In [257…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
import statistics
from bokeh.plotting import figure, show
import plotly.graph_objects as go

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

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

In [295…
```
#method1
m1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
m1

#method2
m2 = np.ones((3,3), dtype = int)

m2[0,:] = [1,2,3]
m2[1,:] = [4,5,6]
m2[2,:] = [7,8,9]

m2

#method3

m3 =np.fromfunction(lambda i,j : i*3+j+1 , (3,3), dtype =int)
m3

print('Method 1 : ', m1)
print('Method 2 : ', m2)
print('Method 3 : ', m3)
```

```
Method 1 :  [[1 2 3]
 [4 5 6]
 [7 8 9]]
Method 2 :  [[1 2 3]
 [4 5 6]
 [7 8 9]]
Method 3 :  [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

2. Using numpy function, generate an array of 100evenly spaced numbers between 1 to 10 and reshape that 1D array into a 2D array.

In [15]:
```
#Generating array of 100 evenly spaced numbers between 1 to 10
a = np.linspace(1,10, 100)
a
```

Out[15]:
```
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.        ])
```

In [17]:
```
# reshaping array from 1D into 2D array
b = a.reshape(50,2)
b.ndim
```

Out[17]:
```
2
```

In [39]:
```
b
```

Out[39]:
```
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.        ]])
```

3. Explain the following terms.

a) The difference in np.array, np.asarray and np.asanyarray.

b) The difference between Deep copy and shallow copy.

In [ ]:
```
#3. a) The difference in np.array, np.asarray and np.asanyarray

# np.array : np.array, it creates a copy of the object array or the original array and does not reflect any cha

# np.asarray : np.asarray, it would reflect all the changes made to the original array.

# np.asanyarray : np.asanyarray() function is used to convert a given input to an ndarray, but pass ndarray sub
```

In [ ]:
```
#3. b) The difference between Deep copy and shallow copy.

# Deep copy :- Copies both the structure and all nested objects, resulting in a completely independent copy.Cha

# Shallow copy :- Copies the structure but shares references to the nested objects.Changes made to a nested obj
```

4. Generate a 3x3 array with random floating-point numbers between 5 and 20. Then, round each number in the array to 2 decimal places.

In [34]:
```
arr = np.around(np.random.uniform(low = 5, high = 20, size=(3,3,3)), decimals = 2)
arr
```

Out[34]:
```
array([[[ 9.88,  9.75,  6.58],
        [ 6.64, 14.  , 18.78],
        [ 7.42,  5.83, 18.76]],

       [[19.91, 17.04, 11.72],
        [10.32, 19.91, 19.45],
        [ 6.78, 12.8 , 13.59]],

       [[15.96, 14.79, 16.59],
        [10.09, 10.46,  5.38],
        [18.43, 16.63,  9.52]]])
```

In [35]:
```
arr.ndim
```

Out[35]:
```
3
```

5. Create a NumPy array with random integers between 1 & 10 of shape (5, 6). After creating the array perform following operations :

a) Extract all even intergers from array b) Extract all odd intergers from array

In [38]:
```
# 5. Create a NumPy array with random integers between 1 & 10 of shape (5, 6)
arr1 = np.random.randint(1, 10, size = (5,6))
arr1
```

Out[38]:
```
array([[2, 2, 2, 3, 3, 6],
       [3, 3, 9, 8, 3, 2],
       [2, 6, 9, 9, 6, 9],
       [3, 4, 2, 3, 3, 3],
       [4, 3, 7, 1, 9, 9]])
```

In [41]:
```
# 5.a) Extract all even intergers from array
arr_even = arr1[arr1 % 2 == 0]
arr_even
```

Out[41]:
```
array([2, 2, 2, 6, 8, 2, 2, 6, 6, 4, 2, 4])
```

In [42]:
```
#5.b) Extract all odd intergers from array
arr_odd = arr1[arr1 % 2 == 1]
arr_odd
```

Out[42]:
```
array([3, 3, 3, 3, 9, 3, 9, 9, 9, 3, 3, 3, 3, 3, 7, 1, 9, 9])
```

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 maximum values along each depth level (third axis) b) Perform element-wise multiplication of between both array

In [297…
```
# 6. Create a 3D NumPy array of shape (3,3,3) containing random integers between 1 and 10.
arr2 = np.random.randint(1, 10, size = (3,3,3))
arr2
```

Out[297]:
```
array([[[5, 6, 7],
        [9, 9, 7],
        [6, 3, 7]],

       [[9, 8, 9],
        [4, 1, 6],
        [6, 7, 5]],

       [[8, 7, 1],
        [4, 8, 7],
        [1, 4, 1]]])
```

In [298…
```
#6. a) Find the indices of maximum values along each depth level (third axis)
indices= np.argmax(arr2,axis = 2)
indices
```

Out[298]:
```
array([[2, 0, 2],
       [0, 2, 1],
       [0, 1, 1]], dtype=int64)
```

In [300…
```
#6. b) Perform element-wise multiplication of between both array

multiplication = arr2*indices
multiplication
```

Out[300]:
```
array([[[10,  0, 14],
        [ 0, 18,  7],
        [ 0,  3,  7]],

       [[18,  0, 18],
        [ 0,  2,  6],
        [ 0,  7,  5]],

       [[16,  0,  2],
        [ 0, 16,  7],
        [ 0,  4,  1]]], dtype=int64)
```

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.

In [79]:
```
#we have read file
df = pd.read_csv(r"C:\Users\Akshata Jain\Downloads\People_Data.csv")
df
```

Out[79]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	S
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Probation officer	9
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-1931	Dancer	8
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	(599)782-0605	25-11-2013	Copy	5
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	NaN	17-11-2012	Counselling psychologist	6
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	(390)417-1635x3010	15-04-1923	Biomedical engineer	10
...	...	...	...	...	...	...	...	...	...	
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	021.775.2933	05-01-1959	Personnel officer	9
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	Education administrator	5
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	Commercial/residential surveyor	6
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	(915)292-2254	31-08-1971	Ambulance person	10
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079.752.5424x67259	24-01-1947	Nurse, learning disability	9

1000 rows × 10 columns

In [80]:
```
#check for datatype for  all columns
df.dtypes
```

Out[80]:
```
Index             int64
User Id          object
First Name       object
Last Name        object
Gender           object
Email            object
Phone            object
Date of birth    object
Job Title        object
Salary            int64
dtype: object
```

In [84]:
```
#We have deal with null using fillna()
df['Phone'] = df['Phone'].fillna(0)
df
```

Out[84]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	S
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Probation officer	9
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	0	26-07-1931	Dancer	8
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	(599)782-0605	25-11-2013	Copy	5
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	0	17-11-2012	Counselling psychologist	6
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	(390)417-1635x3010	15-04-1923	Biomedical engineer	10
...	...	...	...	...	...	...	...	...	...	
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	021.775.2933	05-01-1959	Personnel officer	9
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	Education administrator	5
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	Commercial/residential surveyor	6
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	(915)292-2254	31-08-1971	Ambulance person	10
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079.752.5424x67259	24-01-1947	Nurse, learning disability	9

1000 rows × 10 columns

In [87]:

```
# we have remove non-numeric character.
df['Phone'] = df['Phone'].replace(regex=[r'\D+'], value="")
df
```

Out[87]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	Sa
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	8571398239	27-01-2014	Probation officer	9(
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	0	26-07-1931	Dancer	8(
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	5997820605	25-11-2013	Copy	5(
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	0	17-11-2012	Counselling psychologist	6!
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	39041716353010	15-04-1923	Biomedical engineer	10(
...	...	...	...	...	...	...	...	...	...	
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	0217752933	05-01-1959	Personnel officer	9(
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	0011497107799721	06-10-2001	Education administrator	5(
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	1750774412833265	13-05-1918	Commercial/residential surveyor	6(
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	9152922254	31-08-1971	Ambulance person	10(
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079752542467259	24-01-1947	Nurse, learning disability	9(

1000 rows × 10 columns

In [95]:
```
# conver Phone column into numeric type.
df['Phone'] = pd.to_numeric(df['Phone'])
```

In [94]:
```
# Check data tyupe
df['Phone'].dtype
```

Out[94]:
```
dtype('int64')
```

8. Perform following task using people dataset

a) Read data.csv using pandas skipping the first 50 rows

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

c) Display first 10 rows of the filtered dataset

d) Extract the 'Salary column as a series and display its last 5 values

In [104…
```
#8.a) Read data.csv using pandas skipping the first 50 rows
df = pd.read_csv(r"C:\Users\Akshata Jain\Downloads\People_Data.csv", skiprows = 50, header = None)
df
```

Out[104]:

	0	1	2	3	4	5	6	7	8	
0	50	afF3018e9cdd1dA	George	Mercer	Female	douglascontreras@example.net	+1-326-669-0118x4341	11-09-1941	Human resources officer	7(
1	51	CccE5DAb6E288e5	Jo	Zavala	Male	pamela64@example.net	001-859-448-9935x54536	23-11-1992	Nurse, adult	8(
2	52	DfBDc3621D4bcec	Joshua	Carey	Female	dianashepherd@example.net	001-274-739-8470x814	07-01-1915	Seismic interpreter	7(
3	53	f55b0A249f5E44D	Rickey	Hobbs	Female	ingramtiffany@example.org	241.179.9509x498	01-07-1910	Barrister	6(
4	54	Ed71DcfaBFd0beE	Robyn	Reilly	Male	carriecrawford@example.org	207.797.8345x6177	27-07-1982	Engineer, structural	10(
...	...	...	...	...	...	...	...	...	...	
946	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	021.775.2933	05-01-1959	Personnel officer	9(
947	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	Education administrator	5(
948	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	Commercial/residential surveyor	6(
949	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	(915)292-2254	31-08-1971	Ambulance person	10(
950	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079.752.5424x67259	24-01-1947	Nurse, learning disability	9(

951 rows × 10 columns

In [114…

```
#8. b) Only read the columns : 'Last Name', 'Gender', 'Email', 'Phone' and 'Salary' from the file.
df1 = pd.read_csv(r"C:\Users\Akshata Jain\Downloads\People_Data.csv")
df_new = df1[['Last Name', 'Gender', 'Email', 'Phone', 'Salary']]
df_new
```

Out[114]:

	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
...	...	...	...	...	...
995	Bryant	Female	lyonsdaisy@example.net	021.775.2933	90000
996	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	50000
997	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	60000
998	Phelps	Male	wanda04@example.net	(915)292-2254	100000
999	Tran	Female	deannablack@example.org	079.752.5424x67259	90000

1000 rows × 5 columns

In [115…

```
#8.c) Display first 10 rows of the filtered dataset
df_new.head(10)
```

Out[115]:

	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 [116…
```
#8. d) Extract the 'Salary column as a series and display its last 5 values
Salary = df_new['Salary'].tail(5)
Salary
```

Out[116]:
```
995     90000
996     50000
997     60000
998    100000
999     90000
Name: Salary, dtype: int64
```

9. Filter and select rows from People_dataset, where the 'Last Name' column contains the name 'Duke', 'Gender' column contains the word Female and Salary shoud be less than 85000.

In [140…
```
df2 = df1[(df1['Last Name']=='Duke') & (df1['Gender']=='Female') & (df1['Salary']< 85000)]
df2
```

Out[140]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	Salary
45	46	99A502C175C4EBd	Olivia	Duke	Female	diana26@example.net	001-366-475-8607x04350	13-10-1934	Dentist	60000
210	211	DF17975CC0a0373	Katrina	Duke	Female	robin78@example.com	740.434.0212	21-09-1935	Producer, radio	50000
457	458	dcE1B7DE83c1076	Traci	Duke	Female	perryhoffman@example.org	+1-903-596-0995x489	11-02-1997	Herbalist	50000
729	730	c9b482D7aa3e682	Lonnie	Duke	Female	kevinkramer@example.net	982.692.6257	12-05-2015	Nurse, adult	70000

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

In [142…
```
random_df = pd.DataFrame(np.random.randint(1, 6, size = (7,5)))
random_df
```

Out[142]:

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

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 'oining these Series by column, and, change the names of the columns to 'col1', 'col2', etc

In [154…
```
#11. a) The first Series should contain random numbers ranging from 10 to 50.
first_series = np.random.randint(10, 50, size = (50,1))
first_series
```

Out[154]:
```
array([[10],
       [37],
       [18],
       [45],
       [23],
       [11],
       [23],
       [42],
       [29],
       [31],
       [37],
       [49],
       [25],
       [30],
       [11],
       [15],
       [13],
       [19],
       [14],
       [40],
       [28],
       [24],
       [42],
       [26],
       [43],
       [29],
       [43],
       [18],
       [19],
       [28],
       [38],
       [42],
       [20],
       [13],
       [12],
       [32],
       [18],
       [40],
       [33],
       [43],
       [26],
       [17],
       [37],
       [31],
       [46],
       [34],
       [14],
       [16],
       [44],
       [45]])
```

In [156…
```
#11.b) The second Series should contain random numbers ranging from 100 to 1000.

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

Out[156]:
```
array([[592],
       [641],
       [735],
       [437],
       [437],
       [508],
       [175],
       [998],
       [109],
       [900],
       [326],
       [911],
       [220],
       [525],
       [118],
       [533],
       [935],
       [301],
       [851],
       [754],
       [441],
       [674],
       [559],
       [117],
       [980],
       [529],
       [229],
       [228],
       [785],
       [672],
       [389],
       [510],
       [735],
       [290],
       [607],
       [890],
       [518],
       [731],
       [843],
       [591],
       [741],
       [130],
       [165],
       [670],
       [213],
       [723],
       [165],
       [167],
       [600],
       [554]])
```

In [169…
```
#11. c) Create a DataFrame by 'oining these Series by column, and, change the names of the columns to 'col1', '

first_series_df = pd.DataFrame(first_series)
first_series_df
second_series_df = pd.DataFrame(second_series)
second_series_df

result = pd.concat([first_series_df, second_series_df], axis=1, join='outer')
result1 = result.set_axis(['col1', 'col2'], axis='columns')
result1
```

Out[169]:

	col1	col2
0	10	592
1	37	641
2	18	735
3	45	437
4	23	437
5	11	508
6	23	175
7	42	998
8	29	109
9	31	900
10	37	326
11	49	911
12	25	220
13	30	525
14	11	118
15	15	533
16	13	935
17	19	301
18	14	851
19	40	754
20	28	441
21	24	674
22	42	559
23	26	117
24	43	980
25	29	529
26	43	229
27	18	228
28	19	785
29	28	672
30	38	389
31	42	510
32	20	735
33	13	290
34	12	607
35	32	890
36	18	518
37	40	731
38	33	843
39	43	591
40	26	741
41	17	130
42	37	165
43	31	670
44	46	213
45	34	723
46	14	165
47	16	167
48	44	600
49	45	554

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

In [174…
```
#12.a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.
df12 = pd.read_csv(r"C:\Users\Akshata Jain\Downloads\People_Data.csv")
df12 = df12.drop(['Email', 'Phone', 'Date of birth'], axis=1)
df12
```

Out[174]:

	Index	User Id	First Name	Last Name	Gender	Job Title	Salary
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	Probation officer	90000
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	Dancer	80000
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	Copy	50000
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	Counselling psychologist	65000
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	Biomedical engineer	100000
...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	Personnel officer	90000
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	Education administrator	50000
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	Commercial/residential surveyor	60000
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	Ambulance person	100000
999	1000	8b756f6231DDC6e	Lee	Tran	Female	Nurse, learning disability	90000

1000 rows × 7 columns

In [176…
```
#12. b) Delete the rows containing any missing values.
null_delete = df12.dropna(axis = 0, how='any')
null_delete
```

Out[176]:

	Index	User Id	First Name	Last Name	Gender	Job Title	Salary
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	Probation officer	90000
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	Dancer	80000
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	Copy	50000
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	Counselling psychologist	65000
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	Biomedical engineer	100000
...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	Personnel officer	90000
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	Education administrator	50000
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	Commercial/residential surveyor	60000
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	Ambulance person	100000
999	1000	8b756f6231DDC6e	Lee	Tran	Female	Nurse, learning disability	90000

1000 rows × 7 columns

In [178…
```
#12. d) Print the final output also
null_delete
```

Out[178]:

	Index	User Id	First Name	Last Name	Gender	Job Title	Salary
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	Probation officer	90000
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	Dancer	80000
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	Copy	50000
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	Counselling psychologist	65000
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	Biomedical engineer	100000
...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	Personnel officer	90000
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	Education administrator	50000
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	Commercial/residential surveyor	60000
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	Ambulance person	100000
999	1000	8b756f6231DDC6e	Lee	Tran	Female	Nurse, learning disability	90000

1000 rows × 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'.

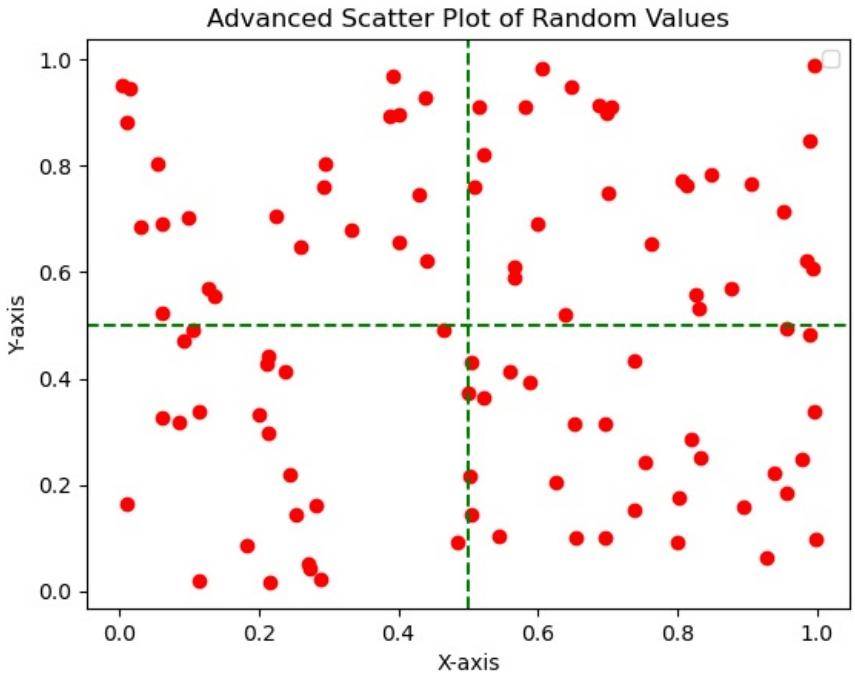
f) Display a legend for the scatter plot, the horizontal line, and the vertical line.

In [184…
```
#13. Create two NumPy arrays, x and y, each containing 100 random float values between 0 and 1.
x= np.random.uniform(0, 1, size = 100)
x
y= np.random.uniform(0, 1, size = 100)
y
```

Out[184]:
```
array([0.68457314, 0.52360129, 0.74988844, 0.31383048, 0.44153161,
       0.7663809 , 0.08704315, 0.52033232, 0.2154033 , 0.56862196,
       0.14467461, 0.74692024, 0.39360379, 0.58893101, 0.42896986,
       0.77325287, 0.01793618, 0.55588702, 0.78399308, 0.65457977,
       0.31721992, 0.16105047, 0.55906138, 0.05083768, 0.36417495,
       0.56805667, 0.09805345, 0.90958758, 0.09143458, 0.24084491,
       0.33739133, 0.62248211, 0.41172373, 0.41272367, 0.49522564,
       0.69217196, 0.68964827, 0.09935155, 0.1841267 , 0.06194745,
       0.76164125, 0.22151882, 0.31423526, 0.02144239, 0.76044684,
       0.16415177, 0.20415809, 0.28588556, 0.60608329, 0.70217044,
       0.04287231, 0.62254334, 0.43293731, 0.49040045, 0.09311121,
       0.84669288, 0.24836654, 0.88226633, 0.10227866, 0.70571999,
       0.71368007, 0.89700938, 0.48241914, 0.09975609, 0.96989126,
       0.94607203, 0.33236803, 0.76273194, 0.64737522, 0.17635543,
       0.89328201, 0.9117383 , 0.80294322, 0.53103447, 0.91209699,
       0.92921448, 0.61084546, 0.29669256, 0.49213483, 0.98941846,
       0.1436666 , 0.01661411, 0.91361728, 0.95112101, 0.65514393,
       0.25137372, 0.32714553, 0.2182965 , 0.15232971, 0.98342494,
       0.82054582, 0.15865055, 0.94785353, 0.3720411 , 0.33919494,
       0.68016173, 0.80423308, 0.42682047, 0.47076646, 0.89961965])
```

In [205…
```
#13. a) Create a scatter plot using x and y, setting the color of the points to red and the marker style to 'o'
plt.scatter(x, y, c= "red", marker='o')

#13. b) Add a horizontal line at y = 0.5 using a dashed line style and label it as 'y = 0.5'
plt.axhline(y=0.5, color='green', linestyle='--')

#13. c) Add a vertical line at x = 0.5 using a dotted line style and label it as 'x = 0.5'
plt.axvline(x=0.5, color='Green', linestyle='--')

#13. d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

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

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

plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are igno
red when legend() is called with no argument.
```



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.

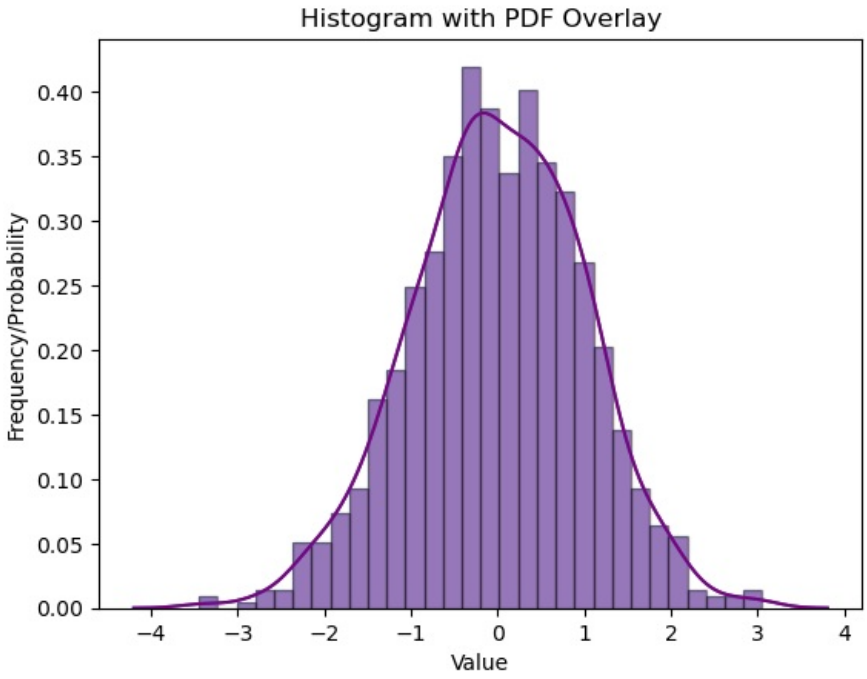
b) Overlay a line plot representing the normal distribution's probability density function (PDF).

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'.

In [240…

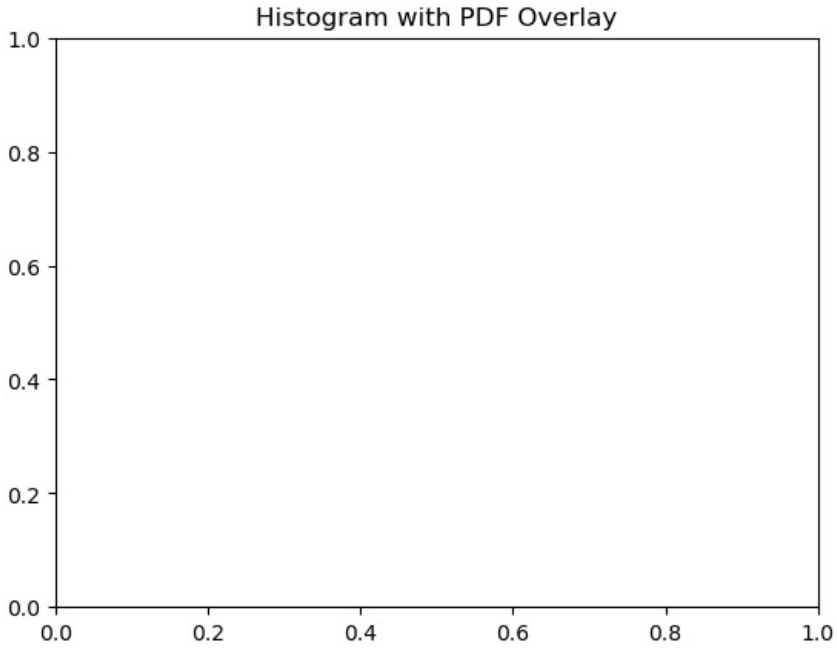
```
#15. Create a NumPy array data containing 1000 samples from a normal distribution
Noraml_dist = np.random.normal(size=1000)
Noraml_dist

#15. a) Plot a histogram of the data with 30 bins.
sns.distplot(Noraml_dist, bins=30)

#15. b) Overlay a line plot representing the normal distribution's probability density function (PDF).
sns.distplot(Noraml_dist, bins=30, color = 'purple', hist_kws={"edgecolor": 'black'})

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

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

plt.show()
```



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

In [241…
```
plt.title('Histogram with PDF Overlay')
```

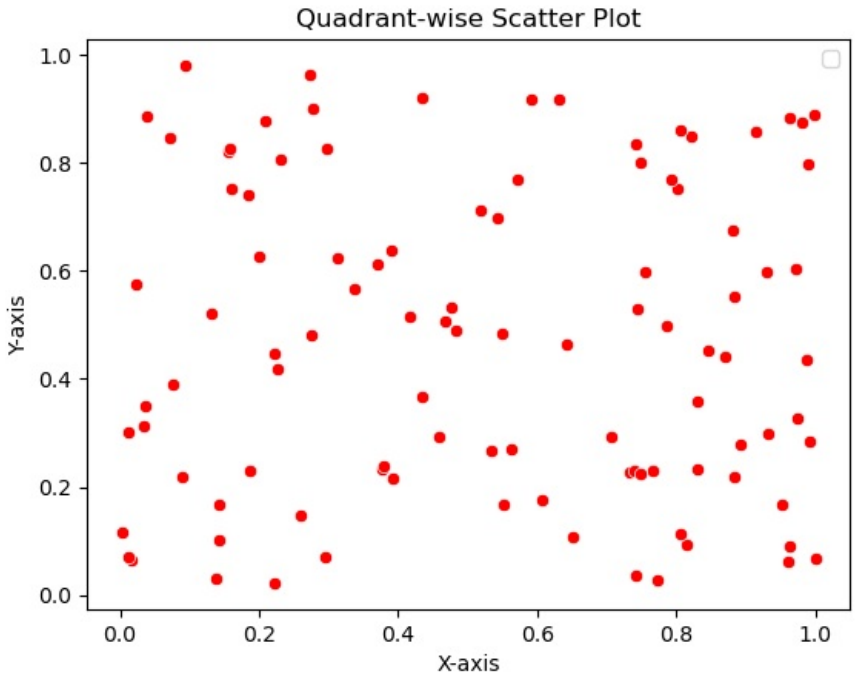
Out[241]:
```
Text(0.5, 1.0, 'Histogram with PDF Overlay')
```



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'.

In [269…
```
x= np.random.uniform(0, 1, size = 100)
x
y= np.random.uniform(0, 1, size = 100)
y
sns.scatterplot(x= np.random.uniform(0, 1, size = 100), y= np.random.uniform(0, 1, size = 100), color = 'Red')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

plt. title('Quadrant-wise Scatter Plot')

plt.legend()

plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are igno
red when legend() is called with no argument.
```



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

In [243…
```
#Wave Function'

# Generate data
x = np.arange(0, 5*np.pi, 0.1)
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='sin(x)', width=800, height=400)

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

# Show grid lines
p.grid.grid_line_color = 'black'
p.grid.grid_line_alpha = 0.5

# Show 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'.

In [309…
```
from bokeh.models import ColumnDataSource
from bokeh.palettes import Bright6
from bokeh.plotting import figure, show

fruits = ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries']
counts = [5, 3, 4, 2, 4, 6]

source = ColumnDataSource(data=dict(fruits=fruits, counts=counts, color=Bright6))

p = figure(x_range=fruits, y_range=(0,9), height=350, title="Fruit Counts",
           toolbar_location=None, tools="")

p.vbar(x='fruits', top='counts', width=0.9, color='color', legend_field="fruits", source=source)

p.xgrid.grid_line_color = None
p.legend.orientation = "horizontal"
p.legend.location = "top_center"

show(p)
```

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

In [279…
```
import plotly.express as px
```

```
x = np.random.uniform(0, 1, size = 100)
y = x*2

fig = px.line( x = x ,
               y = y,
               title = 'Simple Line 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'

In [289…

```
#Q21. Using Plotly, create an interactive pie chart of randomly generated data, add labels and percentages, set

# Generate random data
Class = ['I', 'II', 'III', 'General']
Passangers = np.random.randint(1, 100, size=len(labels))

# Create the pie chart
fig = go.Figure(data=[go.Pie(labels=Class, values=Passangers, hole=0.4)])

# Update layout for title and labels
fig.update_layout(title_text='Interactive Pie Chart',annotations=[dict(text='Pie', x=0.5, y=0.5, font_size=20,

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