In [143]:
```
#1.Demonstrate three different methods for creating identical 2D arrays in NumPy• Provide t
#method and the final output after each method
```

In [144]:
```
#a.Using np.array
import numpy as np

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

```
Array created using np.array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [145]:
```
#b.Using np.zeros
array2 = np.zeros((3, 3), dtype=int)
array2[0] = [1, 2, 3]
array2[1] = [4, 5, 6]
array2[2] = [7, 8, 9]
print("\nArray created using np.zeros and filling values:\n", array2)
```

```
Array created using np.zeros and filling values:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [146]:
```
#c.Using np.full
array3 = np.full((3, 3), 0, dtype=int)
array3[0, :] = [1, 2, 3]
array3[1, :] = [4, 5, 6]
array3[2, :] = [7, 8, 9]
print("\nArray created using np.full and setting individual elements:\n", array3)
```

```
Array created using np.full and setting individual elements:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [147]:
```
#2.Using the Numpy function, generate an array of 100 evenly spaced numbers between 1 and 1
#Reshape that 1D array into a 2D array

import numpy as np

# Generate a 1D array of 100 evenly spaced numbers between 1 and 10
array_1d = np.linspace(1, 10, 100)

# Reshape the 1D array into a 2D array with shape (10, 10)
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.        ]]
```

In [148]:
```
#3Explain the following termsg
# The difference in npY.array, np.asarray and np.asanyarray.
# The difference between Deep copy and shallow copy.
```

Shallow Copy:

Creates a new object, but inserts references into it to the objects found in the original. For arrays, this means the new array object will reference the same data as the original array.

Deep Copy:

Creates a new object and recursively copies all objects found in the original, creating a fully independent copy. For arrays, this means the new array object will have its own copy of the data.

np.array always creates a new array, optionally copying data. np.asarray converts input to an array without copying if already an array. np.asanyarray behaves like np.asarray but retains the subclass type. Shallow copy references the same data, while deep copy creates an entirely independent copy.

In [149]:
```
#4. Generate a 4x4 array with random floating-point numbers between 5 and 20. Then, round ea
#the array to 2 decimal places.
import numpy as np

# Generate a 4x4 array with random floating-point numbers between 5 and 20
random_array = np.random.uniform(5, 20, (4, 4))

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

print("Random 4x4 array with numbers between 5 and 20:\n", random_array)
print("\nRounded 4x4 array (2 decimal places):\n", rounded_array)
```

```
Random 4x4 array with numbers between 5 and 20:
 [[10.33053273 10.35060336  5.24492754  7.77848488]
 [11.01889251 18.93937126  6.49422395 19.179523  ]
 [18.04232796 11.81243595  9.90051323  8.49116194]
 [14.2169706   5.49611887  5.23409097 11.43193584]]

Rounded 4x4 array (2 decimal places):
 [[10.33 10.35  5.24  7.78]
 [11.02 18.94  6.49 19.18]
 [18.04 11.81  9.9   8.49]
 [14.22  5.5   5.23 11.43]]
```

In [150]:
```
#5.Create a NumPy array with random integers between 1 and 10 of shape (5,6 ). After creati
#perform the following operations:
```

In [151]:
```python
#a)Extract all even integers from array.
import numpy as np

# Generate a 5x6 array with random integers between 1 and 10
random_array = np.random.randint(1, 11, (5, 6))

print("Random 5x6 array with integers between 1 and 10:\n", random_array)

# Extract all even integers from the array
even_integers = random_array[random_array % 2 == 0]
print("\nEven integers extracted from the array:\n", even_integers)
```

```
Random 5x6 array with integers between 1 and 10:
 [[ 6 10  5  7  6  2]
 [ 4  4  9 10  6  6]
 [ 7  1 10  8  6  2]
 [ 6  7  7  9  8  6]
 [ 4  3 10 10  4  3]]

Even integers extracted from the array:
 [ 6 10  6  2  4  4 10  6  6 10  8  6  2  6  8  6  4 10 10  4]
```

In [152]:
```python
#b)Extract all odd integers from array.
import numpy as np

# Generate a 5x6 array with random integers between 1 and 10
random_array = np.random.randint(1, 11, (5, 6))

print("Random 5x6 array with integers between 1 and 10:\n", random_array)

# Extract all odd integers from the array
odd_integers = random_array[random_array % 2 != 0]
print("\nOdd integers extracted from the array:\n", odd_integers)
```

```
Random 5x6 array with integers between 1 and 10:
 [[ 6  5  2  6  9  4]
 [ 6  9  5  2  8  9]
 [ 2  3  2  2  8  6]
 [ 1  5  2  2  7  7]
 [ 1  3  4  8 10  3]]

Odd integers extracted from the array:
 [5 9 9 5 9 3 1 5 7 7 1 3 3]
```

In [153]:
```python
#6.Create a 3D NumPy array of shape (3, 3, 3) containing random integers between 1 and 10.
#Perform the following operations:
```

In [154]:
```python
#a) Find the indices of the maximum values along each depth level (third axis).
import numpy as np

# Create a 3D array of shape (3, 3, 3) with random integers between 1 and 10
array_3d = np.random.randint(1, 11, (3, 3, 3))

print("3D array of shape (3, 3, 3) with random integers between 1 and 10:\n", array_3d)

# Find the indices of the maximum values along each depth level (third axis)
max_indices = np.argmax(array_3d, axis=2)

print("\nIndices of the maximum values along each depth level:\n", max_indices)
```

```
3D array of shape (3, 3, 3) with random integers between 1 and 10:
 [[[ 5 10  1]
  [ 7 10  3]
  [ 5  8  4]]

 [[ 1  6  5]
  [ 1  3  4]
  [ 2  8  2]]

 [[ 4  5  2]
  [ 8  5  1]
  [ 3  8  5]]]

Indices of the maximum values along each depth level:
 [[1 1 1]
 [1 2 1]
 [1 0 1]]
```

In [155]:
```
#b) Perform element-wise multiplication of between both array.
import numpy as np

# Create a 3D array of shape (3, 3, 3) with random integers between 1 and 10
array_3d = np.random.randint(1, 11, (3, 3, 3))

# Create another 3D array with the same shape
another_array_3d = np.random.randint(1, 11, (3, 3, 3))

print("Original 3D array:\n", array_3d)
print("\nAnother 3D array:\n", another_array_3d)

# Perform element-wise multiplication between both arrays
elementwise_multiplication = array_3d * another_array_3d

print("\nElement-wise multiplication result:\n", elementwise_multiplication)
```

```
Original 3D array:
 [[[ 1  3  7]
  [10  3  5]
  [10 10  6]]

 [[ 5  5 10]
  [ 9  2  6]
  [ 8  1  2]]

 [[ 4 10  3]
  [ 9  3  5]
  [ 9  3 10]]]

Another 3D array:
 [[[ 9  8  9]
  [ 3  4  4]
  [ 7  1  4]]

 [[ 7  4 10]
  [ 7  4  3]
  [ 3  3  7]]

 [[ 5  7  1]
  [ 5 10  3]
  [ 2  7  2]]]

Element-wise multiplication result:
 [[[  9  24  63]
  [ 30  12  20]
  [ 70  10  24]]

 [[ 35  20 100]
  [ 63   8  18]
  [ 24   3  14]]

 [[ 20  70   3]
  [ 45  30  15]
  [ 18  21  20]]]
```

In [156]:
```
#7. Clean and transform the 'Phone' column in the sample dataset to remove non-numeric chara
#convert it to a numeric data type. Also display the table attributes and data types of eacl

import pandas as pd

df = pd.read_csv('People Data.csv')
```

In [157]:
```
df.head()
```

Out[157]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Prob
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-1931	D
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	(599)782-0605	25-11-2013	
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	NaN	17-11-2012	Couns psycho
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	(390)417-1635x3010	15-04-1923	Biom eng

In [158]:
```
# Clean and transform the 'Phone' column
df['Phone'] = df['Phone'].str.replace(r'\D', '').astype(float)

# Display the table attributes and data types of each column
print("Table 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
```

In [159]:
```
# Display the cleaned DataFrame
print("\nCleaned DataFrame:")
print(df)
```

```
Cleaned DataFrame:
     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]
```

In [160]:
```
#8. Perform the following tasks using people dataset:
```

In [161]:
```
#a) Read the 'data.csv' file using pandas, skipping the first 50 rows
import pandas as pd

df = pd.read_csv('People Data.csv', skiprows=50)
df.head()
```

Out[161]:

	50	afF3018e9cdd1dA	George	Mercer	Female	douglascontreras@example.net	+1-326-669-0118x4341	11-09-1941	I res
0	51	CccE5DAb6E288e5	Jo	Zavala	Male	pamela64@example.net	001-859-448-9935x54536	23-11-1992	
1	52	DfBDc3621D4bcec	Joshua	Carey	Female	dianashepherd@example.net	001-274-739-8470x814	07-01-1915	S inte
2	53	f55b0A249f5E44D	Rickey	Hobbs	Female	ingramtiffany@example.org	241.179.9509x498	01-07-1910	B
3	54	Ed71DcfaBFd0beE	Robyn	Reilly	Male	carriecrawford@example.org	207.797.8345x6177	27-07-1982	Er str
4	55	FDaFD0c3f5387EC	Christina	Conrad	Male	fuentesclaudia@example.net	001-599-042-7428x143	06-01-1998	Pr

In [162]:
```
import pandas as pd

df = pd.read_csv('People Data.csv')
df['Salary']
```

Out[162]:
```
0       90000
1       80000
2       50000
3       65000
4      100000
        ...
995     90000
996     50000
997     60000
998    100000
999     90000
Name: Salary, Length: 1000, dtype: int64
```

In [163]:
```
#a) Read the 'data.csv' file using pandas, skipping the first 50 rows
import pandas as pd

df = pd.read_csv('People Data.csv', skiprows=50)
df.head()
```

Out[163]:

	50	afF3018e9cdd1dA	George	Mercer	Female	douglascontreras@example.net	+1-326-669-0118x4341	11-09-1941	I res
0	51	CccE5DAb6E288e5	Jo	Zavala	Male	pamela64@example.net	001-859-448-9935x54536	23-11-1992	
1	52	DfBDc3621D4bcec	Joshua	Carey	Female	dianashepherd@example.net	001-274-739-8470x814	07-01-1915	S inte
2	53	f55b0A249f5E44D	Rickey	Hobbs	Female	ingramtiffany@example.org	241.179.9509x498	01-07-1910	B
3	54	Ed71DcfaBFd0beE	Robyn	Reilly	Male	carriecrawford@example.org	207.797.8345x6177	27-07-1982	Er str
4	55	FDaFD0c3f5387EC	Christina	Conrad	Male	fuentesclaudia@example.net	001-599-042-7428x143	06-01-1998	Pr

In [164]:
```
#b) Only read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and ‘Salary’ from the file
import pandas as pd

# Read the CSV file with specific columns
df = pd.read_csv('People Data.csv', usecols=['Last Name', 'Gender', 'Email', 'Phone', 'Sala

# Display the first few rows of the DataFrame
print(df.head())
```

```
  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
```

In [165]:
```
#c) Display the first 10 rows of the filtered dataset.

print(df.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
```

In [166]:
```
#d) Extract the ‘Salary’' column as a Series and display its last 5 values.
# Extract the 'Salary' column as a Series
salary_series = df['Salary']

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

```
995     90000
996     50000
997     60000
998    100000
999     90000
Name: Salary, dtype: int64
```

In [167]:
```
#9. Filter and select rows from the People_Dataset, where the 'Last Name' column contains tl
#'Gender' column contains the word Female and ‘Salary’ should be less than 85000.
import pandas as pd

# Read the People_Dataset
df = pd.read_csv('People Data.csv')

# Filter and select rows based on conditions
filtered_df = df[(df['Last Name'].str.contains('Duke')) &
                 (df['Gender'] == 'Female') &
                 (df['Salary'] < 85000)]

# Display the filtered DataFrame
print(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
```

In [168]:
```
#10. Create a 7*5 Dataframe in Pandas using a series generated from 35 random integers betwe
import pandas as pd
import numpy as np

# Generate 35 random integers between 1 and 6
random_integers = np.random.randint(1, 7, 35)

# Reshape the random integers into a 7x5 array
reshaped_array = random_integers.reshape(7, 5)

# Create DataFrame from the reshaped array
df = pd.DataFrame(reshaped_array)

# Display the DataFrame
print(df)
```

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

In [169]:
```
#11. Create two different Series, each of length 50, with the following criteria:
```

In [170]:
```
#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 numbers ranging from 10 to 50
series1 = pd.Series(np.random.randint(10, 51, 50))

# Display the first few elements of the Series
print(series1.head())
```

```
0    50
1    36
2    40
3    15
4    25
dtype: int32
```

In [171]:
```
#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 numbers ranging from 100 to 1000
series2 = pd.Series(np.random.randint(100, 1001, 50))

# Display the first few elements of the Series
print(series2.head())
```

```
0    1000
1     328
2     939
3     418
4     891
dtype: int32
```

In [172]:
```
#c) Create a DataFrame by joining these Series by column, and, change the names of the colu
import pandas as pd
import numpy as np

# Create the first Series with random numbers ranging from 10 to 50
series1 = pd.Series(np.random.randint(10, 51, 50))

# Create the second Series with random numbers ranging from 100 to 1000
series2 = pd.Series(np.random.randint(100, 1001, 50))

# Concatenate the Series along the columns axis to create a DataFrame
df = pd.concat([series1, series2], axis=1)

# Change the column names to 'col1', 'col2', etc.
df.columns = [f'col{i}' for i in range(1, df.shape[1] + 1)]

# Display the DataFrame
print(df.head())
```

```
   col1  col2
0    29   610
1    14   614
2    30   364
3    20   366
4    13   361
```

In [173]:
```
#12. Perform the following operations using people data set:
```

In [174]:
```
#a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.

# Delete the specified columns
columns_to_drop = ['Email', 'Phone', 'Date of birth']

# Filter out columns that are present in the DataFrame
columns_to_drop = [col for col in columns_to_drop if col in df.columns]

# Drop the specified columns
df = df.drop(columns=columns_to_drop)

# Display the modified DataFrame
print(df.head())
```

```
   col1  col2
0    29   610
1    14   614
2    30   364
3    20   366
4    13   361
```

In [175]:
```
#b) Delete the rows containing any missing values.

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

# Display the modified DataFrame
print(df.head())
```

```
   col1  col2
0    29   610
1    14   614
2    30   364
3    20   366
4    13   361
```

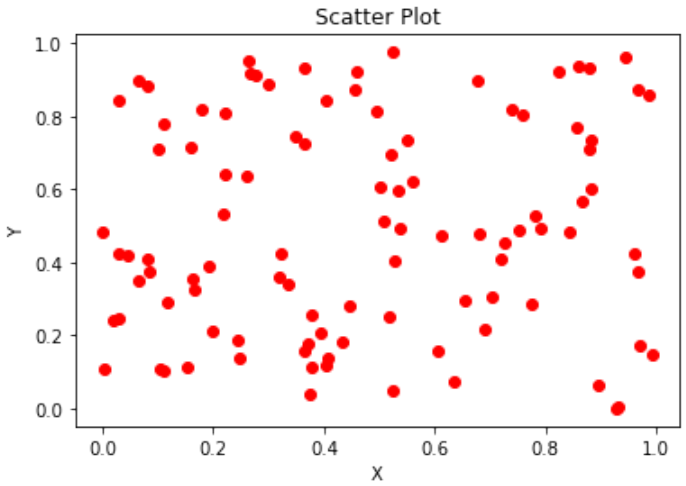
In [176]:
```
#c.Print the final output also
df.head()
```

Out[176]:

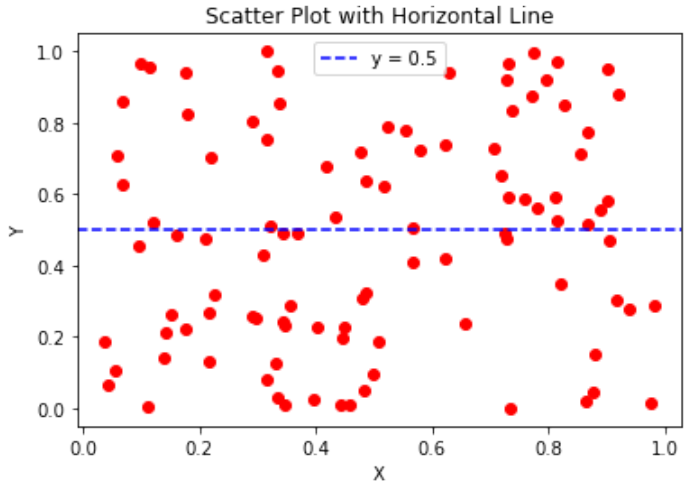
	col1	col2
0	29	610
1	14	614
2	30	364
3	20	366
4	13	361

In [177]:
```
#13. Create two NumPy arrays, x and y, each containing 100 random float values between 0 and
#following tasks using Matplotlib and NumPy:
```

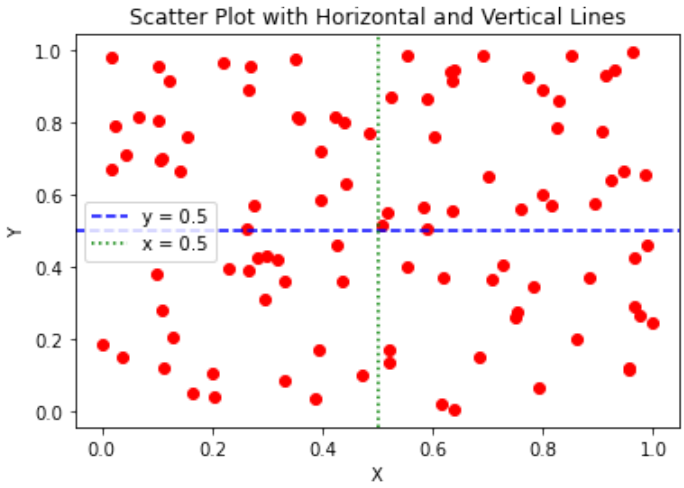
In [178]:
```
#a) Create a scatter pot using x and y, setting the color of the points to red and the marke
import numpy as np
import matplotlib.pyplot as plt

# Generate random float values for x and y arrays
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
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter Plot')

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



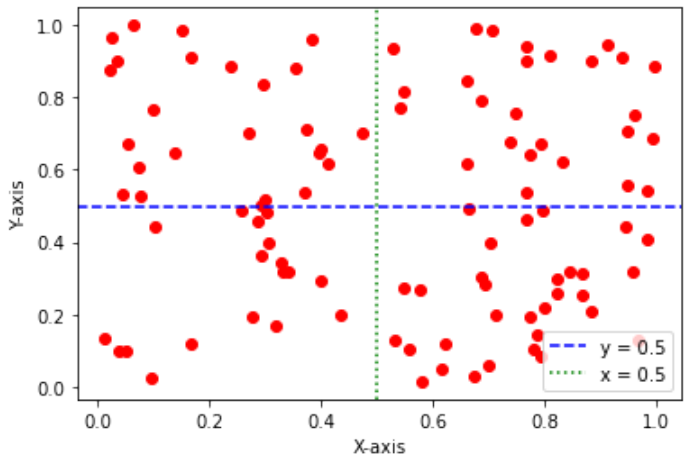
In [179]:
```
#b) Add a horizontal line at y = 0.5 using a dashed line style and label it as 'y = 0.5
import numpy as np
import matplotlib.pyplot as plt

# Generate random float values for x and y arrays
x = np.random.rand(100)
y = np.random.rand(100)

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

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

# Add labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter Plot with Horizontal Line')

# Add legend
plt.legend()

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



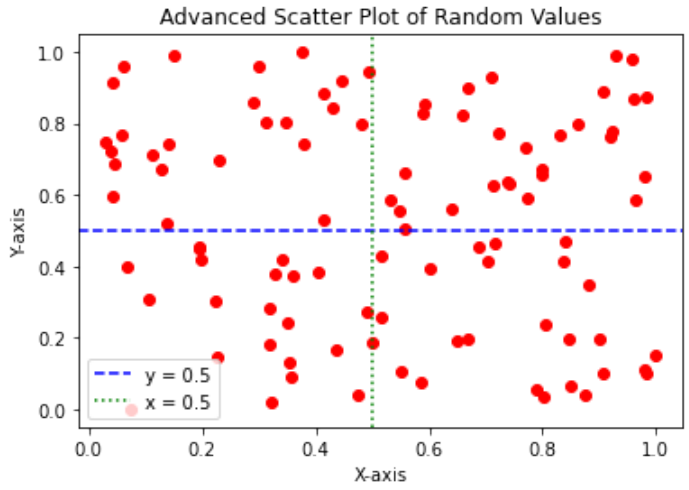
In [180]:
```
#c) Add a vertical line at x = 0.5 using a dotted line style and label it as 'x = 0.5'.
import numpy as np
import matplotlib.pyplot as plt

# Generate random float values for x and y arrays
x = np.random.rand(100)
y = np.random.rand(100)

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

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='blue', 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 labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Scatter Plot with Horizontal and Vertical Lines')

# Add legend
plt.legend()

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



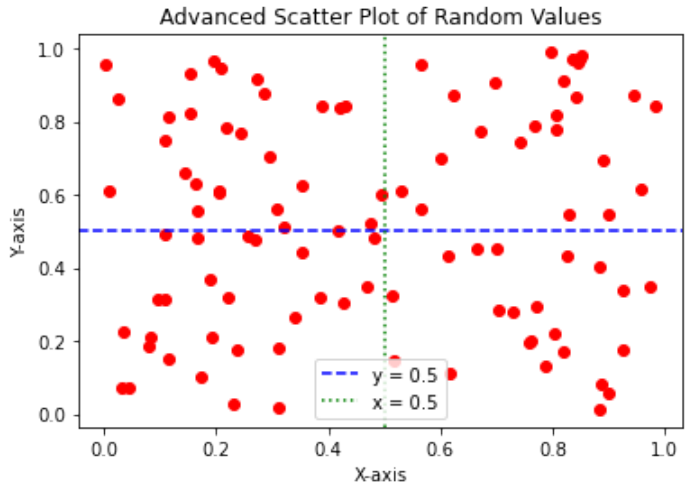
In [181]:
```
#d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.
import numpy as np
import matplotlib.pyplot as plt

# Generate random float values for x and y arrays
x = np.random.rand(100)
y = np.random.rand(100)

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

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='blue', 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 labels for x-axis and y-axis
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Add legend
plt.legend()

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



In [182]:
```
#e) Set the title of the plot as 'Advanced Scatter Plot of Random Values'.
import numpy as np
import matplotlib.pyplot as plt

# Generate random float values for x and y arrays
x = np.random.rand(100)
y = np.random.rand(100)

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

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='blue', 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 labels for x-axis and y-axis
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Add title to the plot
plt.title('Advanced Scatter Plot of Random Values')

# Add legend
plt.legend()

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

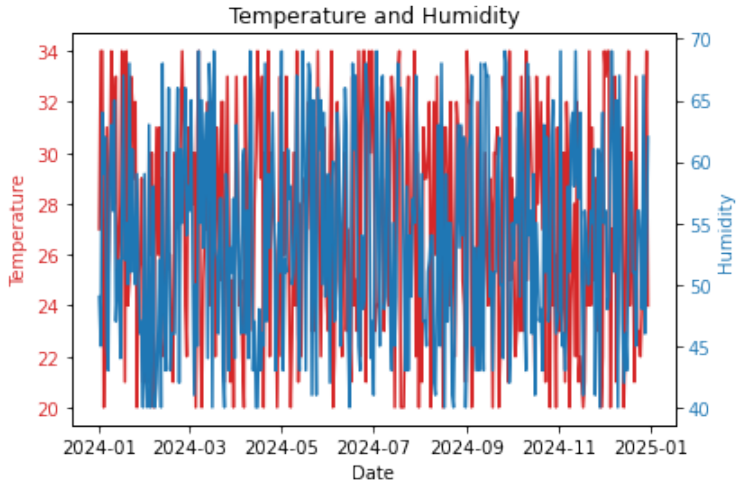


In [183]:
```
#f) Display a legend for the scatter plot, the horizontal line, and the vertical line
import numpy as np
import matplotlib.pyplot as plt

# Generate random float values for x and y arrays
x = np.random.rand(100)
y = np.random.rand(100)

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

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='blue', 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 labels for x-axis and y-axis
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Add title to the plot
plt.title('Advanced Scatter Plot of Random Values')

# Add legend
plt.legend()

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



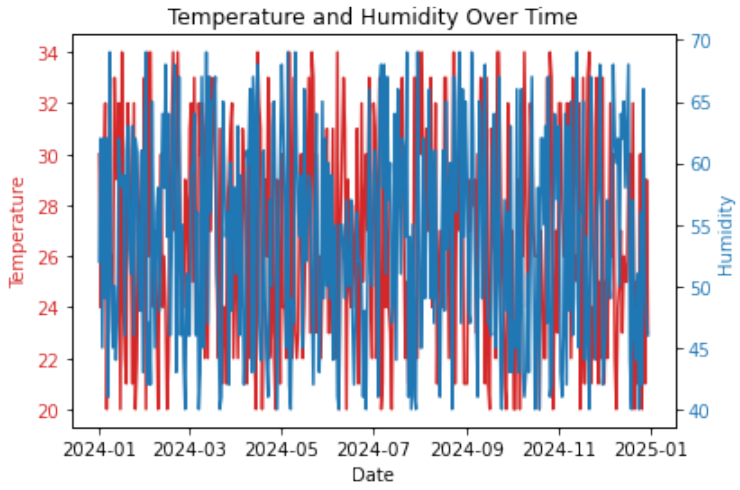
In [184]:
```
#14. Create a time-series dataset in a Pandas DataFrame with columns: 'Date', 'Temperature',
#Perform the following tasks using Matplotlib:
```

In [185]:
```
#a) Plot the 'Temperature' and 'Humidity' on the same plot with different y-axes (left y-ax
#right y-axis for 'Humidity').
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a time-series dataset in a Pandas DataFrame
date_range = pd.date_range(start='2024-01-01', periods=365)
temperature = np.random.randint(20, 35, size=365)
humidity = np.random.randint(40, 70, size=365)

data = {'Date': date_range, 'Temperature': temperature, 'Humidity': humidity}
df = pd.DataFrame(data)

# Plot 'Temperature' and 'Humidity' on the same plot with different y-axes
fig, ax1 = plt.subplots()

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

# Create a second y-axis for Humidity
ax2 = ax1.twinx()

# Plot Humidity on the secondary y-axis
color = 'tab:blue'
ax2.set_ylabel('Humidity', color=color)
ax2.plot(df['Date'], df['Humidity'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

# Add a title to the plot
plt.title('Temperature and Humidity')

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



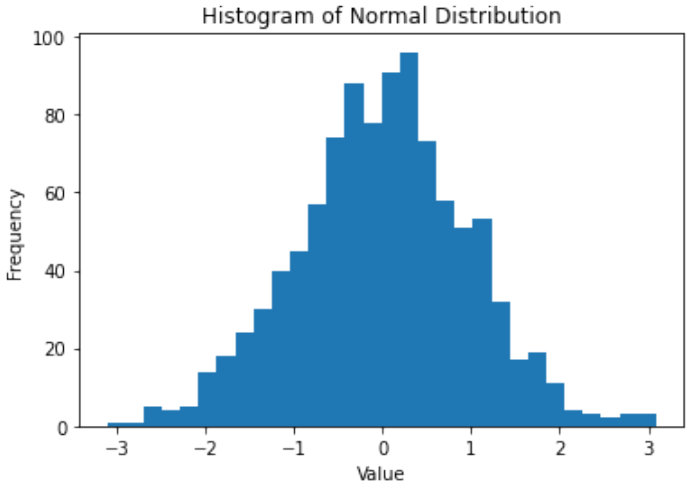
In [186]:
```
#b) Label the x-axis as 'Date'.
```

In [187]:
```
#c) Set the title of the plot as 'Temperature and Humidity Over Time'.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a time-series dataset in a Pandas DataFrame
date_range = pd.date_range(start='2024-01-01', periods=365)
temperature = np.random.randint(20, 35, size=365)
humidity = np.random.randint(40, 70, size=365)

data = {'Date': date_range, 'Temperature': temperature, 'Humidity': humidity}
df = pd.DataFrame(data)

# Plot 'Temperature' and 'Humidity' on the same plot with different y-axes
fig, ax1 = plt.subplots()

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

# Create a second y-axis for Humidity
ax2 = ax1.twinx()

# Plot Humidity on the secondary y-axis
color = 'tab:blue'
ax2.set_ylabel('Humidity', color=color)
ax2.plot(df['Date'], df['Humidity'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

# Add a title to the plot
plt.title('Temperature and Humidity Over Time')

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

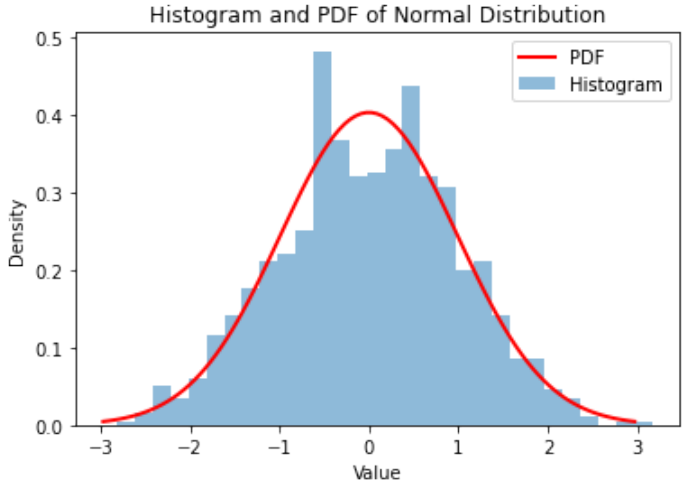


In [188]:
```
#15. Create a NumPy array data containing 1000 samples from a normal distribution.
#Perform the following tasks using Matplotlib:
```

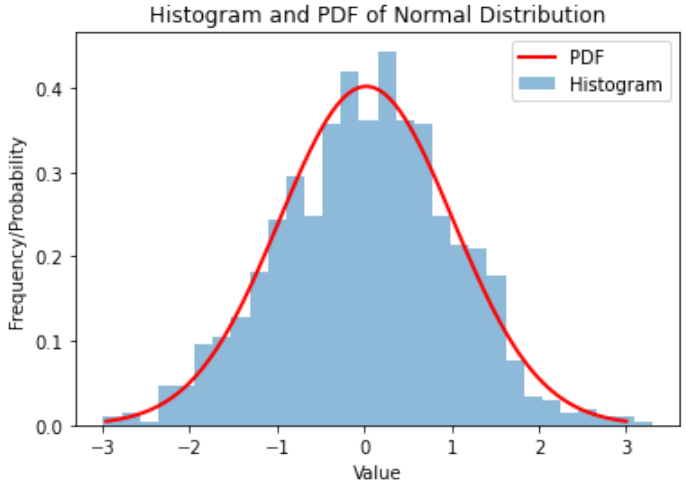
In [189]:
```
#a) Plot a histogram of the data with 30 bins.
import numpy as np
import matplotlib.pyplot as plt

# Generate NumPy array data containing 1000 samples from a normal distribution
data = np.random.normal(size=1000)

# Plot a histogram of the data with 30 bins
plt.hist(data, bins=30)

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Normal Distribution')

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



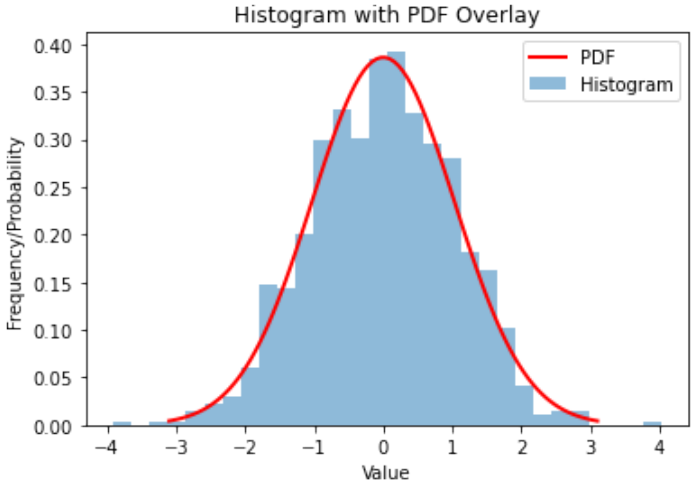
In [190]:
```
#b) Overlay a line plot representing the normal distribution's probability density function
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Generate NumPy array data containing 1000 samples from a normal distribution
data = np.random.normal(size=1000)

# Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.5, label='Histogram')

# Compute the PDF values for the normal distribution
mu, sigma = np.mean(data), np.std(data)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
pdf = norm.pdf(x, mu, sigma)

# Overlay a line plot representing the normal distribution's PDF
plt.plot(x, pdf, color='red', linewidth=2, label='PDF')

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

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



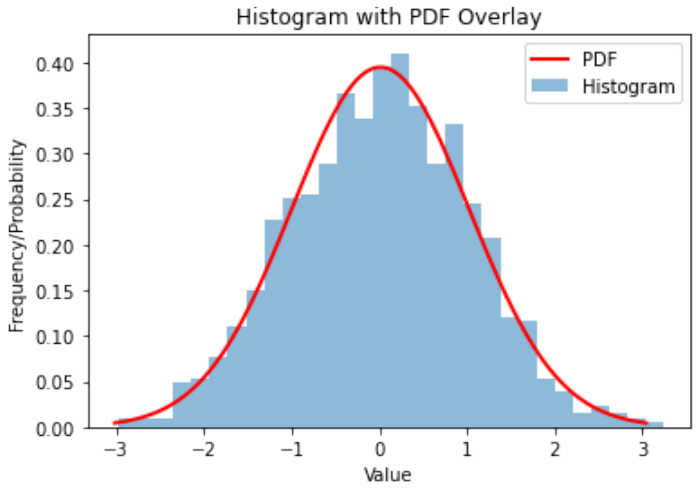
In [191]:
```
#c) Label the x-axis as 'Value' and the y-axis as 'Frequency/Probability'.
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Generate NumPy array data containing 1000 samples from a normal distribution
data = np.random.normal(size=1000)

# Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.5, label='Histogram')

# Compute the PDF values for the normal distribution
mu, sigma = np.mean(data), np.std(data)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
pdf = norm.pdf(x, mu, sigma)

# Overlay a line plot representing the normal distribution's PDF
plt.plot(x, pdf, color='red', linewidth=2, label='PDF')

# Add labels, title, and legend
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram and PDF of Normal Distribution')
plt.legend()

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



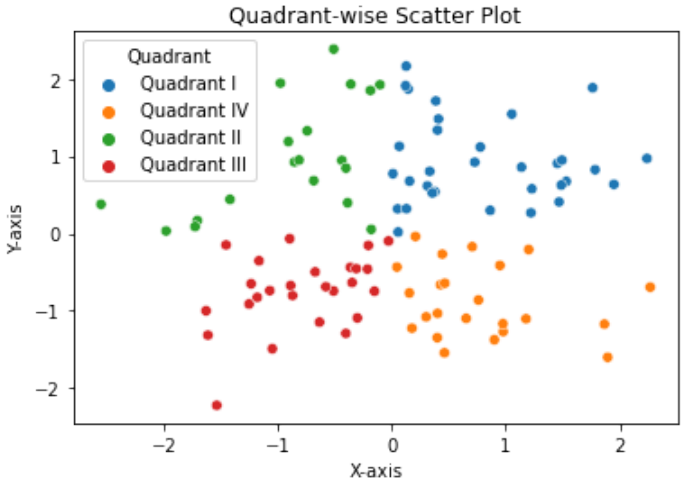
In [192]:
```python
#d) Set the title of the plot as 'Histogram with PDF Overlay'.
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Generate NumPy array data containing 1000 samples from a normal distribution
data = np.random.normal(size=1000)

# Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.5, label='Histogram')

# Compute the PDF values for the normal distribution
mu, sigma = np.mean(data), np.std(data)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
pdf = norm.pdf(x, mu, sigma)

# Overlay a line plot representing the normal distribution's PDF
plt.plot(x, pdf, color='red', linewidth=2, label='PDF')

# Add labels, title, and legend
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram with PDF Overlay')
plt.legend()

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



In [193]:
```
#16. Set the title of the plot as 'Histogram with PDF Overlay'.
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Generate NumPy array data containing 1000 samples from a normal distribution
data = np.random.normal(size=1000)

# Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.5, label='Histogram')

# Compute the PDF values for the normal distribution
mu, sigma = np.mean(data), np.std(data)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100)
pdf = norm.pdf(x, mu, sigma)

# Overlay a line plot representing the normal distribution's PDF
plt.plot(x, pdf, color='red', linewidth=2, label='PDF')

# Add labels, title, and legend
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram with PDF Overlay')

# Add legend
plt.legend()

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



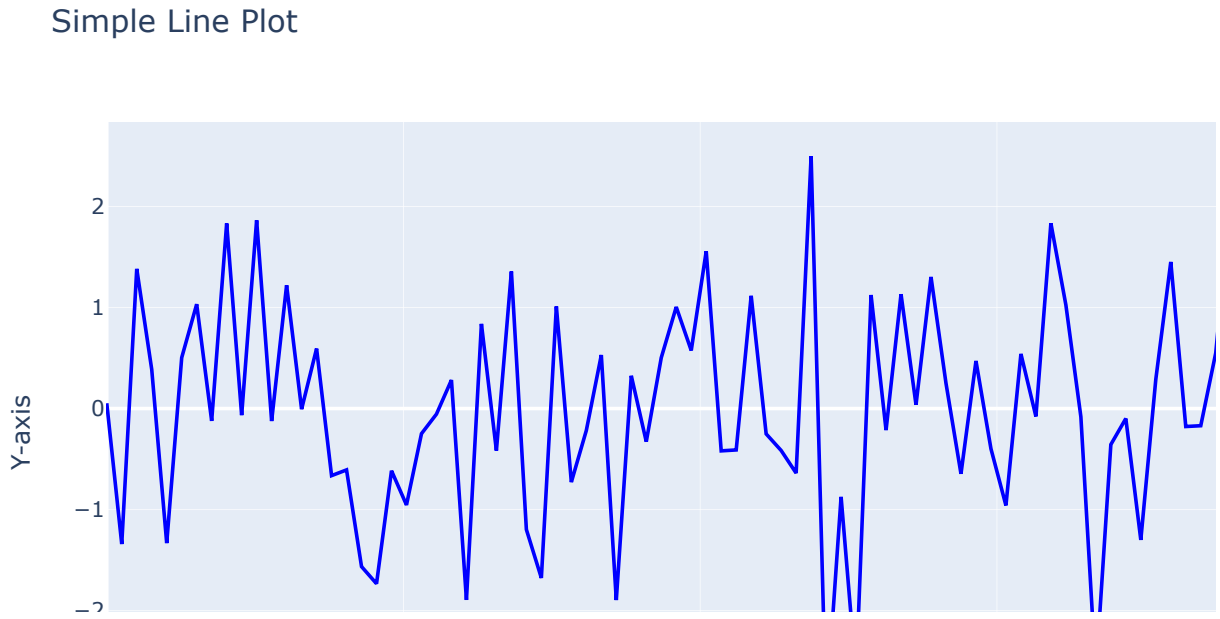
In [194]:
```
#17. Create a Seaborn scatter plot of two random arrays, color points based on their positi
#origin (quadrants), add a legend, label the axes, and set the title as 'Quadrant-wise Scat
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Generate two random arrays
np.random.seed(0)
x = np.random.randn(100)
y = np.random.randn(100)

# Determine the quadrant for each point
quadrant = []
for xi, yi in zip(x, y):
    if xi >= 0 and yi >= 0:
        quadrant.append('Quadrant I')
    elif xi < 0 and yi >= 0:
        quadrant.append('Quadrant II')
    elif xi < 0 and yi < 0:
        quadrant.append('Quadrant III')
    else:
        quadrant.append('Quadrant IV')

# Create the Seaborn scatter plot
sns.scatterplot(x=x, y=y, hue=quadrant, palette='tab10')

# Add labels for the axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Add a title to the plot
plt.title('Quadrant-wise Scatter Plot')

# Add legend
plt.legend(title='Quadrant')

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



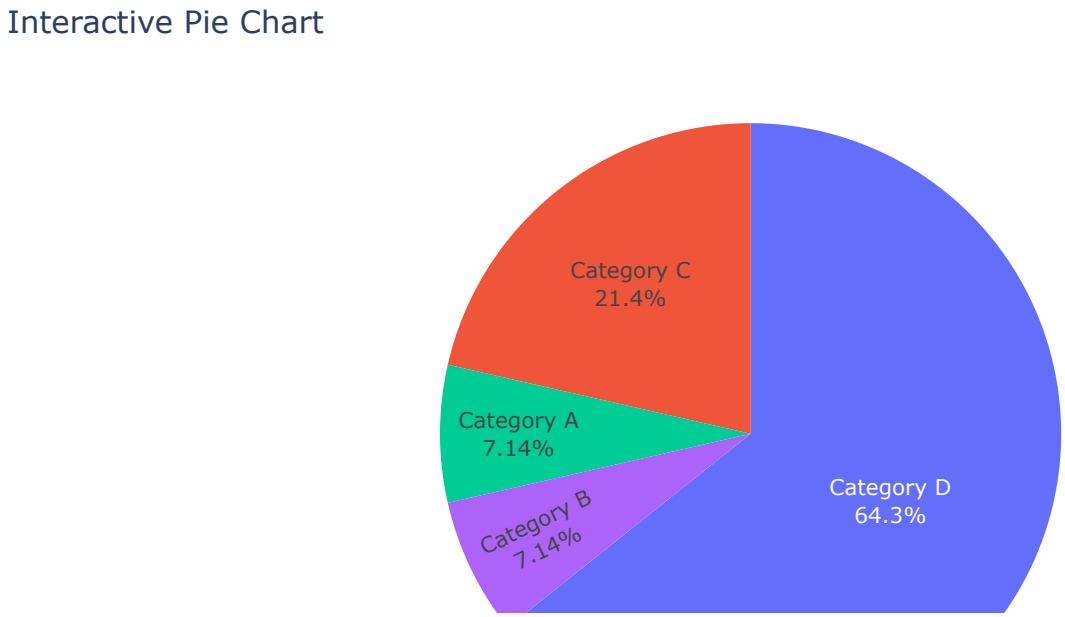
In [195]:
```
#18 With Bokeh, plot a line chart of a sine wave function, add grid lines, label the axes, 
from bokeh.plotting import figure, show
import numpy as np

# Generate x values
x = np.linspace(0, 2*np.pi, 100)

# Generate y values for the sine wave function
y = np.sin(x)

# Create a new plot with title and axis labels
p = figure(title='Sine Wave Function', x_axis_label='X', y_axis_label='Y')

# Add a line renderer with legend and line thickness
p.line(x, y, legend_label='sin(x)', line_width=2)

# Add grid lines
p.grid.grid_line_alpha = 0.3

# Show the plot
show(p)
```

In [196]:
```
#19 Using Bokeh, generate a bar chart of randomly generated categorical data, color bars bas
#values, add hover tooltips to display exact values, label the axes, and set the title as '
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, HoverTool
import numpy as np

# Generate random categorical data and their corresponding values
categories = ['A', 'B', 'C', 'D']
values = np.random.randint(1, 10, size=len(categories))

# Create a new plot with title and axis labels
p = figure(title='Random Categorical Bar Chart', x_axis_label='Category', y_axis_label='Val

# Create a ColumnDataSource
source = ColumnDataSource(data=dict(categories=categories, values=values))

# Add a vbar glyph to the plot
p.vbar(x='categories', top='values', width=0.5, color='navy', legend_label='values', source

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

# Label the axes
p.xaxis.major_label_orientation = 1
p.xaxis.axis_label_text_font_size = "12pt"
p.yaxis.axis_label_text_font_size = "12pt"

# Show the plot
show(p)
```

In [197]:
```
#20 Using Plotly, create a basic line plot of a randomly generated dataset, label the axes,
import plotly.graph_objs as go
import numpy as np

# Generate random data for x and y axes
x = np.linspace(0, 10, 100)
y = np.random.randn(100)

# Create a trace for the line plot
trace = go.Scatter(
    x=x,
    y=y,
    mode='lines',
    line=dict(color='blue', width=2),
    name='Random Data'
)

# Create layout with axis labels and title
layout = go.Layout(
    title='Simple Line Plot',
    xaxis=dict(title='X-axis'),
    yaxis=dict(title='Y-axis')
)

# Create figure object
fig = go.Figure(data=[trace], layout=layout)

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



In [198]:
```
#21 Using Plotly, create an interactive pie chart of randomly generated data, add labels and
import plotly.graph_objs as go
import numpy as np

# Generate random data for the pie chart
labels = ['Category A', 'Category B', 'Category C', 'Category D']
values = np.random.randint(1, 10, size=len(labels))

# Create a trace for the pie chart
trace = go.Pie(
    labels=labels,
    values=values,
    hoverinfo='label+percent',
    textinfo='label+percent'
)

# Create layout with title
layout = go.Layout(
    title='Interactive Pie Chart'
)

# Create figure object
fig = go.Figure(data=[trace], layout=layout)

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



In [ ]:

In [ ]:

In [ ]:

In [ ]: