```
#DATA TOOLKIT ASSIGNMENT SUBMISSION


#QUESTION 1- DEMONSTRATE THRE  DIFF METHODS FOR CREATING IDENTICAL ......
#METHOD1
import numpy as np
array1=np.array([[1,2,3],[4,5,6]])
print("method1:\n",array1)
```

```
method1:
 [[1 2 3]
 [4 5 6]]
```

```
#method 2
array2=np.zeros((2,3),dtype=int)
array2[0]=[1,2,3]
array2[1]=[4,5,6]
print("method2:\n",array2)
```

```
method2:
 [[1 2 3]
 [4 5 6]]
```

```
#method 3
array3=np.arange(1,7).reshape(2,3)
print("method3:\n",array3)
```

```
method3:
 [[1 2 3]
 [4 5 6]]
```

```
#QUESTION-2 USING A NUMPY FUNCTION.....
import numpy as np
array_1d=np.linspace(1,10,100)
array_2d=array_1d.reshape(10,10)
print(array_2d)
```

```
[[ 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.        ]]
```

```
#QUESTION3- DIFFERENCE BETWEEN.....,
#np.array:
#this function creates a new numpy array from any object exposing the array interface
#it always creates a new array regardless of the input types
#np.asarray:
#this function also creates an array bit it will return the input as numpy array if the input is already an array
#it does not create a copy if the input is already a numpy array which can save memory
#np.asanyarray:
#this function is similar to np.asarray but it allows the input to be of any array like type,including subclasses
#it will return a base array if the input is a subclass of np.ndarray but will not create a copy of it
#if the input is already a numpy array it will not create a copy


#QUESTION NO 4 GENERATE A 3X3 ARRAY.........?
import numpy as np
random_array=np.random.uniform(5,20,(3,3))
rounded_array=np.round(random_array,2)
print("random 3x3 array:\n",random_array)
print("\n rounded 3x3 array:\n",rounded_array)
```

```
random 3x3 array:
 [[18.73753976  9.54853036 14.26953247]
 [11.93887817 14.14827465  8.14052923]
 [17.54749288 11.67348763 12.57242276]]
```

```
 rounded 3x3 array:
 [[18.74  9.55 14.27]
 [11.94 14.15  8.14]
 [17.55 11.67 12.57]]
```

```
#QUESTION NO 5:
import numpy as np
random_array=np.random.randint(1,11,size=(5,6))
print("original array:\n",random_array)
#a)extract all even integers from array
even_integers=random_array[random_array%2==0]
print("\neven integers:\n",even_integers)
#b extract all odd integers from the array
odd_integers=random_array[random_array%2!=0]
print("\nodd integers:\n",odd_integers)
```

```
original array:
 [[ 2  2  1  5  9  2]
 [ 9  1  3  9 10  9]
 [ 4  3  1  2  6  8]
 [ 6  9  2  5  5  7]
 [ 9  6  5  2 10  7]]

even integers:
 [ 2  2  2 10  4  2  6  8  6  2  6  2 10]

odd integers:
 [1 5 9 9 1 3 9 9 3 1 9 5 5 7 9 5 7]
```

```
#question 6-CREATE A 3D NUMPY ARRAY OF SHAPE (3,3,3).....?
import numpy as np
random_3d_array=np.random.randint(1,11,size=(3,3,3))
print("original 3d array:\n",random_3d_array)
#a)find the indices of the maximum values
max_indices=np.argmax(random_3d_array,axis=2)
print("\nindices of maximum values along depth levels:\n",max_indices)
#B)perform element wise mulltiplication of the original
multiplied_array=random_3d_array*random_3d_array
print("\nelement wise multiplication:\n",multiplied_array)
```

```
original 3d array:
 [[[ 4  4  1]
  [ 9  8  8]
  [ 4  8  7]]

 [[ 8  5  7]
  [ 6 10  8]
  [ 4  4  3]]

 [[ 4  4  1]
  [ 8  2  5]
  [ 1 10  8]]]

indices of maximum values along depth levels:
 [[0 0 1]
 [0 1 0]
 [0 0 1]]

element wise multiplication:
 [[[ 16  16   1]
  [ 81  64  64]
  [ 16  64  49]]

 [[ 64  25  49]
  [ 36 100  64]
  [ 16  16   9]]

 [[ 16  16   1]
  [ 64   4  25]
  [  1 100  64]]]
```

```
#QUESTION NO 7:clean and transform the phone.......?
import pandas as pd
data={
    'name':['alice','bob','charlie'],
    'phone':['(123) 456-7890','987-654-3210','555.123.4567'],
    'age':[25,30,35]

}
df=pd.DataFrame(data)
print("original dataframe:\n",df)
df['phone']=df['phone'].replace(r'\D','',regex=True).astype(float)
```

```
print("\ncleaned dataframe:\n,df")
print("\ndataframe attributes and data types:")
print(df.dtypes)
```

```
original dataframe:
      name           phone  age
0    alice  (123) 456-7890   25
1      bob    987-654-3210   30
2  charlie    555.123.4567   35

cleaned dataframe:
,df

dataframe attributes and data types:
name      object
phone    float64
age        int64
dtype: object
```

```
#question no 8 perform the following tasks using people dataset
import pandas as pd
#a)read the 'data.csv' file skipping the first 50 rows
df=pd.read_csv()
#b) only read the specified columns
df_filtered=df[['last name','gender','email','phone','salary']]
#c) display the first  10 rows
print("first 10 rows of the filtered dataset:")
print(df_filtered.head(10))
#D)extract the salary  column as a series and display
salary_series=df_filtered['salary']
print("\nlast 5 values of the 'salary' column:")
print(salary_series.tail(5))
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-14-e1c9d0d9ad3c> in <cell line: 4>()
      2 import pandas as pd
      3 #a)read the 'data.csv' file skipping the first 50 rows
----> 4 df=pd.read_csv('data.csv',skiprows=50)
      5 #b) only read the specified columns
      6 df_filtered=df[['last name','gender','email','phone','salary']]

4 frames
/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    871         if ioargs.encoding and "b" not in ioargs.mode:
    872             # Encoding
--> 873             handle = open(
    874                 handle,
    875                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: 'data.csv'
```

Next steps: Explain error

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/People Data (1).csv')
df.head()
```

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	Salary
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Probation officer	90000
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-1931	Dancer	80000

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#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
import pandas as pd
```

```
people_dataset=pd.read_csv('/content/People Data (1).csv')
#fliter the dataset
filtered_data=people_dataset[
    (people_dataset['Last Name'].str.contains('Duke',case=False))&
    (people_dataset['Gender'].str.contains('female',case=False))&
    (people_dataset['Salary']<85000)
]
print(filtered_data)
```

```
      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?
import pandas as pd
import numpy as np
random_integers=np.random.randint(1,7,size=35)
df=pd.DataFrame(random_integers.reshape(7,5),columns=['col1','col1','col3','col4','col5'])
print(df)
```

```
   col1  col1  col3  col4  col5
0     1     2     3     4     1
1     1     3     2     4     6
2     2     4     3     3     4
3     3     1     5     2     1
4     3     4     4     3     6
5     4     1     4     4     4
6     2     2     3     5     6
```

```
#11)KKg 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',

import numpy as np
import pandas as pd
series1=pd.Series(np.random.randint(10,51,size=50))
series2=pd.Series(np.random.randint(100,1001,size=50))
df=pd.DataFrame({'col1':series1,'col2':series2})
print(df)
```

```
    col1  col2
0     33   927
1     43   708
2     40   620
3     44   661
4     37   967
5     48   237
6     26   822
7     17   489
8     34   856
9     12   338
10    26   344
11    37   172
12    37   427
13    32   673
14    34   766
15    37   192
16    13   902
17    11   775
18    27   721
19    29   224
20    50   507
21    14   970
22    15   745
23    34   939
24    16   833
25    50   172
26    21   896
```

```
27    48   926
28    50   623
29    34   482
30    32   636
31    45   428
32    14   230
33    38   508
34    40   996
35    10   988
36    12   116
37    47   778
38    17   471
39    39   295
40    42   265
41    48   738
42    45   534
43    15   162
44    33   135
45    40   542
46    45   182
47    27   651
48    19   335
49    13   536
```

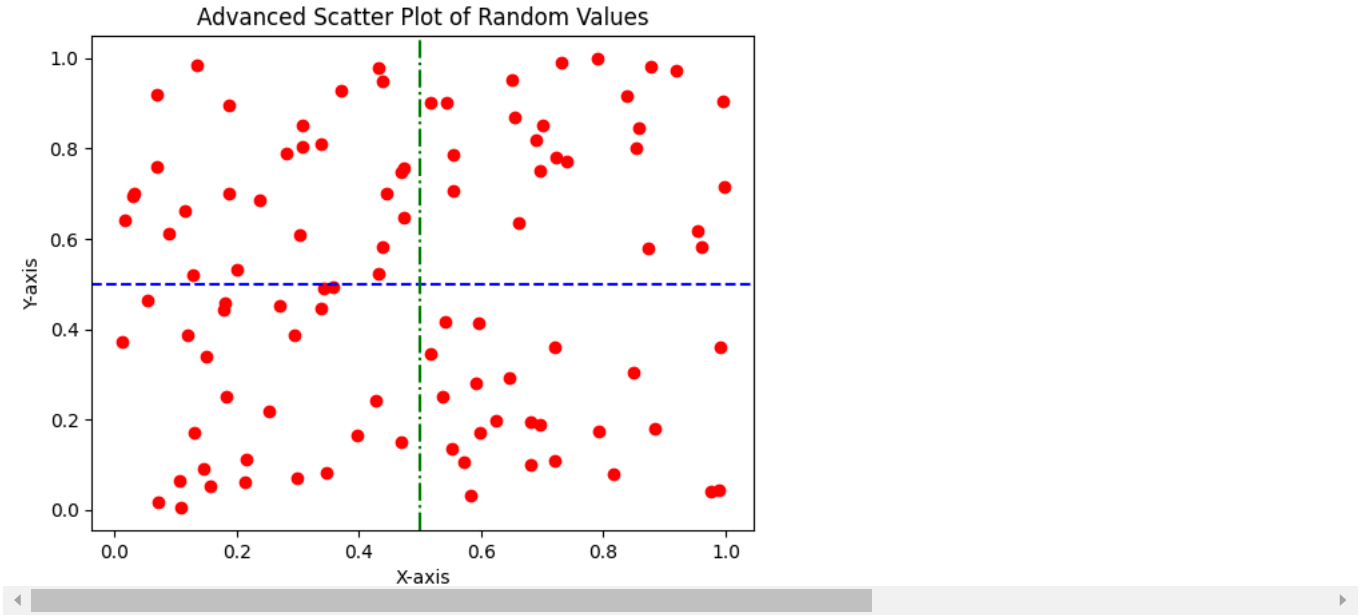
```
#12) Perform the following operations using people data set:

#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&
import pandas as pd
people_dataset=pd.read_csv('/content/People Data (1).csv')
people_dataset.drop(columns=['Email','Phone','Date of birth'],inplace=True)
people_dataset.dropna(inplace=True)
print(people_dataset)
```

```
      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.
import numpy as np
import matplotlib.pyplot as plt
x=np.random.rand(100)
y=np.random.rand(100)
#a)create a scatter plot
plt.scatter(x,y,color='red',marker='o',label='data points')
```

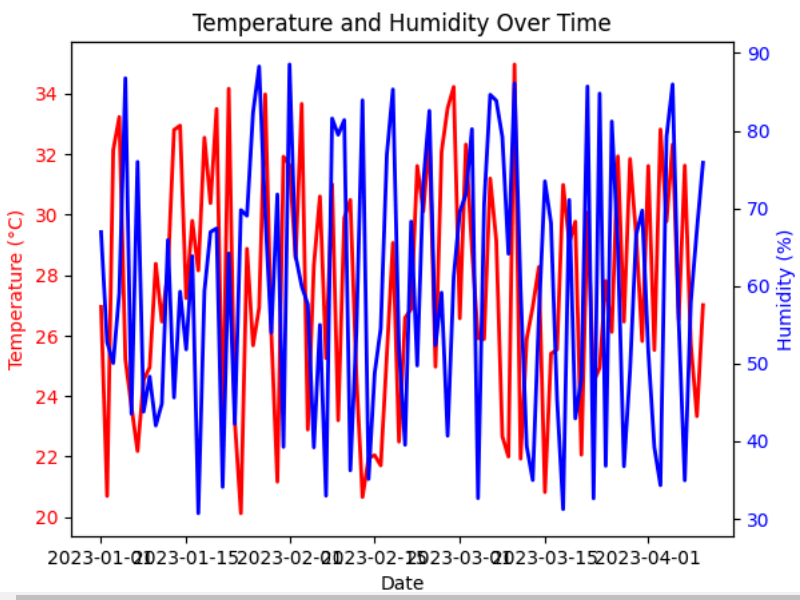
```
#b)adda horizontal line at y=0.5
plt.axhline(y=0.5,color='blue',linestyle='--',label='y=0.5')
#c add a vertical line at y=0.5
plt.axvline(x=0.5,color='green',linestyle='-.',label='x=0.5')
#d)label the x and y axis
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
#e)set the title of the plot
plt.title('Advanced Scatter Plot of Random Values')
```

```
Text(0.5, 1.0, 'Advanced Scatter Plot of Random Values')
```

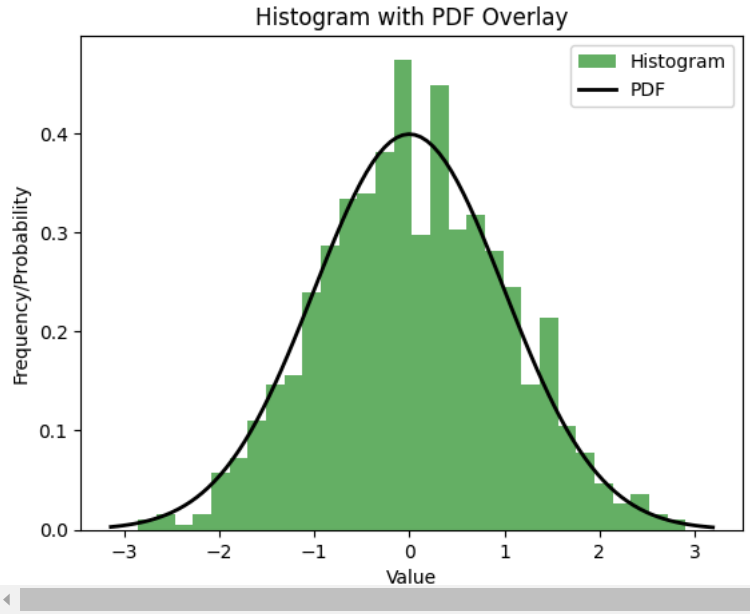



```
#Create a time-series dataset in a Pandas DataFrame with columns: 'Date', 'Temperature', 'Humidity' and
#Perform the following tasks using Matplotlib:



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

# Create a date range
date_range = pd.date_range(start='2023-01-01', periods=100, freq='D')

# Generate random temperature and humidity data
temperature = np.random.uniform(low=20, high=35, size=len(date_range))  # Random temperatures between 20 and 35
humidity = np.random.uniform(low=30, high=90, size=len(date_range))  # Random humidity between 30 and 90

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

# Set the 'Date' as the index
data.set_index('Date', inplace=True)

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

# Plot Temperature
ax1.plot(data.index, data['Temperature'], color='red', label='Temperature', linewidth=2)
ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature (°C)', color='red')
ax1.tick_params(axis='y', labelcolor='red')

# Create a second y-axis for Humidity
ax2 = ax1.twinx()
ax2.plot(data.index, data['Humidity'], color='blue', label='Humidity', linewidth=2)
ax2.set_ylabel('Humidity (%)', color='blue')
ax2.tick_params(axis='y', labelcolor='blue')
```

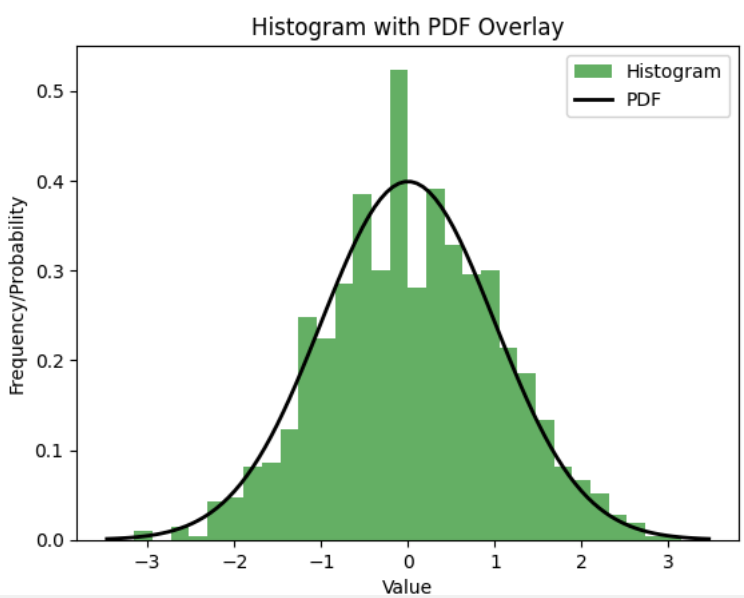
```
# b) Label the x-axis
plt.xlabel('Date')

# c) Set the title of the plot
plt.title('Temperature and Humidity Over Time')

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



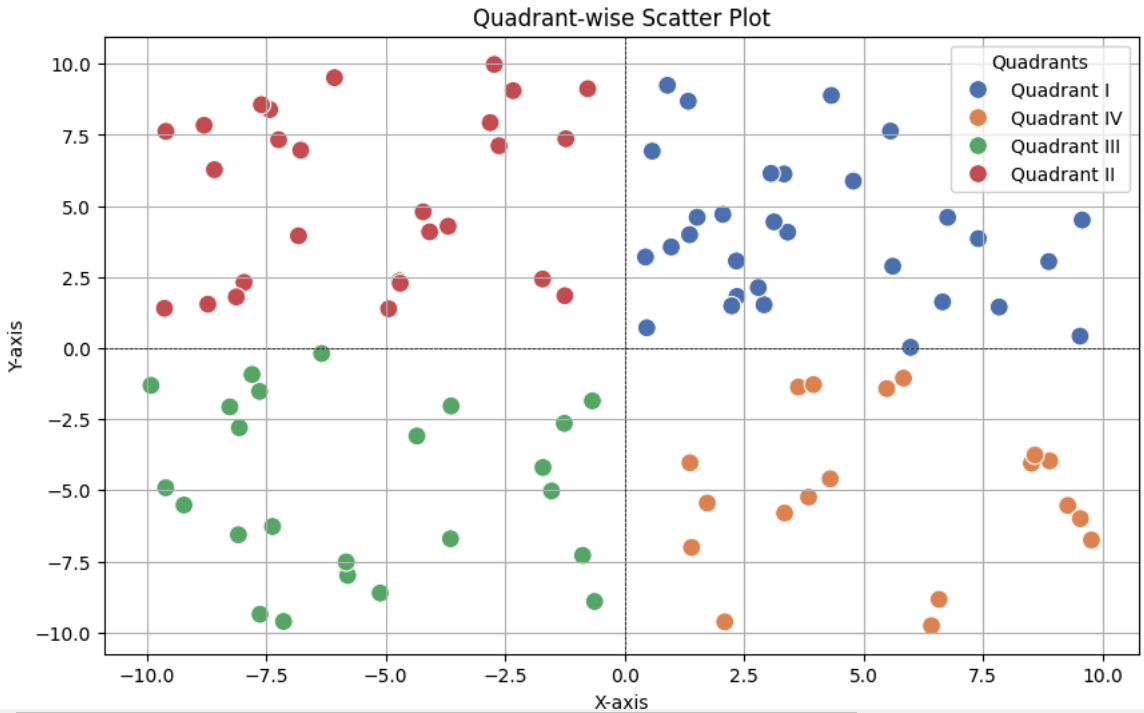

```
#15)create a numpy array data containing 1000 samples from a normal distribution perform the following tasks using matplotlib
#a) Plot a histogram of the data with 30 bins.

#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
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Create a NumPy array with 1000 samples from a normal distribution
data = np.random.normal(loc=0, scale=1, size=1000)  # mean=0, std=1

# a) Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.6, color='g', label='Histogram')

# b) Overlay a line plot representing the normal distribution's probability density function (PDF)
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create an array of values from xmin to xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF for a normal distribution
plt.plot(x, p, 'k', linewidth=2, label='PDF')

# c) Label the x-axis and y-axis
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

# d) Set the title of the plot
plt.title('Histogram with PDF Overlay')

# Display the legend
plt.legend()

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




```
#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

# Create a NumPy array with 1000 samples from a normal distribution
data = np.random.normal(loc=0, scale=1, size=1000)  # mean=0, std=1

# a) Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.6, color='g', label='Histogram')

# b) Overlay a line plot representing the normal distribution's probability density function (PDF)
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create an array of values from xmin to xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF for a normal distribution
plt.plot(x, p, 'k', linewidth=2, label='PDF')

# c) Label the x-axis and y-axis
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

# d) Set the title of the plot
plt.title('Histogram with PDF Overlay')  # Title of the plot

# Display the legend
plt.legend()

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



```
#17) Create a Seaborn scatter plot of two random arrays, color points based on their position relative to the
#rigin (quadrants), add a legend, label the axes, and set the title as 'Quadrant-wise Scatter Plot
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Create two random arrays
np.random.seed(0)  # For reproducibility
x = np.random.uniform(-10, 10, size=100)
y = np.random.uniform(-10, 10, size=100)

# Create a DataFrame to hold the data and determine quadrants
data = pd.DataFrame({'X': x, 'Y': y})

# Define quadrants based on x and y values
def get_quadrant(row):
    if row['X'] > 0 and row['Y'] > 0:
        return 'Quadrant I'
    elif row['X'] < 0 and row['Y'] > 0:
        return 'Quadrant II'
    elif row['X'] < 0 and row['Y'] < 0:
        return 'Quadrant III'
    else:
        return 'Quadrant IV'

# Apply the function to determine quadrants
data['Quadrant'] = data.apply(get_quadrant, axis=1)

# Create a Seaborn scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='X', y='Y', hue='Quadrant', palette='deep', s=100)

# Label the axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set the title
plt.title('Quadrant-wise Scatter Plot')

# Display the legend
plt.legend(title='Quadrants')

# Show the plot
plt.grid()
plt.axhline(0, color='black',linewidth=0.5, ls='--')
plt.axvline(0, color='black',linewidth=0.5, ls='--')
plt.show()
```




```
#18)8 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'8
```

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

# Prepare the output to display inline (use output_file('sine_wave.html') to save as HTML file)
output_notebook()

# Generate x values
x = np.linspace(0, 2 * pi, 100)  # 100 points from 0 to 2π
y = np.sin(x)  # Sine of x

# Create a Bokeh figure
p = figure(title="Sine Wave Function", x_axis_label='X', y_axis_label='sin(X)',
           width=800, height=400)

# Add a line renderer for the sine wave
p.line(x, y, line_width=2, color='blue', legend_label='sin(x)')

# Add grid lines
p.grid.grid_line_alpha = 0.5  # Set grid line transparency

# Display the plot
show(p)
```

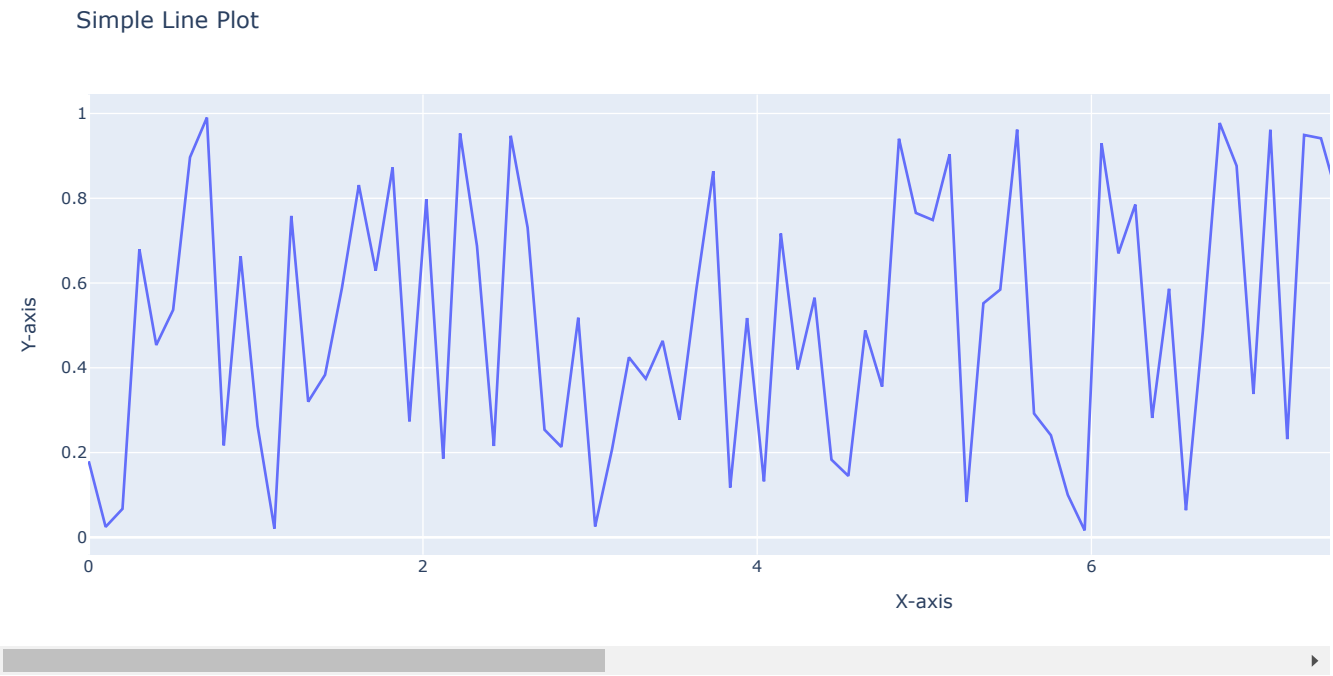
```
#19)Using Bokeh, generate a bar chart of randomly generated categorical data, color bars based on their
#values, add hover tooltips to display exact values, label the axes, and set the title as 'Random Categorical
#Bar Chart'
import numpy as np
import pandas as pd
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource, HoverTool

# Prepare the output to display inline (use output_file('bar_chart.html') to save as HTML file)
output_notebook()

# Generate random categorical data
categories = ['Category A', 'Category B', 'Category C', 'Category D', 'Category E']
values = np.random.randint(10, 100, size=len(categories))  # Random values between 10 and 100

# Create a DataFrame for better handling
data = pd.DataFrame({'Category': categories, 'Value': values})

# Create a ColumnDataSource
source = ColumnDataSource(data)

# Create a Bokeh figure
p = figure(x_range=data['Category'], title="Random Categorical Bar Chart",
           x_axis_label='Categories', y_axis_label='Values',
           plot_height=400, plot_width=600)

# Add bars with color based on their values
p.vbar(x='Category', top='Value', source=source, width=0.9,
       line_color='white', fill_color='blue')

# Add hover tooltips
hover = HoverTool()
hover.tooltips
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-36-0ded963fea4d> in <cell line: 24>()
     22
     23 # Create a Bokeh figure
---> 24 p = figure(x_range=data['Category'], title="Random Categorical Bar Chart",
     25            x_axis_label='Categories', y_axis_label='Values',
     26            plot_height=400, plot_width=600)

                                 1 frames
/usr/local/lib/python3.10/dist-packages/bokeh/core/has_props.py in _raise_attribute_error_with_matches(self, name, properties)
    377             matches, text = sorted(properties), "possible"
    378
--> 379         raise AttributeError(f"unexpected attribute {name!r} to {self.__class__.__name__}, {text} attributes are
{nice_join(matches)}")
    380
    381     def __str__(self) -> str:

AttributeError: unexpected attribute 'plot_height' to figure, similar attributes are outer_height, height or min_height
```

Next steps: Explain error

#20)8 Using Plotly, create a basic line plot of a randomly generated dataset, label the axes, and set the title as
#'Simple Line Plot'8

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

# Generate random data
x = np.linspace(0, 10, 100)  # 100 points from 0 to 10
y = np.random.rand(100)  # Random values

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

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

# Label the axes
fig.update_layout(
    title='Simple Line Plot',
    xaxis_title='X-axis',
    yaxis_title='Y-axis'
)

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



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

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

```
# Generate random data
labels = ['Category A', 'Category B', 'Category C', 'Category D', 'Category E']
values = np.random.randint(10, 100, size=len(labels))  # Random values between 10 and 100

# Create a pie chart
fig = go.Figure(data=[go.Pie(labels=labels, values=values,
                             textinfo='percent+label',
                             hoverinfo='label+percent')])

# Set the title
fig.update_layout(title='Interactive Pie Chart')

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