```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from scipy.stats import norm

import warnings
warnings.filterwarnings("ignore")
```

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

```
arr1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
arr1
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
arr2 = np.zeros((3, 3))
arr2
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
arr3 = np.ones((3, 3))
arr3
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

2. Using the NumPy function, generate an array of 100 evenly spaced numbers between 1 and 10 and Reashape that 1D array into a 2D array.

```
arr_1d = np.linspace(1, 10, 100)
arr_1d
```

```
array([ 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.        ])

arr_1d.ndim

1

arr_2d = arr_1d.reshape(20, 5)
arr_2d

array([[ 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.
]])
```

```
arr_2d.ndim
```

```
2
```

3. Explain the following terms:

```
The difference in np.array, np.asarray and np.asanyarray.
The difference between deep copy and shallow copy.

np.array always creates a new array.
np.asarray avoids creating a new array if the input is already a NumPy
array with the same dtype.
np.asanyarray behaves like np.asarray but preserves ndarray
subclasses.
Shallow copy copies the structure but not the elements, leading to
shared references between the copy and the original.
Deep copy recursively copies everything, resulting in two independent
objects.
```

4. Generate a 3×3 array with random floating-point numbers between 5 and 20. Then, round each number in the array of 2 decimal places.

```
arr = np.random.uniform(5, 20, (3, 3))
arr
```

```
array([[ 8.61344908, 15.08013811, 13.07804197],
       [ 9.53159794, 19.55420627,  7.17927754],
       [16.46736266, 15.61893546, 12.73668265]])
```

```
arr.round(decimals = 2)
```

```
array([[ 8.61, 15.08, 13.08],
       [ 9.53, 19.55,  7.18],
       [16.47, 15.62, 12.74]])
```

5. Create a NumPy array with random integers between 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
```

```
arr1 = np.random.randint(1, 10, (5, 6))
arr1
```

```
array([[4, 9, 1, 8, 4, 4],
       [1, 2, 2, 6, 4, 2],
       [4, 1, 4, 5, 4, 3],
       [5, 5, 1, 6, 6, 9],
       [1, 8, 4, 4, 6, 1]])
```

```
arr_even = arr1[arr1%2 == 0]
arr_even
```

```
array([4, 8, 4, 4, 2, 2, 6, 4, 2, 4, 4, 4, 6, 6, 8, 4, 4, 6])
```

```
arr_odd = arr1[arr1%2 != 0]
arr_odd
```

```
array([9, 1, 1, 1, 5, 3, 5, 5, 1, 9, 1, 1])
```

6. Create a 3D NumPy array of shape (3, 3, 3) containing random integers between 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 array.
```

```
arr_3d_1 = np.random.randint(1, 10, (3, 3, 3))
arr_3d_1
```

```
array([[[4, 4, 8],
        [9, 8, 1],
        [4, 6, 7]],

       [[9, 4, 9],
        [9, 5, 2],
        [1, 8, 5]],

       [[5, 9, 8],
        [6, 4, 6],
        [8, 1, 4]]])

np.argmax(arr_3d_1, axis = 2)

array([[2, 0, 2],
       [0, 0, 1],
       [1, 0, 0]], dtype=int64)

arr_3d_2 = np.random.randint(1, 10, (3, 3, 3))
arr_3d_2

array([[[5, 8, 9],
        [3, 5, 3],
        [8, 2, 8]],

       [[4, 2, 1],
        [8, 9, 4],
        [6, 4, 2]],

       [[4, 3, 2],
        [4, 1, 3],
        [1, 8, 1]]])

arr_3d_1 * arr_3d_2

array([[[20, 32, 72],
        [27, 40,  3],
        [32, 12, 56]],

       [[36,  8,  9],
        [72, 45,  8],
        [ 6, 32, 10]],

       [[20, 27, 16],
        [24,  4, 18],
        [ 8,  8,  4]]])
```

7. 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 table attributes and data types of each column.

8. Perform the following tasks using people dataset:

```
a) Read the 'data.csv' 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
values
```

```
people = pd.read_csv("E:\Data Analytics Study Material\Python\
Assignments\Assignment Data Toolkit\People Data.csv")
```

```
people.iloc[50:-1]
```

```
     Index          User Id First Name Last Name  Gender  \
50      51  CccE5DAb6E288e5         Jo    Zavala    Male
51      52  DfBDc3621D4bcec     Joshua     Carey  Female
52      53  f55b0A249f5E44D     Rickey     Hobbs  Female
53      54  Ed71DcfaBFd0beE      Robyn    Reilly    Male
54      55  FDaFD0c3f5387EC  Christina    Conrad    Male
..     ...              ...        ...       ...     ...
994    995  E54d5DDEeE6569E    Beverly      Ball    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

                         Email                    Phone Date of birth
\
50         pamela64@example.net  001-859-448-9935x54536    23-11-1992

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

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

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

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

..                          ...                     ...           ...

994  charlenehuerta@example.com        573-943-0389x380    01-07-1995

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


                            Job Title  Salary
50                       Nurse, adult   80000
51                 Seismic interpreter   70000
52                           Barrister   60000
53                Engineer, structural  100000
54                     Producer, radio   50000
..                                 ...     ...
994          Publishing rights manager   85000
995                  Personnel officer   90000
996            Education administrator   50000
997  Commercial/residential surveyor   60000
998                   Ambulance person  100000

[949 rows x 10 columns]
```

```
people[['Last Name', 'Gender','Email', 'Phone', 'Salary']].head(10)
```

```
  Last Name  Gender                             Email                Phone
Salary
0   Mahoney    Male            pwarner@example.org        857.139.8239
90000
1    Rivers  Female  fergusonkatherine@example.net                 NaN
80000
2    Lowery  Female            fhoward@example.org       (599)782-0605
50000
3    Hooper    Male          zjohnston@example.com                 NaN
65000
4      Rice  Female              elin@example.net  (390)417-1635x3010
100000
5  Caldwell    Male          kaitlin13@example.net          8537800927
50000
6   Hoffman    Male         jeffharvey@example.com   093.655.7480x7895
60000
7  Andersen    Male           alicia33@example.org          4709522945
65000
8      Mays    Male             jake50@example.com        013.820.4758
50000
9  Mitchell    Male      lanechristina@example.net  (560)903-5068x4985
50000
```

```
people['Salary'].tail()
```

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

```
999     90000
Name: Salary, dtype: int64
```

9. 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 be less than 85000.

```
df = pd.read_csv("E:\Data Analytics Study Material\Python\Assignments\
Assignment Data Toolkit\People Data.csv")

df.head()
```

```
   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

                           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

                  Job Title  Salary
0         Probation officer   90000
1                    Dancer   80000
2                      Copy   50000
3  Counselling psychologist   65000
4       Biomedical engineer  100000
```

```
df[(df['Last Name'] == 'Duke') & (df['Gender'] == 'Female') &
(df['Salary'] < 85000)]
```

```
     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
```

10. Create a 7*5 Dataframe in Pandas using a series generated from 35 random integers between 1 to 6?

```
array = np.random.randint(1, 6, (7, 5))

array

array([[2, 2, 1, 5, 3],
       [3, 2, 4, 3, 2],
       [2, 2, 4, 5, 5],
       [5, 4, 1, 1, 5],
       [3, 3, 2, 1, 2],
       [4, 5, 4, 2, 4],
       [5, 2, 2, 5, 3]])

pd.DataFrame(array)
```

	0	1	2	3	4
0	2	2	1	5	3
1	3	2	4	3	2
2	2	2	4	5	5
3	5	4	1	1	5
4	3	3	2	1	2
5	4	5	4	2	4
6	5	2	2	5	3

11. 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.

arr1 = np.random.randint(10, 50, (1, 50))
arr1

array([[43, 23, 30, 21, 24, 45, 36, 28, 11, 18, 25, 24, 48, 41, 14,
12,
        42, 41, 44, 40, 42, 10, 22, 43, 21, 20, 44, 33, 20, 24, 33,
35,
        36, 42, 35, 14, 41, 16, 37, 49, 44, 38, 48, 42, 42, 26, 49,
18,
        23, 36]])
```

```
arr2 = np.random.randint(100, 1000, (1, 50))
arr2
```

```
array([[856, 727, 337, 444, 585, 778, 838, 704, 248, 886, 430, 443,
770,
        187, 542, 275, 412, 629, 192, 409, 988, 203, 782, 846, 332,
386,
        903, 142, 154, 778, 549, 397, 558, 169, 672, 889, 805, 716,
946,
        364, 936, 648, 680, 476, 220, 761, 452, 113, 510, 958]])
```

```
arr1 = arr1.flatten()
```

```
arr2 = arr2.flatten()
```

```
df1 = pd.Series(arr1)
df1
```

```
0     43
1     23
2     30
3     21
4     24
5     45
6     36
7     28
8     11
9     18
10    25
11    24
12    48
13    41
14    14
15    12
16    42
17    41
18    44
19    40
20    42
21    10
22    22
23    43
24    21
25    20
26    44
27    33
28    20
29    24
30    33
31    35
```

```
32    36
33    42
34    35
35    14
36    41
37    16
38    37
39    49
40    44
41    38
42    48
43    42
44    42
45    26
46    49
47    18
48    23
49    36
dtype: int32
```

```
df2 = pd.Series(arr2)
df2
```

```
0     856
1     727
2     337
3     444
4     585
5     778
6     838
7     704
8     248
9     886
10    430
11    443
12    770
13    187
14    542
15    275
16    412
17    629
18    192
19    409
20    988
21    203
22    782
23    846
24    332
25    386
26    903
```

```
27    142
28    154
29    778
30    549
31    397
32    558
33    169
34    672
35    889
36    805
37    716
38    946
39    364
40    936
41    648
42    680
43    476
44    220
45    761
46    452
47    113
48    510
49    958
dtype: int32
```

```
new_df = pd.concat([df1, df2], axis = 1)
new_df
```

```
     0    1
0   43  856
1   23  727
2   30  337
3   21  444
4   24  585
5   45  778
6   36  838
7   28  704
8   11  248
9   18  886
10  25  430
11  24  443
12  48  770
13  41  187
14  14  542
15  12  275
16  42  412
17  41  629
18  44  192
19  40  409
20  42  988
```

```
21  10  203
22  22  782
23  43  846
24  21  332
25  20  386
26  44  903
27  33  142
28  20  154
29  24  778
30  33  549
31  35  397
32  36  558
33  42  169
34  35  672
35  14  889
36  41  805
37  16  716
38  37  946
39  49  364
40  44  936
41  38  648
42  48  680
43  42  476
44  42  220
45  26  761
46  49  452
47  18  113
48  23  510
49  36  958
```

```
new_df['col1'] = new_df[0]
new_df['col2'] = new_df[1]
new_df
```

```
     0    1  col1  col2
0   43  856    43   856
1   23  727    23   727
2   30  337    30   337
3   21  444    21   444
4   24  585    24   585
5   45  778    45   778
6   36  838    36   838
7   28  704    28   704
8   11  248    11   248
9   18  886    18   886
10  25  430    25   430
11  24  443    24   443
12  48  770    48   770
13  41  187    41   187
14  14  542    14   542
```

```
15  12  275    12   275
16  42  412    42   412
17  41  629    41   629
18  44  192    44   192
19  40  409    40   409
20  42  988    42   988
21  10  203    10   203
22  22  782    22   782
23  43  846    43   846
24  21  332    21   332
25  20  386    20   386
26  44  903    44   903
27  33  142    33   142
28  20  154    20   154
29  24  778    24   778
30  33  549    33   549
31  35  397    35   397
32  36  558    36   558
33  42  169    42   169
34  35  672    35   672
35  14  889    14   889
36  41  805    41   805
37  16  716    16   716
38  37  946    37   946
39  49  364    49   364
40  44  936    44   936
41  38  648    38   648
42  48  680    48   680
43  42  476    42   476
44  42  220    42   220
45  26  761    26   761
46  49  452    49   452
47  18  113    18   113
48  23  510    23   510
49  36  958    36   958
```

12. 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.

people = pd.read_csv('E:\Data Analytics Study Material\Python\
Assignments\Assignment Data Toolkit\People Data.csv')
people
```

```
     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]
```

```
people.drop("Email", axis = 1, inplace = True)
people
```

```
     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

                     Phone Date of birth                         Job
Title  \
0              857.139.8239    27-01-2014                Probation
officer
1                       NaN    26-07-1931
Dancer
2             (599)782-0605    25-11-2013
Copy
3                       NaN    17-11-2012         Counselling
psychologist
4       (390)417-1635x3010    15-04-1923              Biomedical
engineer
..                     ...           ...
...
995            021.775.2933    05-01-1959                Personnel
officer
996   001-149-710-7799x721    06-10-2001          Education
administrator
997  +1-750-774-4128x33265    13-05-1918  Commercial/residential
surveyor
998           (915)292-2254    31-08-1971                 Ambulance
person
999     079.752.5424x67259    24-01-1947       Nurse, learning
disability

     Salary
0     90000
1     80000
2     50000
3     65000
4    100000
..      ...
995   90000
996   50000
```

```
997   60000
998  100000
999   90000

[1000 rows x 9 columns]
```

```
people.drop("Phone", axis = 1, inplace = True)
people
```

```
     Index          User Id First Name Last Name  Gender Date of birth
\
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male    27-01-2014

1        2  3d5AD30A4cD38ed          Jo    Rivers  Female    26-07-1931

2        3  810Ce0F276Badec     Sheryl    Lowery  Female    25-11-2013

3        4  BF2a889C00f0cE1    Whitney    Hooper    Male    17-11-2012

4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female    15-04-1923

..     ...              ...        ...       ...     ...           ...

995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female    05-01-1959

996    997  ECddaFEDdEc4FAB      Donna     Barry  Female    06-10-2001

997    998  2adde51d8B8979E      Cathy  Mckinney  Female    13-05-1918

998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male    31-08-1971

999   1000  8b756f6231DDC6e        Lee      Tran  Female    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 8 columns]
```

```
people.drop("Date of birth", axis = 1, inplace = True)
people
```

```
     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]
```

```
people.dropna(inplace = True)
```

```
people
```

```
     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]
```

13. 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.
```

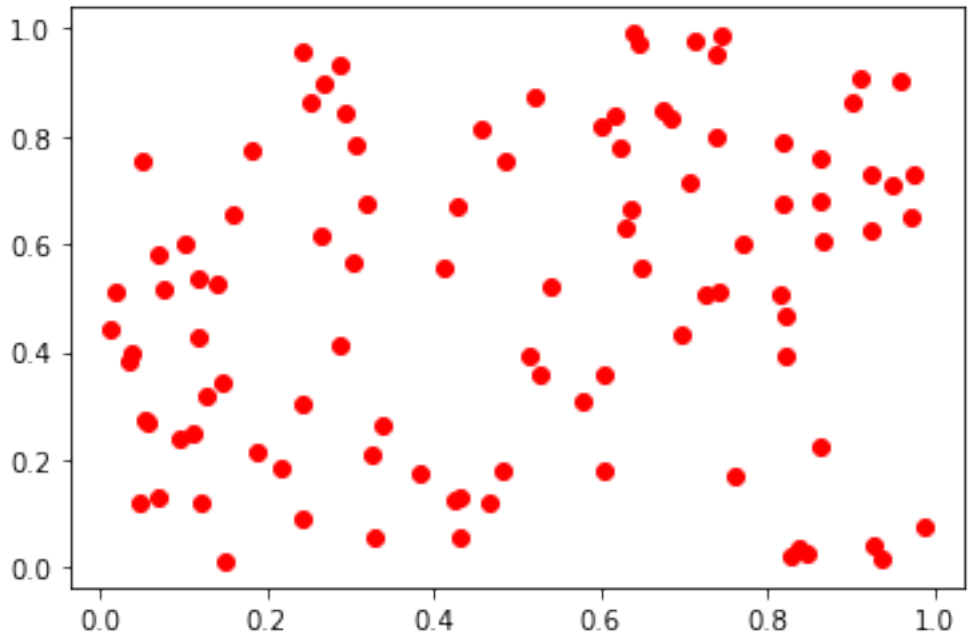
```
x = np.random.random_sample(100)
x
```

```
array([0.03450152, 0.05408247, 0.64406237, 0.32072827, 0.63869749,
       0.92586288, 0.95771343, 0.15979862, 0.1839215 , 0.04706235,
       0.32722445, 0.42678669, 0.41222936, 0.8648849 , 0.45609764,
       0.86261729, 0.2426288 , 0.10127123, 0.21900133, 0.67449038,
       0.32826305, 0.42344608, 0.51295816, 0.46760612, 0.76070377,
       0.3390745 , 0.11987355, 0.43046132, 0.94967136, 0.74216703,
       0.86129364, 0.81667519, 0.81418593, 0.98607433, 0.81676755,
       0.03763505, 0.7392881 , 0.82859246, 0.83746714, 0.38355111,
       0.62384336, 0.0708799 , 0.81985622, 0.63495521, 0.1222844 ,
       0.77027547, 0.91148628, 0.09612498, 0.62972866, 0.02058826,
       0.18881019, 0.30835008, 0.2864213 , 0.51977292, 0.26861803,
       0.05652694, 0.11290715, 0.84630606, 0.52863145, 0.70616786,
       0.57836339, 0.25224715, 0.96944158, 0.07145595, 0.73870773,
       0.64852614, 0.69746197, 0.15130887, 0.01448623, 0.60467567,
       0.29248046, 0.60475688, 0.61803589, 0.71306103, 0.12673121,
       0.05277577, 0.48344543, 0.14873467, 0.90072268, 0.4325124 ,
       0.14121521, 0.26525371, 0.1189574 , 0.86221721, 0.97274223,
       0.92212577, 0.93725187, 0.30260993, 0.74284396, 0.244323  ,
       0.24156568, 0.28884801, 0.53890857, 0.92234619, 0.48418119,
       0.68435702, 0.72473778, 0.59924204, 0.82028578, 0.07540606])
```

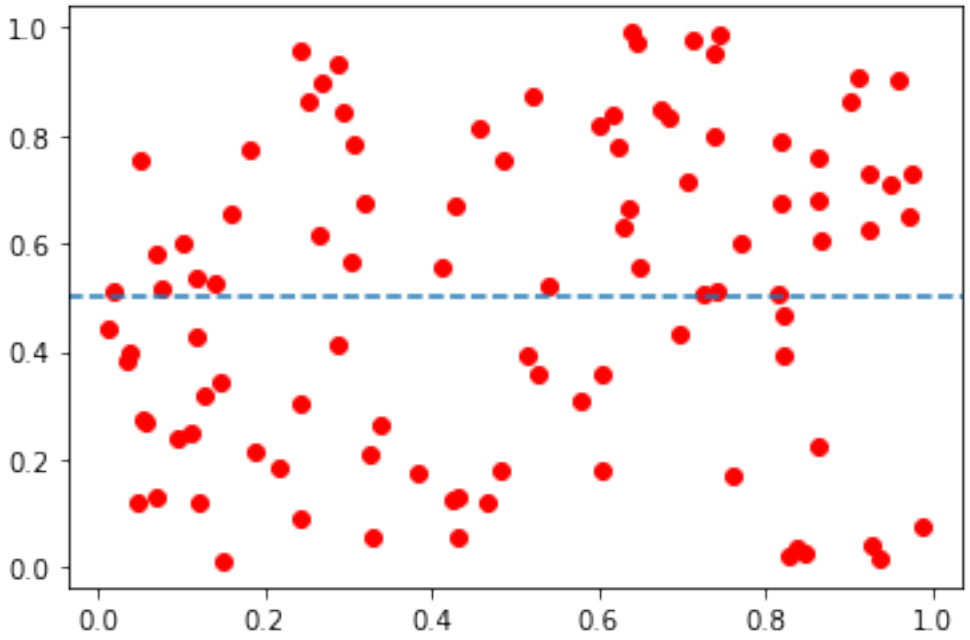
```
y = np.random.random_sample(100)
y
```

```
array([0.38065443, 0.27147584, 0.97168538, 0.67753138, 0.99134597,
       0.03860958, 0.90278228, 0.65762145, 0.77568858, 0.11901429,
       0.21124475, 0.67241243, 0.55651804, 0.60376411, 0.81226655,
       0.75979994, 0.30165445, 0.60002038, 0.18659062, 0.85123894,
       0.05595517, 0.12609642, 0.39098292, 0.12204675, 0.16941429,
       0.26407161, 0.53695996, 0.13076476, 0.71063101, 0.51218107,
       0.22172024, 0.67360629, 0.5088572 , 0.07287654, 0.79023158,
       0.39683224, 0.80049199, 0.02278747, 0.03592216, 0.17411531,
       0.77778906, 0.57936106, 0.39221906, 0.6645119 , 0.12034096,
       0.59992648, 0.90950361, 0.23854215, 0.63183886, 0.51279929,
       0.21228165, 0.78332508, 0.41119015, 0.87476286, 0.89839249,
       0.26683521, 0.24889186, 0.0250145 , 0.35731446, 0.71648513,
       0.30706087, 0.86182981, 0.65111836, 0.13061041, 0.95290847,
       0.55733997, 0.43077578, 0.01131144, 0.44467902, 0.35663423,
       0.84459072, 0.17964897, 0.83825578, 0.97645629, 0.3200896 ,
       0.7537971 , 0.18171784, 0.34482735, 0.86408614, 0.05485457,
       0.52765068, 0.61645071, 0.42586841, 0.67957169, 0.72850444,
       0.6240696 , 0.0144671 , 0.56435624, 0.98914045, 0.95764596,
       0.08966625, 0.93150595, 0.52051555, 0.73185431, 0.75648501,
       0.83384521, 0.50575361, 0.82120787, 0.46629911, 0.51837666])
```

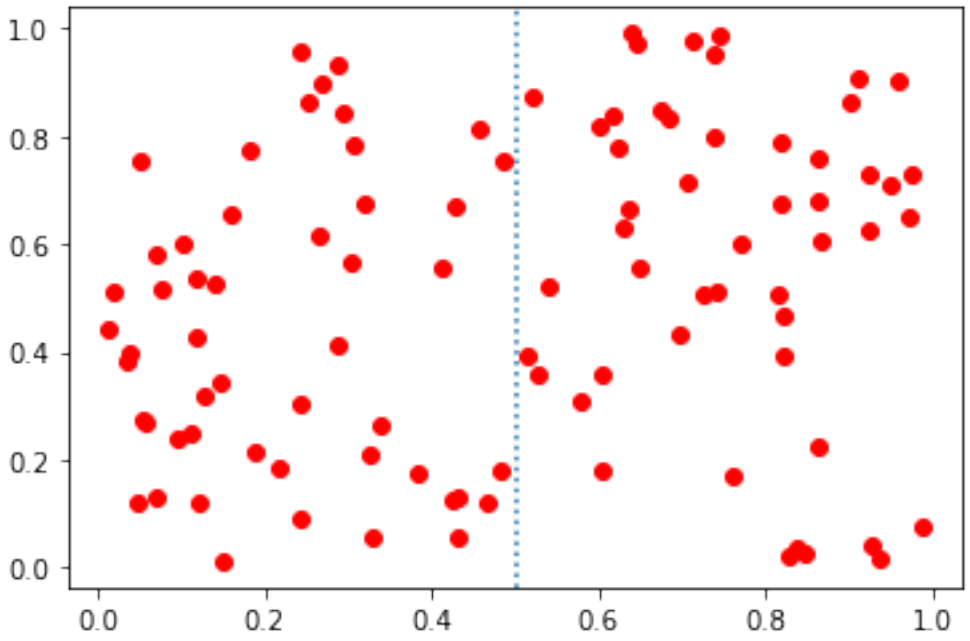
```
plt.scatter(x, y, color = 'red', marker = 'o')
plt.show()
```



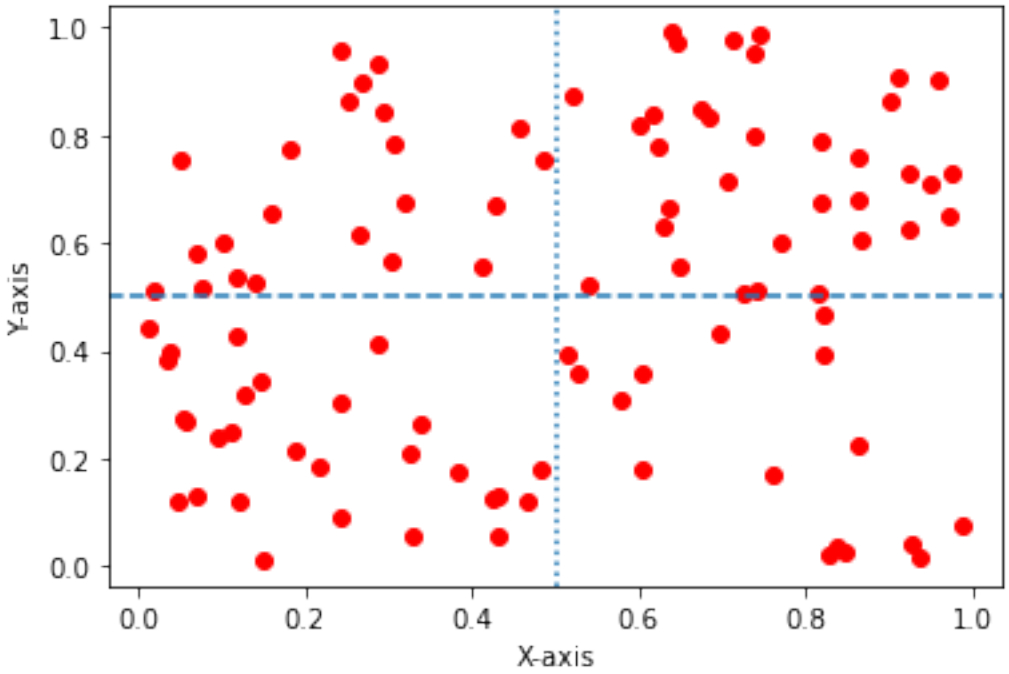
```
plt.axhline(y = 0.5, linestyle = '--', label = 'y = 0.5')
plt.scatter(x, y, color = 'red', marker = 'o')
plt.show()
```



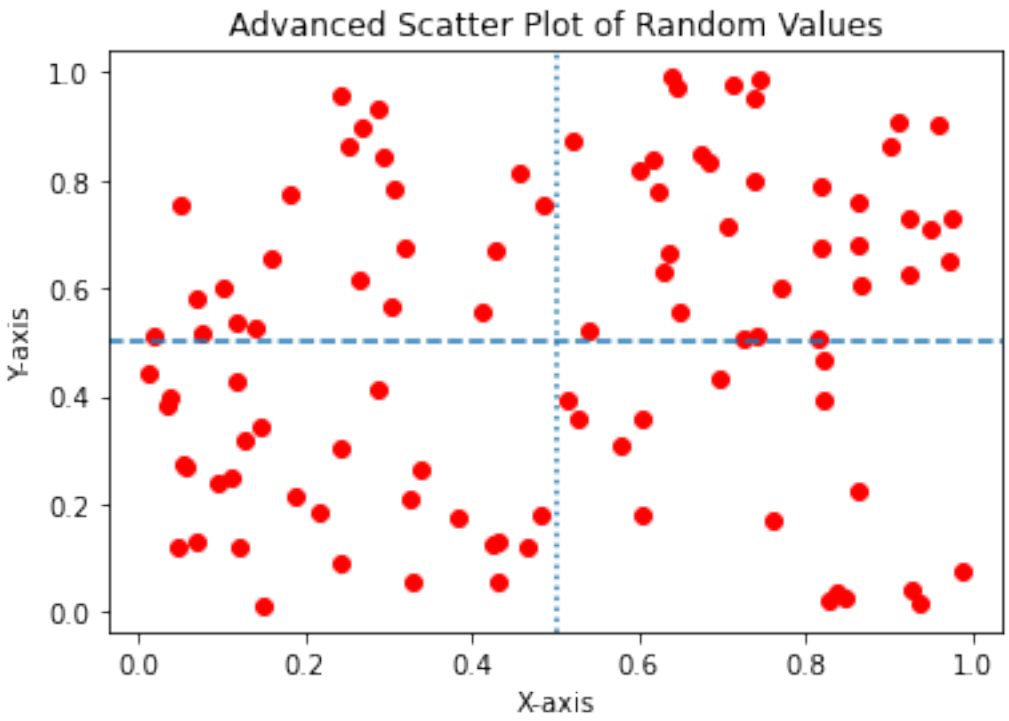

```
plt.axvline(x = 0.5, linestyle = ':', label = 'x = 0.5')
plt.scatter(x, y, color = 'red', marker = 'o')
plt.show()
```



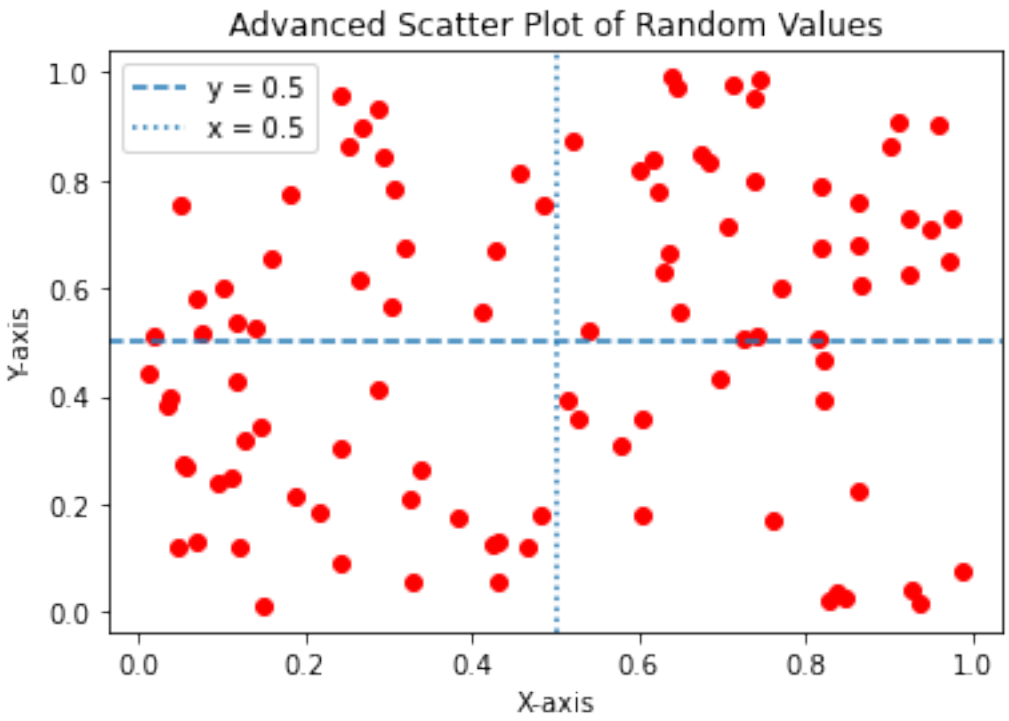
```
plt.axhline(y = 0.5, linestyle = '--', label = 'y = 0.5')
plt.axvline(x = 0.5, linestyle = ':', label = 'x = 0.5')
plt.scatter(x, y, color = 'red', marker = 'o')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```




```
plt.axhline(y = 0.5, linestyle = '--', label = 'y = 0.5')
plt.axvline(x = 0.5, linestyle = ':', label = 'x = 0.5')
plt.scatter(x, y, color = 'red', marker = 'o')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Advanced Scatter Plot of Random Values")
plt.show()
```




```
plt.axhline(y = 0.5, linestyle = '--', label = 'y = 0.5')
plt.axvline(x = 0.5, linestyle = ':', label = 'x = 0.5')
plt.scatter(x, y, color = 'red', marker = 'o')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Advanced Scatter Plot of Random Values")
plt.legend()
plt.show()
```



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

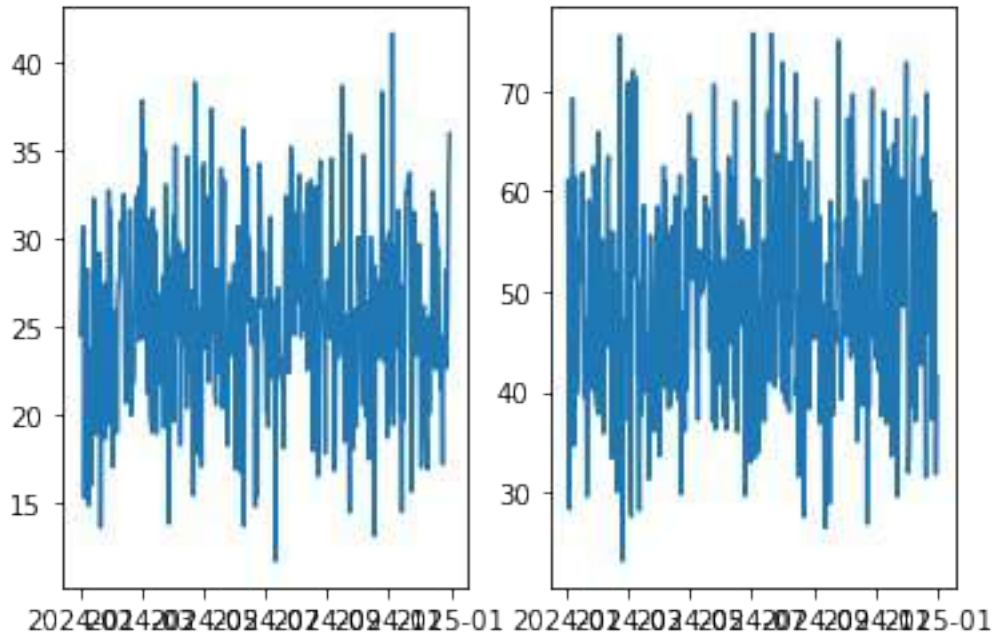
```
a) Plot the 'Temperature' and 'Humidity' on the same plot with
different y-axes (left y-axis for 'Temperature' and right y-axis for
'Humidity').
b) Label the x-axis as 'Date'.
c) Set the title of the plot as 'Temperature and Humidity Over Time'.

dates = pd.date_range(start='2024-01-01', end='2024-12-31', freq='D')
temperature = np.random.normal(loc=25, scale=5, size=len(dates))
humidity = np.random.normal(loc=50, scale=10, size=len(dates))
df = pd.DataFrame({'Date': dates, 'Temperature': temperature,
'Humidity': humidity})
df
```

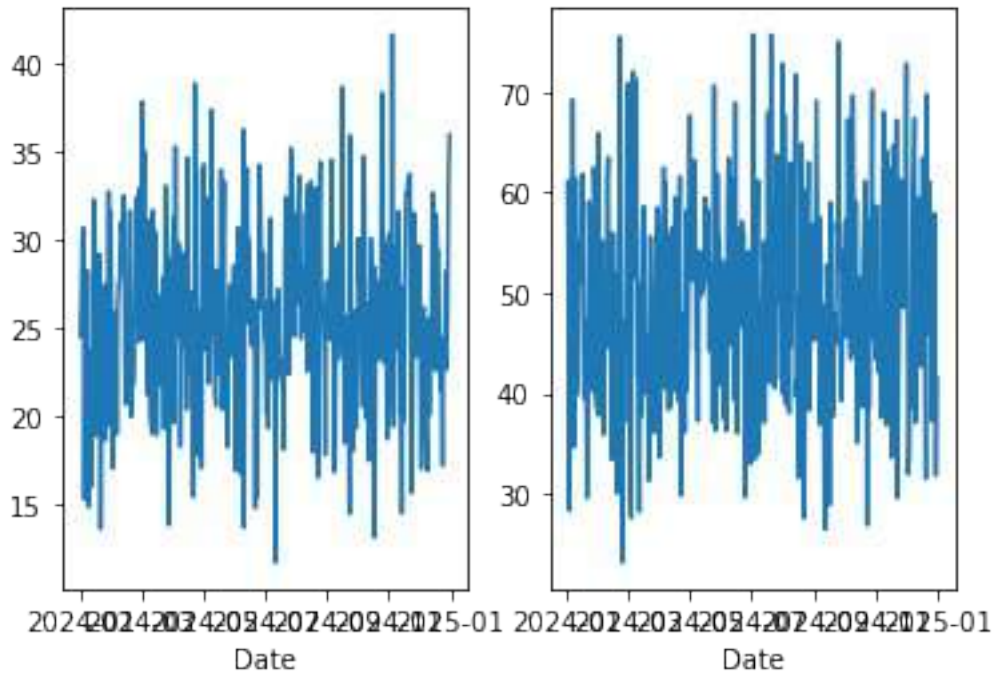
```
           Date  Temperature   Humidity
0    2024-01-01    24.520988  61.071895
1    2024-01-02    26.923908  47.602104
2    2024-01-03    30.075414  28.327537
3    2024-01-04    30.645158  57.860667
4    2024-01-05    15.271863  50.402462
..          ...          ...        ...
361  2024-12-27    28.181095  41.411586
362  2024-12-28    22.639237  45.622352
363  2024-12-29    25.874751  57.892680
364  2024-12-30    33.398381  31.804420
365  2024-12-31    35.915230  41.539272
```

```
[366 rows x 3 columns]
```

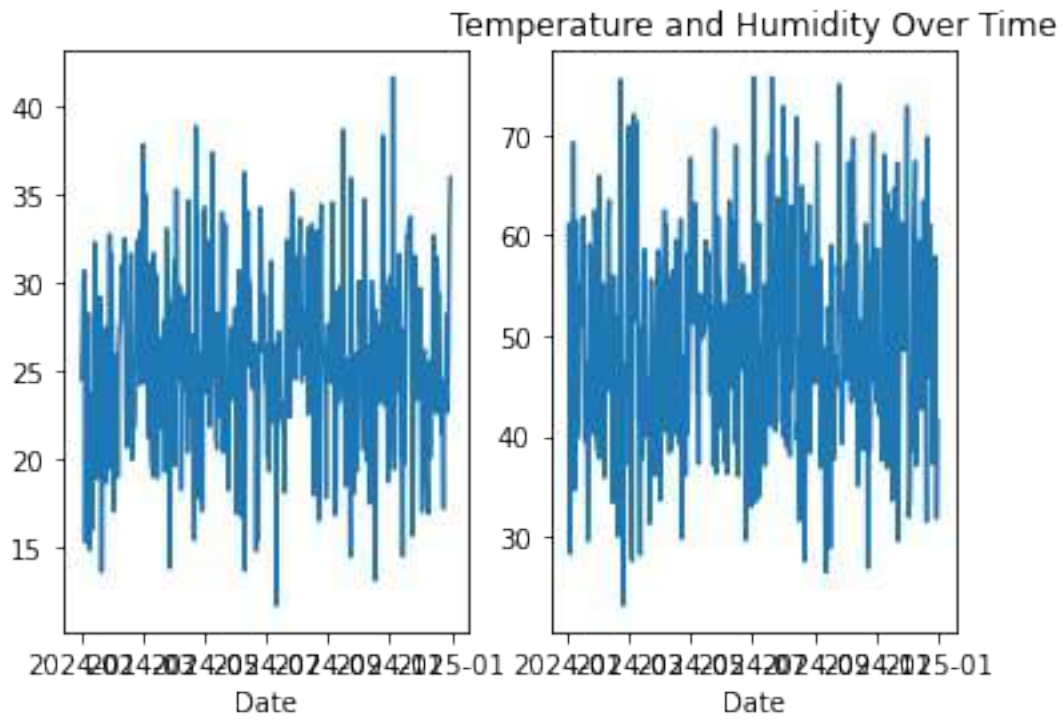
```
plt.figure()
plt.subplot(1, 2, 1)
plt.plot(df['Date'], df['Temperature'])
plt.subplot(1, 2, 2)
plt.plot(df['Date'], df['Humidity'])
plt.show()
```




```
plt.figure()
plt.subplot(1, 2, 1)
plt.plot(df['Date'], df['Temperature'])
plt.xlabel("Date")
plt.subplot(1, 2, 2)
plt.plot(df['Date'], df['Humidity'])
plt.xlabel("Date")
plt.show()
```




```
plt.figure()
plt.subplot(1, 2, 1)
plt.plot(df['Date'], df['Temperature'])
plt.xlabel("Date")
plt.subplot(1, 2, 2)
plt.plot(df['Date'], df['Humidity'])
plt.xlabel("Date")
plt.title("Temperature and Humidity Over Time")
plt.show()
```



15. 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'.

a = np.random.randn(1000)
a

array([-1.19969749e+00,  3.37946031e-01,  3.43447408e-02,
1.00009045e+00,
        2.01388489e-01,  1.74715122e-01,  1.34886170e+00,
9.94209162e-02,
       -7.61060319e-01,  5.77481577e-01, -1.31882284e+00, -
7.53277079e-01,
        5.52521152e-01,  1.14252271e+00, -5.50789850e-01,
5.15685776e-01,
       -6.85405686e-01,  5.81580321e-01,  1.65114839e+00, -
2.52215955e+00,
       -1.23472422e-01, -1.01835971e+00, -2.91575773e-01,
1.22280043e-02,
        1.36292309e-01, -1.94532474e-01,  6.46279111e-01, -
1.48226999e+00,
        8.74264135e-01, -1.96576446e+00,  5.70515611e-01, -
```

```
6.25184070e-01,
       -1.40025695e+00, -1.40104737e+00, -6.21371479e-01, -
1.15465841e+00,
        1.16722251e+00, -5.11616648e-01,  1.08855540e+00,
1.93675780e+00,
       -2.09264769e-01,  1.31857992e-02,  4.79496398e-02, -
1.45253395e+00,
        5.44647371e-01,  1.43735581e+00, -2.51605037e-01, -
4.00981492e-01,
        5.18315075e-01, -3.73111616e-01,  1.31533228e-01,
6.97989177e-02,
        2.84407021e-02, -7.86535293e-01, -9.49539536e-01,
3.38650928e-01,
       -2.89483213e+00, -1.57203149e+00, -1.15039732e+00,
1.23595383e+00,
        2.46638026e-01, -3.07394987e-01,  5.56439028e-01, -
1.45652352e+00,
       -6.54832943e-01,  1.01786172e+00, -7.07460664e-01,
1.01927500e+00,
        1.77364179e+00, -4.18392053e-01,  3.83772942e-01, -
2.58765762e-01,
       -2.59093253e-01,  4.34487745e-01, -4.24722795e-02,
6.37563765e-01,
        7.13427225e-01, -4.20500985e-01, -1.52666569e+00,
1.07342068e+00,
       -2.57336307e-01, -2.62462731e-01,  1.04072076e-01, -
5.88621381e-01,
       -6.92313495e-01,  5.38948306e-02, -4.63801873e-02,
6.40239621e-01,
       -2.87262147e-01,  1.51251209e+00, -4.96970411e-01, -
2.74177850e-01,
        7.08607192e-01, -7.10492373e-01, -6.12568010e-01, -
4.70694946e-01,
        1.64390155e-01, -7.91624384e-01,  1.26484026e+00,
3.15850952e-01,
       -8.55877305e-01,  7.91910933e-01,  8.30807530e-02,
3.59617419e-01,
        5.73085898e-01, -2.30552471e-01, -2.12797813e+00, -
5.18919752e-01,
       -7.00558323e-02,  1.81347973e+00, -5.68868964e-01,
1.35025201e+00,
       -1.69688956e-03,  1.80153478e-01,  2.07589292e+00,
3.08398687e-01,
        1.10602436e+00,  1.07984901e-01,  8.44221691e-01,
1.48256009e+00,
       -1.34224163e+00, -1.41591265e+00, -5.35509237e-01,
1.07480885e+00,
        1.51372685e-01,  1.90025271e-01, -3.47511300e-01,
2.12071385e-01,
```

```
       -9.22027755e-01, -1.21086105e+00, -8.26342223e-01,
1.89103645e-01,
        5.68646963e-01, -1.58081483e-01, -6.71677521e-01, -
1.90140749e+00,
       -1.31441381e-01, -6.40372030e-01, -1.34380739e+00,
1.15429506e-01,
       -9.86805845e-01,  7.39618485e-01, -2.37806652e+00,
9.23697399e-01,
       -8.91490658e-01,  1.03325429e+00, -7.39904662e-01,
5.50441428e-01,
       -2.17150894e-01,  7.99747829e-01, -4.72819080e-01, -
2.76938646e+00,
       -1.42878541e-01,  1.07572030e+00,  2.55314093e+00,
1.46626742e+00,
       -1.94325524e-02, -2.16544006e+00,  1.91198538e+00,
6.15604485e-01,
       -4.26863190e-01, -1.85563397e+00, -8.75663950e-01,
6.01428052e-01,
       -1.31900856e+00,  8.09726565e-01,  9.31095569e-01,
2.41803365e-02,
        6.32509776e-01,  1.37080594e+00, -1.12361141e+00,
1.48863889e+00,
       -7.99233214e-01,  1.26818623e+00, -4.94411737e-01,
8.31034974e-01,
        5.64065141e-01, -1.12515492e-01,  1.55089993e+00,
9.39379978e-02,
        1.02090262e+00, -5.55905942e-01, -8.85374927e-01, -
5.89948637e-01,
        2.68439346e-02,  1.06383502e+00,  1.07328872e+00,
2.69637589e-01,
        1.17997547e+00,  6.94257404e-01,  3.59504747e-01, -
2.35470010e+00,
       -2.34508687e-01,  1.08160677e+00,  1.36892615e+00, -
1.48794255e+00,
        1.08182503e+00, -4.13041832e-01,  1.33834476e+00, -
3.78745632e-01,
       -1.62264340e+00, -1.33657117e+00, -3.72801005e-01,
5.23016561e-01,
        5.47258566e-01,  1.76407426e+00,  7.02809492e-01, -
3.08097147e-01,
        5.11272333e-01,  1.25252662e+00, -9.38070048e-01,
9.88461334e-01,
       -2.26597531e-01,  1.22687527e+00,  2.18456203e-01,
4.15170472e-01,
        2.55389143e-01, -8.65856137e-01,  1.61435966e+00, -
4.06308092e-01,
       -4.94018003e-02, -7.17040982e-01, -2.07836903e+00,
1.23501486e+00,
        1.69143742e+00,  3.81023572e-01, -1.00349105e-01, -
```

```
6.18624448e-01,
        1.58734200e+00, -7.34929846e-01,  1.53289529e+00,
1.28225766e+00,
        3.33662591e-01,  3.12963847e-01, -4.93218377e-01, -
8.40419541e-01,
       -1.18887763e+00, -3.96173271e-01,  3.83493763e-01, -
1.52183865e-01,
        9.24987453e-01, -3.55676308e-01,  9.29471665e-01,
1.69326963e+00,
       -6.37788217e-01, -9.60907091e-01,  2.00769380e+00,
4.70394796e-01,
       -8.48365058e-02,  2.79406509e+00, -4.25070446e-01,
2.31502105e-02,
        6.31002187e-01,  3.72973515e-01,  8.00638069e-01, -
3.84355292e-02,
        4.76800513e-01, -7.80727634e-01, -3.10330270e-01, -
9.28116999e-01,
       -4.23393214e-01, -1.63517786e-01, -1.44170788e+00,
8.14437598e-01,
        7.01461407e-01,  8.71056428e-01,  9.76840191e-01, -
8.81059403e-01,
       -1.39906909e+00,  1.34782890e+00, -4.01734152e-01,
6.30549664e-01,
       -1.00495830e+00, -2.76455201e-01, -1.20873734e-01,
2.09464334e-01,
        3.16029846e-01,  1.76224952e+00, -5.64167050e-03, -
1.50637071e+00,
        2.67564251e-01, -5.87087949e-01, -1.11866902e-01,
2.55730127e-01,
       -1.73789821e-01,  6.32366595e-01, -4.97310970e-01, -
7.73184013e-01,
        3.65911926e-01, -1.44114917e+00, -7.74268575e-01, -
5.15380879e-01,
        1.16131364e+00, -8.71598860e-01, -1.28123504e+00, -
1.04206566e+00,
        3.59687161e-01, -3.33380126e-01,  1.38655130e+00, -
3.02622603e-01,
       -1.32205091e+00,  4.59026480e-01,  3.36707157e-01, -
9.07146160e-01,
       -1.89219965e+00,  1.90244006e+00, -2.63198759e-01,
4.36052827e-01,
       -2.28010667e-02, -9.47510205e-01,  2.56308283e-01, -
7.54673367e-01,
       -3.03125198e-01,  1.85714984e-01,  9.04221286e-02,
3.47763370e-01,
        4.11190654e-02, -1.67818575e+00, -2.38993121e-01,
4.47264798e-01,
       -2.75082037e-01,  1.43875794e-01, -8.44190637e-01,
2.33180670e+00,
```

```
       -5.88287102e-01,  3.32191828e-01,  1.00132086e+00,
4.84957375e-01,
        4.72187481e-01, -1.28142919e+00, -1.42221614e+00, -
2.63359382e-01,
       -3.50201288e-01,  7.07437884e-01,  4.26911462e-02,
1.59541125e-01,
        4.67180155e-01,  3.11387369e-01,  1.32826674e+00, -
6.43484564e-01,
       -5.32888558e-01, -9.38076762e-02,  9.30420092e-01, -
1.25851789e-01,
       -5.78510675e-02, -1.69854622e+00,  3.63717156e-01, -
4.94788874e-01,
       -7.89352945e-01,  5.70857585e-01,  8.98256950e-01,
2.52563789e-01,
       -3.46610482e-01, -3.87917531e-01,  1.22242109e+00, -
6.08214169e-01,
       -1.06826418e+00,  5.10889663e-01, -7.58370365e-01, -
2.43282375e+00,
        4.89011962e-02,  4.76925392e-01, -1.40202694e-01,
1.96269373e+00,
       -2.54733375e+00, -1.08518230e+00, -1.02275743e+00,
5.59021112e-01,
       -3.52862229e-01, -1.91389222e-01,  9.63857026e-01,
1.44302724e-01,
        1.00873725e+00,  7.95263125e-01,  1.78003072e+00,
2.75249973e-01,
       -1.73412546e+00,  9.42952080e-01,  6.64592318e-01, -
2.73135529e-01,
       -1.62985141e-01,  1.31204329e+00,  1.57601208e+00,
1.17107141e+00,
       -1.56270731e+00, -2.97890012e-01, -8.61982637e-01, -
5.38049239e-01,
       -1.34758287e+00,  9.87825388e-01,  5.51478595e-01,
1.33283059e+00,
       -6.75158946e-01,  6.17269813e-01, -2.02084982e+00, -
6.56703064e-01,
       -1.59334680e+00,  4.35207242e-01, -1.85743098e+00,
1.56787687e-01,
       -1.34164917e+00, -1.32276326e+00, -7.80540979e-01,
5.98243382e-01,
       -5.11999316e-01,  5.45309034e-01, -8.95352611e-01, -
3.80675074e-01,
        9.31805716e-01,  4.53073695e-02, -8.65388838e-01, -
1.87105311e-01,
        7.55025530e-01, -8.98712684e-01,  6.58555072e-01,
1.50334486e+00,
        1.62478032e+00,  1.94027527e+00,  6.38556605e-01,
3.63431610e-01,
        6.12488503e-01,  1.69983589e+00,  4.28471940e-01,
```

```
6.04042177e-01,
        7.16453380e-01,  9.36036562e-01,  2.73389383e-01, -
1.47559500e-01,
        6.60613824e-01,  1.11572715e+00, -1.30726806e-01,
9.03618414e-01,
        4.17395341e-01,  2.78964010e+00,  2.15269607e+00,
1.15112042e-01,
       -5.53924248e-01,  1.73032497e+00,  3.30749830e-01, -
1.82186322e+00,
       -5.20497882e-01,  1.24986771e+00, -2.05099660e+00, -
6.52476743e-02,
       -1.19178507e+00,  6.21288266e-01,  1.17252620e+00,
2.05941574e-01,
       -2.07359369e-01,  7.97632616e-01,  8.44878621e-01, -
8.84378655e-01,
       -1.06263901e+00,  1.58216952e+00,  3.57123954e-01, -
6.37630061e-02,
        1.95874583e+00,  8.25621279e-01,  4.11063812e-01, -
7.32595271e-01,
       -3.34288095e-01,  9.09980431e-01,  1.72067592e+00,
4.86699252e-01,
       -1.85965372e+00,  7.35123109e-01,  7.44600306e-01,
9.92358241e-01,
       -9.84121574e-01,  2.12267291e+00,  5.66501208e-01,
5.03552272e-01,
        5.76033876e-02,  5.12282185e-01, -5.71701718e-01, -
4.45856939e-01,
       -5.92181660e-01, -8.55179603e-01, -1.33133226e-01,
6.63363736e-01,
        2.16184899e+00, -4.97365251e-01, -9.75840748e-02,
1.26603932e+00,
       -1.72553220e+00,  7.70241718e-01,  9.95573229e-02, -
2.66208707e+00,
        9.10960669e-01,  1.54682408e+00,  5.29521250e-01,
6.28396291e-02,
        9.73451954e-01, -8.11665509e-01,  1.27564102e+00,
6.81528784e-01,
       -2.52463793e-03,  1.68335203e+00, -1.70934269e+00,
1.69220658e-01,
        8.91601528e-01,  1.32998462e+00,  2.04458552e-01,
1.20331997e+00,
       -1.60168294e+00, -5.47836428e-01, -4.81143128e-01, -
5.71364334e-01,
        1.68395841e+00,  3.02808413e-02,  9.01516096e-02, -
1.82996266e+00,
       -1.04944745e+00,  4.02053477e-01,  4.54597550e-01, -
1.55901869e-01,
       -7.68099747e-01, -3.18208145e-01, -1.29449234e+00, -
8.92085722e-02,
```

```
        8.54604448e-01,  1.14377527e+00, -6.78452555e-01, -
1.02451257e-02,
       -6.44013901e-01, -7.50282354e-01,  1.76555484e+00,
5.92137547e-02,
       -7.78019032e-01, -8.39012096e-01,  7.55520267e-01,
7.14501656e-02,
        4.83354530e-01,  1.15805321e+00,  3.38748268e-01, -
1.66260158e-01,
       -3.81744623e-02,  1.26075252e-01,  1.74991214e+00, -
3.69921603e-02,
        5.16810634e-01,  1.34140205e+00, -8.12221067e-01,
1.45318333e+00,
        3.29433413e-01,  8.12111443e-01,  5.42353342e-01, -
7.16669619e-01,
       -1.76909057e-01, -1.22428046e+00, -4.98849943e-01, -
1.25685665e+00,
       -5.89881949e-01, -1.19304158e+00,  1.67870942e+00,
5.57470373e-01,
       -2.58294094e-01,  5.65373246e-01,  1.33106242e+00, -
4.11039062e-01,
       -7.49703788e-02, -2.16692055e+00, -2.22810837e+00, -
6.19696842e-01,
        8.84269614e-02, -1.40932173e+00,  1.57447950e-01, -
5.05165216e-01,
       -1.09319024e+00, -5.58060806e-01,  6.94197946e-01, -
2.05328011e+00,
        5.82813353e-01, -1.61159696e+00,  1.17840008e-01,
5.69568679e-01,
       -7.73099994e-01,  4.35762434e-02,  9.70640506e-01, -
8.39942734e-01,
        1.04687219e-01,  1.03381485e+00, -1.04933126e+00, -
7.20987103e-01,
        2.02924609e+00, -1.40486027e-01, -1.58759560e+00,
8.18431862e-01,
       -4.48311576e-01,  9.84096712e-01,  3.45199879e-01,
1.02201664e+00,
       -5.51805525e-01, -4.11621652e-01,  2.48790097e-01,
1.47174611e+00,
        1.06041018e+00,  2.33069964e-01, -2.68395569e-01,
1.18849250e+00,
       -4.54310328e-01,  3.80264459e-01, -2.68169411e-01, -
1.87337726e+00,
       -4.24480787e-02, -1.73385898e-02, -3.89368053e-01, -
9.70523793e-01,
       -2.12198799e+00,  1.10587031e+00, -3.23329348e-02, -
1.47303565e+00,
       -8.44075939e-01,  2.14583574e+00, -1.05103020e-01, -
2.68594107e-01,
       -6.53345109e-01,  1.41939841e+00, -8.76612161e-01,
```

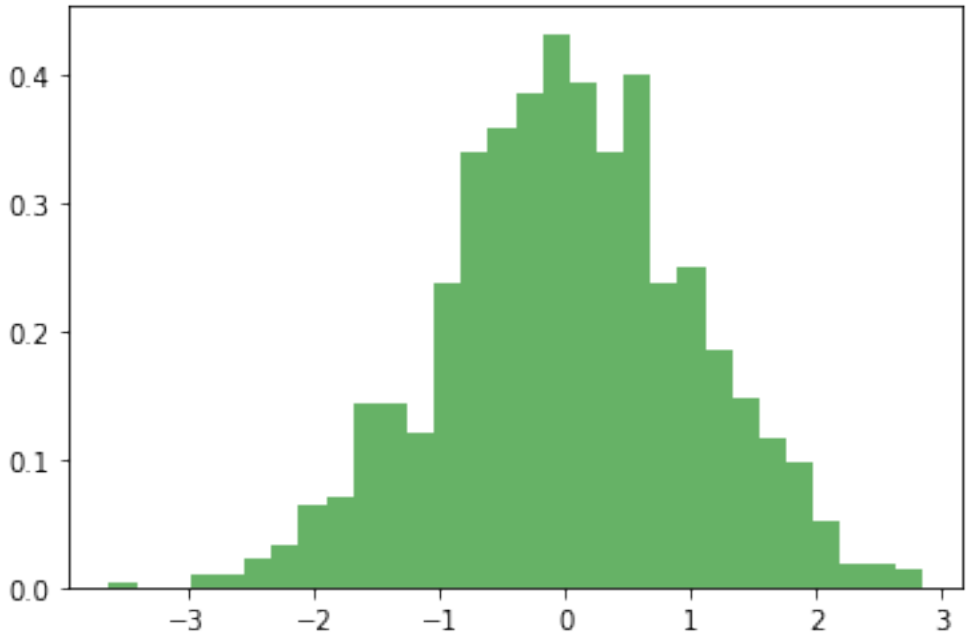
```
5.20886418e-01,
        3.41804177e-01,  3.61038618e-01, -1.10121377e-02, -
6.66913537e-01,
       -9.34775578e-01,  7.69289179e-01, -7.07265480e-01,
6.35037813e-01,
       -9.92308231e-01,  7.00899557e-02,  5.17433642e-01,
5.62727544e-01,
       -3.10859641e-02,  5.90263714e-01,  1.12980056e+00, -
1.70040585e-01,
       -1.47429951e+00, -7.77816172e-02, -6.32232429e-01, -
2.02253850e+00,
       -6.86775727e-01, -2.55037626e-01, -6.39761581e-01, -
1.31360246e+00,
        1.55511964e+00,  3.15233980e-01, -7.39973367e-01,
1.32617485e+00,
       -6.87997805e-01,  2.37261279e-02,  3.91685269e-01,
1.74159941e+00,
        8.65188917e-01, -1.49357883e+00, -5.99970869e-01,
1.07809753e+00,
        1.23485645e+00, -7.20376715e-01,  8.60001529e-02,
6.02128546e-03,
       -1.09730847e-01,  1.69797500e+00,  5.38398023e-01, -
1.67319915e+00,
        4.55551029e-01,  4.96169147e-01, -6.24316125e-01, -
2.37783986e-01,
        2.88481418e-01,  7.12264871e-01, -1.19544001e+00, -
6.14132413e-01,
       -1.08300175e+00,  1.76684978e+00,  1.62119884e+00,
3.17532912e-01,
       -6.54308360e-01, -1.73745881e-01,  4.87723795e-01, -
1.96523310e+00,
        1.21461806e+00,  2.26634686e+00,  6.67179743e-01,
7.24120787e-01,
        1.20757121e+00, -8.27248668e-01, -1.83212565e-01,
7.15925411e-01,
        6.77199081e-02, -7.22924956e-02, -6.01408120e-01, -
5.16333403e-02,
        2.53744084e+00,  9.94778027e-01,  1.81206422e+00,
2.75779551e-01,
       -1.00120928e+00, -1.10369973e+00, -1.68392073e+00, -
5.00414500e-01,
        5.86975179e-01, -1.68092129e+00, -2.38318830e-01, -
2.84043420e-02,
        2.09526407e+00,  9.29678496e-01, -1.95499244e-01,
7.20512419e-01,
        1.89925952e+00,  5.76799523e-01, -6.62259061e-02,
4.23597026e-01,
        1.03157280e+00, -6.43306402e-01, -1.04621114e+00, -
1.58201255e+00,
```

```
        2.33465506e+00,  3.81360294e-01, -2.93888473e-01,
5.21295491e-02,
        1.67675086e+00,  1.67128400e+00, -8.31086419e-01,
1.35345936e+00,
        8.79426193e-01,  1.76207445e+00,  2.36141095e-01, -
2.42468795e-01,
        4.08303390e-02,  2.06539398e+00,  7.52991261e-02, -
2.09960637e+00,
        7.07481092e-01, -1.97380801e+00, -4.29399530e-01,
1.18234869e+00,
        2.56101550e-01, -3.72463997e-01,  2.03251419e+00,
1.03072320e+00,
       -7.33103189e-02,  1.59400477e+00, -1.37909553e+00, -
1.35987115e+00,
       -1.39652560e+00, -3.79622591e-02,  8.91692824e-02,
3.60347412e-01,
        6.90503550e-01,  3.93433664e-01,  7.82422731e-01,
4.76786500e-01,
       -2.38711169e+00, -3.85703957e-02, -1.67185844e+00,
4.40471310e-01,
       -1.95598292e-01, -3.83359327e-01, -1.35621833e+00, -
1.18778107e-01,
        5.04639650e-01, -1.82374240e-01, -6.65064498e-02,
8.58290577e-01,
       -5.71026705e-01,  1.24310555e+00, -1.00478376e+00,
1.45013143e+00,
       -2.15274890e+00, -1.10673977e+00, -1.57764632e+00, -
2.64387373e-01,
       -4.30283377e-01,  7.52404027e-03, -1.09173255e+00,
1.30644828e+00,
        7.03155672e-01, -4.34917138e-01,  3.83843251e-01, -
1.88144030e+00,
        1.09255266e+00, -6.46650581e-01, -1.58512263e+00,
1.87993094e-01,
        8.51443399e-01, -5.47867705e-01,  5.36921714e-01, -
8.36322385e-01,
        8.59124022e-01,  5.17532301e-01,  1.87714396e+00,
1.27427630e+00,
       -7.79519043e-01, -2.17540278e+00,  9.81272632e-01,
1.91649973e+00,
       -2.66742404e-01, -7.82359081e-01, -2.96080096e-01, -
9.87340529e-01,
        7.18092557e-02,  4.00880361e-01,  1.26668900e+00, -
1.65647031e-01,
        2.60943950e-01,  1.23734687e-01,  4.69977411e-01, -
1.42004434e-01,
       -4.22574951e-01,  1.14470357e+00, -7.10952467e-01, -
1.68041102e-01,
       -1.58747008e+00, -3.18699817e-01,  4.16102670e-01, -
```

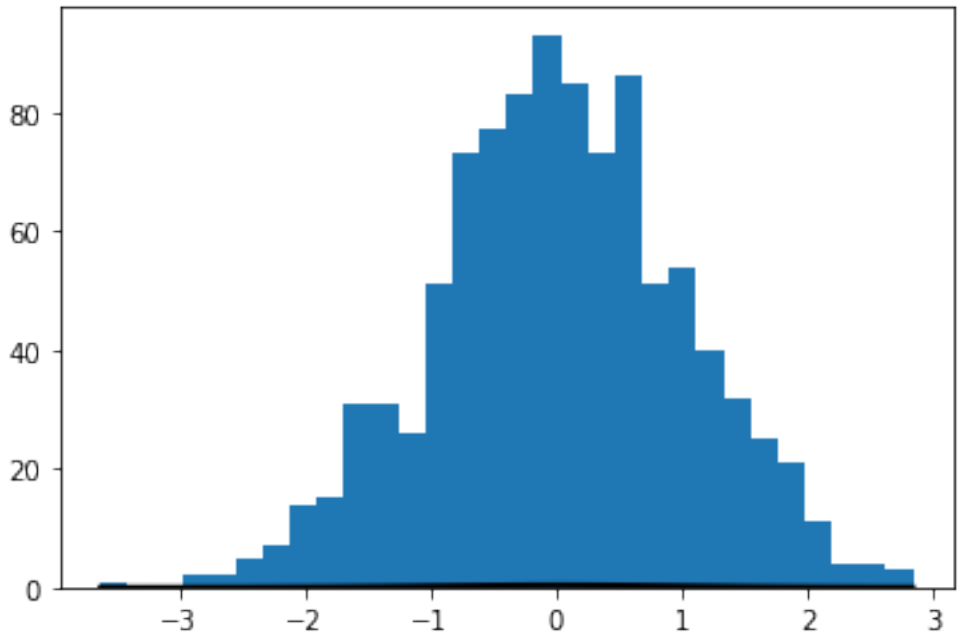
```
1.15867065e+00,
        2.59192128e-01, -1.14082927e-01,  1.85498107e+00,
1.37759858e+00,
        1.30219095e-01, -4.11979472e-01,  1.87180203e-01,
2.04929300e+00,
       -4.84371145e-01,  3.03432624e-01, -1.70386994e+00,
1.27745887e+00,
       -3.71020699e-01, -1.53948527e+00, -2.63255236e-01,
2.37106415e-01,
        9.31207705e-01,  1.18134642e-01,  1.87538490e-01, -
9.86499104e-01,
       -1.07195864e+00,  1.20065831e+00, -2.72246213e-01, -
1.47144381e+00,
        2.47632702e-01,  1.36756418e+00,  1.46230397e+00,
7.49517349e-01,
        6.78020671e-01, -4.42924657e-01,  1.40740088e+00, -
1.88221537e-01,
        1.37918745e+00, -3.81087041e-01,  5.57948613e-01, -
1.27654690e-01,
       -1.10034442e+00,  1.50568305e+00,  2.13805948e-02,
1.10349569e+00,
       -6.66462448e-02, -2.26537741e-01,  1.10567623e+00, -
1.00443438e-02,
       -7.18075248e-01, -7.48144656e-01,  1.21801058e+00,
1.79862651e-01,
        7.21123814e-01, -3.61568063e-01, -2.50112102e-02, -
8.36615753e-01,
       -5.96418078e-01, -1.48258942e+00, -3.62270773e+00,
1.72287135e-01,
       -6.23589607e-01,  7.20390266e-01,  4.80279791e-02, -
2.55116264e-01,
       -2.57249437e-01, -4.78353705e-01, -1.39956205e+00,
4.88692368e-02,
        1.02696232e+00,  3.22474260e-01,  1.95906502e-02, -
3.47153547e-01,
       -4.45134714e-01, -7.98055715e-01,  1.45026147e-01, -
5.63102621e-01,
       -2.65932376e-01,  5.41688201e-02, -9.05318435e-01,
3.73708949e-01,
       -3.37997101e-01, -7.99750378e-01, -3.90245507e-01,
1.35988033e+00,
       -2.81533829e-01,  3.51653056e-01, -2.71801204e-01, -
5.84774648e-01,
        1.48102020e+00, -7.80005034e-01, -8.35977503e-01,
1.96592717e+00,
        6.47783843e-02,  1.61536098e+00, -1.04166271e+00,
1.08682179e+00,
        9.09432115e-01, -5.82833948e-01, -7.88583891e-01, -
1.16474171e-01,
```

```
       -1.62079782e+00,  2.66192182e-01, -1.48685329e+00, -
1.21433383e-01,
        1.10996730e-01, -8.31443658e-01, -1.96173781e+00, -
5.06983700e-01,
       -5.00832179e-01, -1.45597673e+00,  5.45214813e-02, -
8.05866457e-02,
       -1.25052996e-01,  1.02427721e+00, -1.50766756e+00, -
3.65901688e-01,
        4.77411824e-01,  7.39314682e-01, -8.15111766e-01,
7.68158895e-03,
        1.21961308e+00, -5.01609364e-02, -1.79508545e-02, -
6.33019111e-01,
        5.40637963e-01, -6.80755026e-01,  4.95173792e-01,
4.56976480e-01,
       -2.12246655e-01, -7.62211671e-03, -5.09162755e-01, -
5.44217279e-01,
        6.77581376e-01, -3.00149481e-01,  8.08698672e-02,
8.85882951e-02,
       -8.25516950e-01, -5.92252408e-01, -1.79381953e-01,
2.47317266e+00,
       -2.02326672e+00,  1.64646328e-01,  1.85400636e+00, -
4.27730259e-01,
       -4.36689281e-01,  5.84656684e-01,  1.16954644e+00, -
5.44225572e-01,
       -1.44740341e-01,  5.24811549e-01, -2.08157128e+00,
3.52855800e-01,
       -4.19091668e-01, -1.46782575e+00, -6.77647370e-01,
1.19594683e-01,
        2.49154400e+00, -2.41971667e-01,  2.83706657e+00,
4.87669702e-01,
       -2.04027660e+00, -3.14667656e-01, -1.48599370e+00, -
1.48251812e+00,
       -9.69675059e-01,  5.43689897e-01,  2.13237359e-01,
7.62792833e-01,
       -8.64197457e-01,  1.51081060e+00,  1.59575836e-01,
9.07010480e-01,
       -8.66631985e-01, -1.88241822e+00, -7.01801954e-01,
2.92140159e-01,
        6.82979300e-02,  1.91654226e-01, -5.92198768e-01,
5.93566695e-01,
       -7.82794267e-01,  5.78655396e-01, -1.03384174e+00, -
1.23733650e-01,
        6.39680973e-01,  2.31839699e+00, -1.68715203e+00, -
1.22649297e-01])

count, bins, ignored = plt.hist(a, bins=30, density=True, alpha=0.6,
color='g')
plt.show()
```

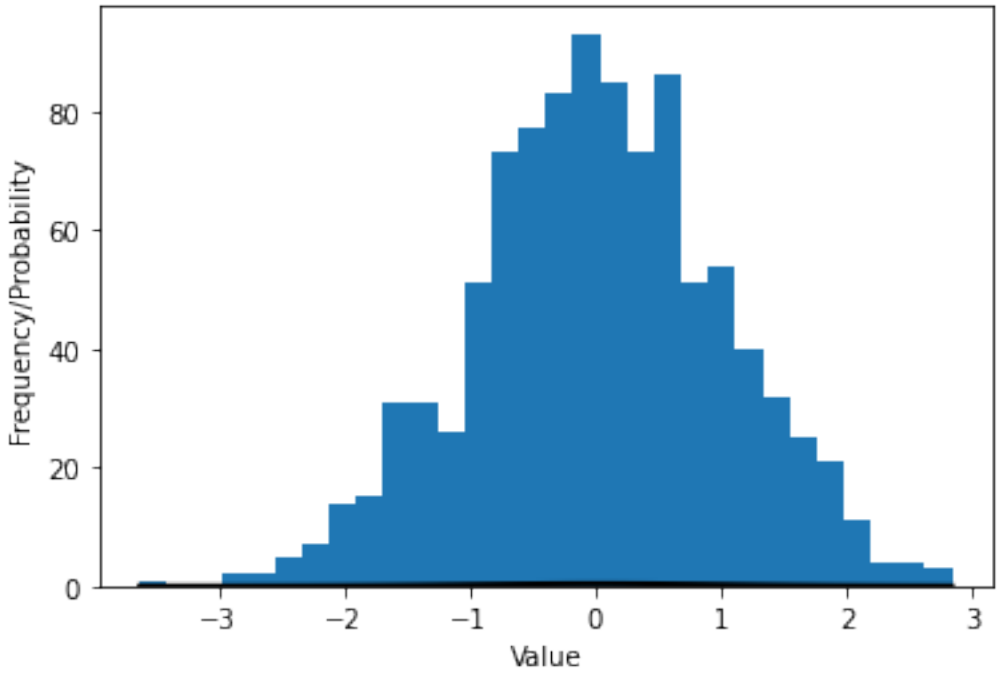



```
plt.hist(x = a, bins = 30)
mean, std_dev = np.mean(a), np.std(a)
pdf_x = np.linspace(min(bins), max(bins), 100)
pdf_y = norm.pdf(pdf_x, mean, std_dev)
plt.plot(pdf_x, pdf_y, 'k', linewidth=2, label='PDF')
plt.show()
```

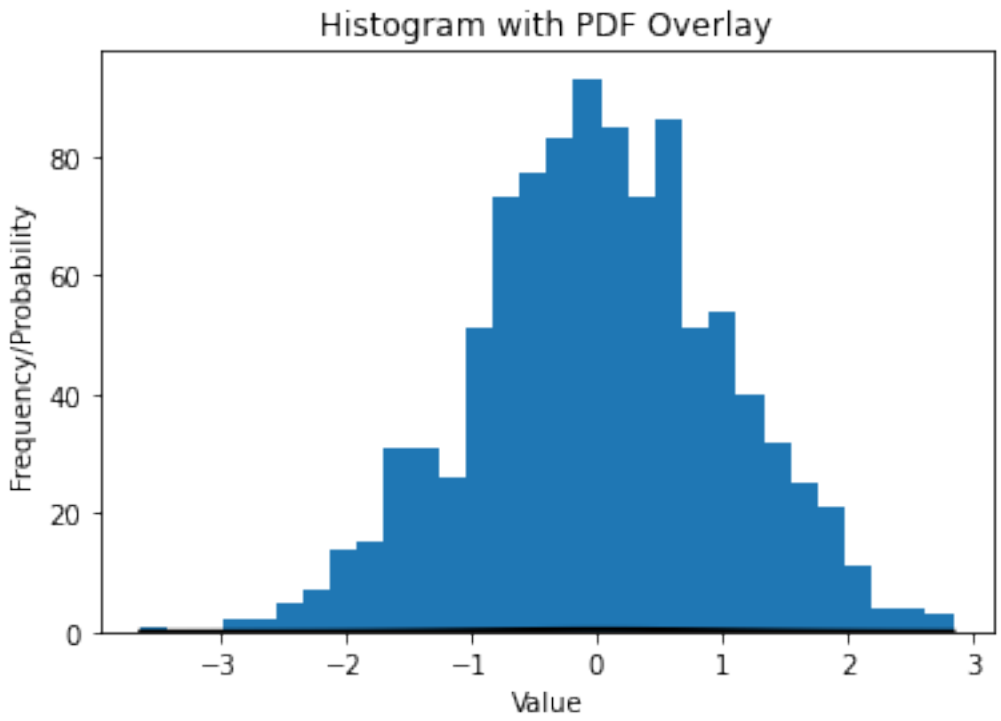



```
plt.hist(x = a, bins = 30)
mean, std_dev = np.mean(a), np.std(a)
```

```
pdf_x = np.linspace(min(bins), max(bins), 100)
pdf_y = norm.pdf(pdf_x, mean, std_dev)
plt.plot(pdf_x, pdf_y, 'k', linewidth=2, label='PDF')
plt.xlabel("Value")
plt.ylabel("Frequency/Probability")
plt.show()
```

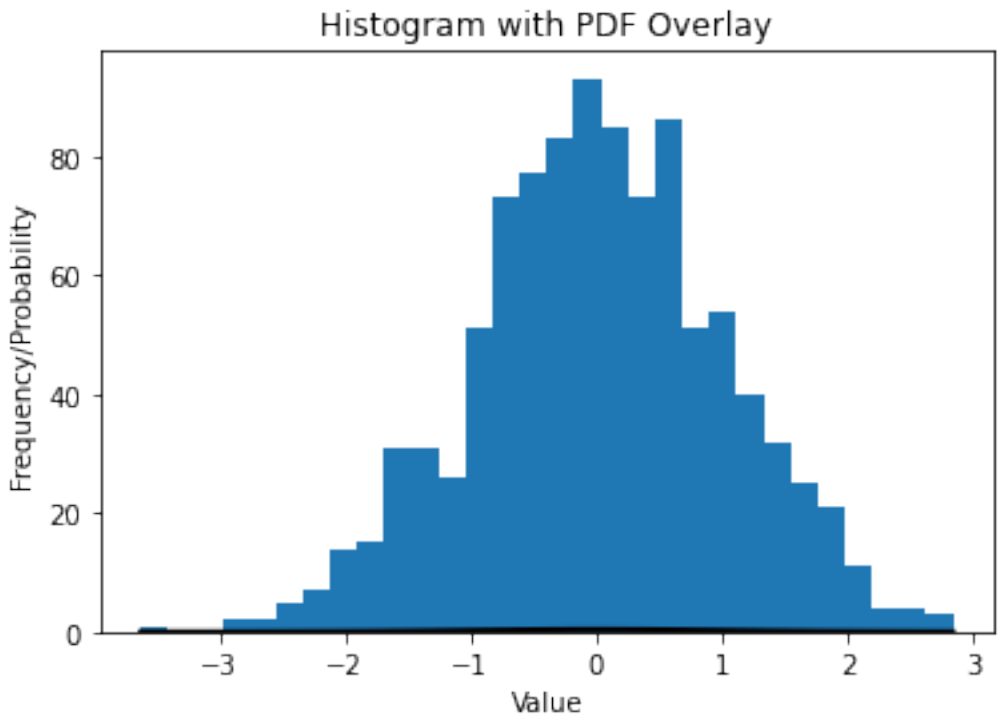



```
plt.hist(x = a, bins = 30)
mean, std_dev = np.mean(a), np.std(a)
pdf_x = np.linspace(min(bins), max(bins), 100)
pdf_y = norm.pdf(pdf_x, mean, std_dev)
plt.plot(pdf_x, pdf_y, 'k', linewidth=2, label='PDF')
plt.xlabel("Value")
plt.ylabel("Frequency/Probability")
plt.title("Histogram with PDF Overlay")
plt.show()
```



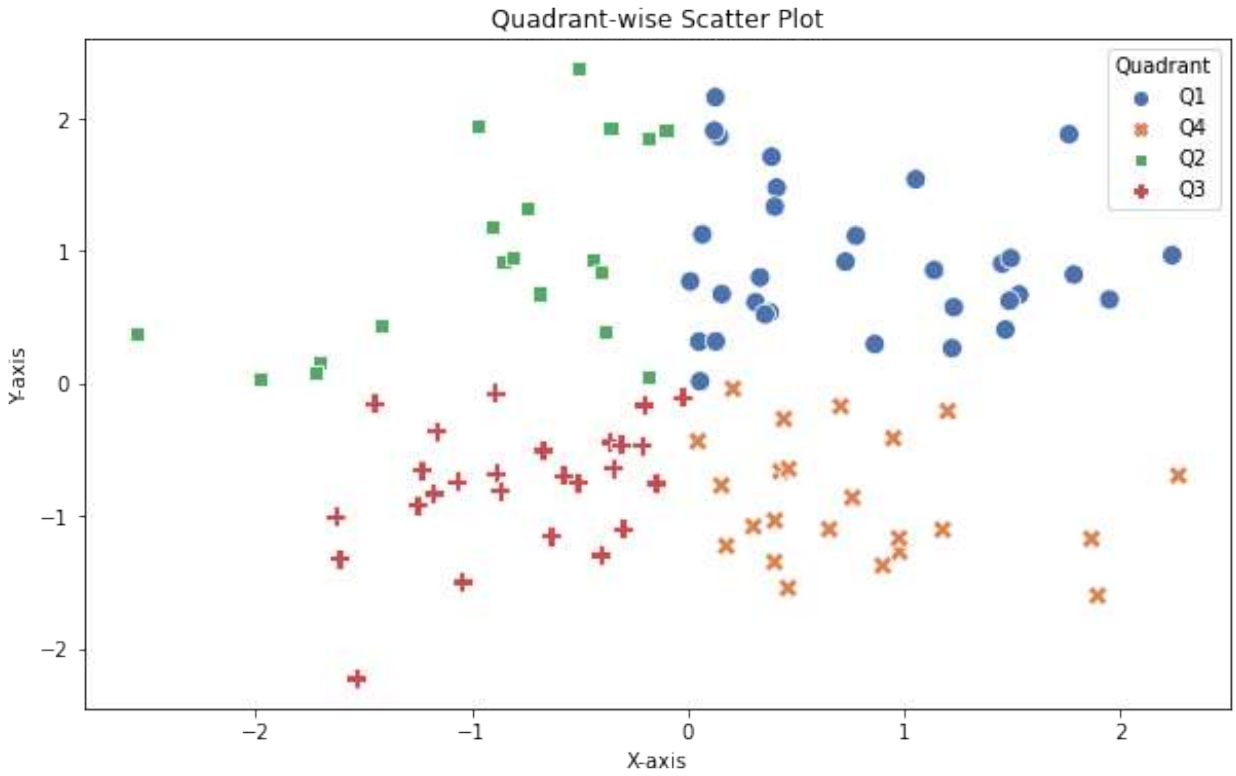
16. Set the title of the plot as 'Histogram with PDF Overlay'.

```
plt.hist(x = a, bins = 30)
mean, std_dev = np.mean(a), np.std(a)
pdf_x = np.linspace(min(bins), max(bins), 100)
pdf_y = norm.pdf(pdf_x, mean, std_dev)
plt.plot(pdf_x, pdf_y, 'k', linewidth=2, label='PDF')
plt.xlabel("Value")
plt.ylabel("Frequency/Probability")
plt.title("Histogram with PDF Overlay")
plt.show()
```



17. 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 Scatter Plot'.

```
np.random.seed(0)
x = np.random.randn(100)
y = np.random.randn(100)
def get_quadrant(x, y):
    if x >= 0 and y >= 0:
        return 'Q1'
    elif x < 0 and y >= 0:
        return 'Q2'
    elif x < 0 and y < 0:
        return 'Q3'
    else:
        return 'Q4'
quadrants = [get_quadrant(xi, yi) for xi, yi in zip(x, y)]
df = pd.DataFrame({'x': x, 'y': y, 'quadrant': quadrants})
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(data=df, x='x', y='y', hue='quadrant',
palette='deep', style='quadrant', s=100)
scatter_plot.legend(title='Quadrant')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
plt.show()
```



18. 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'.

```
output_notebook()
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)
p = figure(title="Sine Wave Function", x_axis_label='X-axis',
y_axis_label='Y-axis')
p.line(x, y, legend_label="Sine Wave", line_width=2)
show(p)
```

```
"\n(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\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(\"1002\");\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\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  \n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-2.4.1.min.js\",
```

```
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-2.4.1.min.js\"];\n
const css_urls = [];\n  \n\n  const inline_js = [\n    function(Bokeh)
{\n      Bokeh.set_log_level(\"info\");\n    },\n    function(Bokeh)
{\n    \n    \n    }\n  ];\n\n  function run_inline_js() {\n    \n
if (root.Bokeh !== undefined || force === true) {\n      \n    for
(let i = 0; i < inline_js.length; i++) {\n
inline_js[i].call(root, root.Bokeh);\n    }\n    if (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(\"1002\")).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\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));"

""
```

19. 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.models import ColumnDataSource, HoverTool
from bokeh.transform import factor_cmap
output_notebook()
categories = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(1, 100, size=len(categories))
df = pd.DataFrame({'categories': categories, 'values': values})
source = ColumnDataSource(df)
colors = ['#c9d9d3', '#718dbf', '#e84d60', '#ddb7b1', '#ffbf00']
color_map = factor_cmap(field_name='categories', palette=colors,
factors=categories)
p = figure(x_range=categories, title="Random Categorical Bar Chart",
x_axis_label='Categories', y_axis_label='Values',
           tools="pan,wheel_zoom,box_zoom,reset")
p.vbar(x='categories', top='values', width=0.9, source=source,
line_color='white', fill_color=color_map)
hover = HoverTool()
hover.tooltips = [("Category", "@categories"), ("Value", "@values")]
p.add_tools(hover)
show(p)
```

```
"\n(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\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(\"1114\");\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\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  \n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-2.4.1.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-2.4.1.min.js\"];\n
const css_urls = [];\n  \n\n  const inline_js = [\n    function(Bokeh)
{\n      Bokeh.set_log_level(\"info\");\n    },\n    function(Bokeh)
{\n    \n    \n    }\n  ];\n\n  function run_inline_js() {\n    \n
if (root.Bokeh !== undefined || force === true) {\n      \n    for
(let i = 0; i < inline_js.length; i++) {\n
inline_js[i].call(root, root.Bokeh);\n    }\n    if (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(\"1114\")).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\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));"

""
```

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

```
x = np.random.randint(1, 10, 5)
x
```

```
array([7, 7, 8, 9, 9])
```

```
y = np.random.randint(1, 10, 5)
y
```

```
array([8, 1, 9, 7, 9])
```

```
fig = go.Figure()
fig.add_trace(go.Scatter(mode = 'lines', x = x, y = y))
```

```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"mode":"lines","type":"scatter","x":[7,7,8,9,9],"y":
[8,1,9,7,9]}],"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}}}}}
```


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

```
np.random.seed(0)
categories = ['Category A', 'Category B', 'Category C', 'Category D',
'Category E']
values = np.random.randint(1, 100, size=len(categories))
df = pd.DataFrame({'Category': categories, 'Values': values})
fig = px.pie(df, names='Category', values='Values', title='Interactive
Pie Chart', labels={'Category': 'Categories'},
             hover_data={'Values': ':.2f'}, hole=0.0)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":[{"customdata":
[[45],[48],[65],[68],[68]],"domain":{"x":[0,1],"y":
[0,1]},"hole":0,"hovertemplate":"Categories=%{label}<br>Values=%
{customdata[0]:.2f}<extra></extra>","labels":["Category A","Category
B","Category C","Category D","Category
E"],"legendgroup":"","name":"","showlegend":true,"textinfo":"percent+l
abel","textposition":"inside","type":"pie","values":
```

```
[45,48,65,68,68]}],"layout":{"legend":{"tracegroupgap":0},"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"}}}
```