282828282,8.383838383838384,8.48
4848484848484,8.585858585858587,8.686868686868687,8.787878787878787,8.
88888888888889,8.98989898989899,9.09090909090909,9.191919191919192,9.2
92929292929292,9.393939393939394,9.494949494949495,9.595959595959595,9
.696969696969697,9.797979797979798,9.8989898989899,10],"y":
[1.764052345967664,0.4001572083672233,0.9787379841057392,2.24089319920
1458,1.8675579901499675,-0.977277879876411,0.9500884175255894,-
0.1513572082976979,-
0.10321885179355784,0.41059850193837233,0.144043571160878,1.4542735069
```

```
62975,0.7610377251469934,0.12167501649282841,0.44386323274542566,0.333
67432737426683,1.4940790731576061,-
0.20515826376580087,0.31306770165090136,-0.8540957393017248,-
2.5529898158340787,0.6536185954403606,0.8644361988595057,-
0.742165020406419,2.2697546239876076,-
1.4543656745987648,4.575851730144607e-2,-
0.1871838500258336,1.5327792143584575,1.469358769900285,0.154947425696
9163,0.37816251960217356,-0.8877857476301128,-1.980796468223927,-
0.3479121493261526,0.15634896910398005,1.2302906807277207,1.2023798487
844113,-0.3873268174079523,-0.30230275057533557,-1.0485529650670926,-
1.4200179371789752,-1.7062701906250126,1.9507753952317897,-
0.5096521817516535,-0.4380743016111864,-
1.2527953600499262,0.7774903558319101,-1.6138978475579515,-
0.2127402802139687,-0.8954665611936756,0.386902497859262,-
0.510805137568873,-1.180632184122412,-2.8182228338654868e-
2,0.42833187053041766,6.651722238316789e-2,0.3024718977397814,-
0.6343220936809636,-0.3627411659871381,-0.672460447775951,-
0.3595531615405413,-0.813146282044454,-
1.7262826023316769,0.17742614225375283,-0.4017809362082619,-
1.6301983469660446,0.4627822555257742,-
0.9072983643832422,5.194539579613895e-
2,0.7290905621775369,0.12898291075741067,1.1394006845433007,-
1.2348258203536526,0.402341641177549,-0.6848100909403132,-
0.8707971491818818,-0.5788496647644155,-
0.31155253212737266,5.616534222974544e-2,-
1.1651498407833565,0.9008264869541871,0.46566243973045984,-
1.5362436862772237,1.4882521937955997,1.8958891760305832,1.17877957115
96507,-0.17992483581235091,-1.0707526215105425,1.0544517269311366,-
0.40317694697317963,1.2224450703824274,0.2082749780768603,0.9766390364
837128,0.3563663971744019,0.7065731681919482,1.0500020720820478e-
2,1.7858704939058352,0.12691209270361992,0.40198936344470165]}],"layou
t":{"template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
```

```
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
```

```
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Simple Line Plot"},"xaxis":{"title":{"text":"X-
axis"}},"yaxis":{"title":{"text":"Y-axis"}}}}
```

Question_21th:-Using Plotly, create an interactive pie chart of randomly generated data, add labels and percentages, set the title as 'Interactive Pie Chart'.

```python
import plotly.graph_objects as go
import numpy as np

# Generate random data
np.random.seed(0)  # For reproducibility
categories = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(10, 100, size=len(categories))

# Create the pie chart
fig = go.Figure(data=[go.Pie(
    labels=categories,
    values=values,
    textinfo='label+percent',  # Show labels and percentages
    hoverinfo='label+value+percent',  # Show additional info on hover
    hole=0.3  # Create a donut chart (set to 0 for a standard pie
chart)
)])

# Update layout with a title
fig.update_layout(title='Interactive Pie Chart')

# Show the plot
fig.show()
```

```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"hole":0.3,"hoverinfo":"label+value+percent","labels":
```

```
["A","B","C","D","E"],"textinfo":"label+percent","type":"pie","values"
:[54,57,74,77,77]}],"layout":{"template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
```

```
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Interactive Pie Chart"}}}
```