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

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

```
method_1 = np.array([[1,2,3],[4,5,6]])
method_1

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

method_2 = np.full((2,3),[[1,2,3],[3,4,9]])
method_2

array([[1, 2, 3],
       [3, 4, 9]])

method_3 = np.random.randint(1,20,(3,4))
method_3

array([[17, 15, 17,  1],
       [ 4, 10,  8, 18],
       [ 7, 12, 10,  5]])
```

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

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

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.        ])
```

```
two_d = one_d.reshape(10,10)
two_d
```

```
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.
]])
```

Q.3. Explain the following terms:

- The difference in np.array, np.asarray and np.asanyarray.
- The difference between Deep copy and shallow copy.

Difference between np.array, np.asarray, and np.asanyarray

np.array: Purpose: Creates a new array.¶ Behavior: Always copies the data, meaning it creates a new array in memory. If we pass an existing array to np.array, it will create a new copy of that array. Usage: Use this when we want to ensure that we have a new array with the exact properties (like dtype) you specify

np.asarray:

Purpose: Converts the input to an array, but does not necessarily make a copy. Behavior: If the input is already an array of the same dtype, it returns the original array without making a copy. If the input is not an array, or the dtype needs to be changed, it will create a new array. Usage: Use this when we want to ensure that we have an array but do not need to copy the data if it's already an array.

np.asanyarray:

Purpose: Similar to np.asarray, but it is more flexible with subclasses of ndarray. Behavior: If the input is a subclass of ndarray (e.g., matrix), np.asanyarray will return the input as is, without forcing it to be a base ndarray. Usage: Use this when we want to ensure that we have an array or array-like object but want to preserve any special subclasses of arrays.

Difference between Deep Copy and Shallow Copy

Shallow Copy:

Definition: A shallow copy creates a new object, but the elements (references) within the object still point to the same memory locations as the original object. Behavior: If the object is a collection (e.g., a list or an array), the shallow copy will have references to the same elements as the original collection. Modifying elements inside the collection will affect both the original and the shallow copy.

```
import copy

original = [[1, 2, 3], [4, 5, 6]]
shallow_copy = copy.copy(original)

shallow_copy[0][0] = 10

print("Original:", original)
print("Shallow Copy:", shallow_copy)
```

```
Original: [[10, 2, 3], [4, 5, 6]]
Shallow Copy: [[10, 2, 3], [4, 5, 6]]
```

Deep Copy:

Definition: A deep copy creates a new object and recursively copies all objects found within the original object, ensuring that no references are shared between the original and the copy. Behavior: The deep copy is completely independent of the original. Modifications to the deep copy will not affect the original object, and vice versa

```
deep_copy = copy.deepcopy(original)

deep_copy[0][0] = 20

print("Original:", original)
print("Deep Copy:", deep_copy)
```

```
Original: [[10, 2, 3], [4, 5, 6]]
Deep Copy: [[20, 2, 3], [4, 5, 6]]
```

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

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

```
rounded_array = np.round(random_array, 2)

print(random_array)

print(rounded_array)
```

```
[[13.6622195   5.99190505  5.69044118]
 [13.94100494 15.0386677  14.08417693]
 [12.90352848 13.32640951 13.61656607]]
[[13.66  5.99  5.69]
 [13.94 15.04 14.08]
 [12.9  13.33 13.62]]
```

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

a)Extract all even integers from array.

b)Extract all odd integers from array¶

```
np.random.seed(42)
array = np.random.randint(1,10,(5,6))
array
```

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

```
array[array % 2==0]
```

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

```
array[array % 2!=0]
```

```
array([7, 5, 7, 3, 7, 5, 3, 5, 5, 1, 9, 1, 3, 7, 9, 3, 5])
```

Q.6.Create a 3D NumPy array of shape (3,3,3) containing random integers between 1 and 10. Perform the following operations:

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

b) Perform element-wise multiplication of between both array.

```
array_3d=np.random.randint(1,10, size=(3,3,3))
array_3d

array([[[3, 7, 5],
        [9, 7, 2],
        [4, 9, 2]],

       [[9, 5, 2],
        [4, 7, 8],
        [3, 1, 4]],

       [[2, 8, 4],
        [2, 6, 6],
        [4, 6, 2]]])

np.argmax(array_3d, axis=2)

array([[1, 0, 1],
       [0, 2, 2],
       [1, 1, 1]])

multiplied_array = np.multiply(array_3d, array_3d)
multiplied_array

array([[[ 9, 49, 25],
        [81, 49,  4],
        [16, 81,  4]],

       [[81, 25,  4],
        [16, 49, 64],
        [ 9,  1, 16]],

       [[ 4, 64, 16],
        [ 4, 36, 36],
        [16, 36,  4]]])
```

Q.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.

```
df = pd.read_csv('People Data.csv')
df.head(5)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Index\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 288,\n        \"min\": 1,\n
\"max\": 1000,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          522,\n          738,\n          741\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"User Id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"89FdFDb8Fa09efF\",\n          \"BBa02EC792cfFf3\",\n
\"b0E2bF69efAB9c5\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"First Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 526,\n        \"samples\":
[\n          \"Maureen\",\n          \"Breanna\",\n
\"Ernest\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Last Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 628,\n        \"samples\":
[\n          \"Mendez\",\n          \"Callahan\",\n
\"Martinez\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Email\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 1000,\
n        \"samples\": [\n          \"fernando58@example.com\",\n
\"willisannette@example.org\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Phone\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 979,\n        \"samples\":
[\n          \"(500)343-9851x714\",\n          \"703.544.7090\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Date of birth\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 991,\n        \"samples\": [\n          \"30-
01-1945\",\n          \"29-12-2008\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Job Title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 519,\n        \"samples\": [\n
\"Furniture designer\",\n          \"Field seismologist\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 16136,\n
\"min\": 50000,\n        \"max\": 100000,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          80000,\n
60000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df['Phone'] = df['Phone'].fillna('0000000000')
```

```
df['Phone'] = df['Phone'].str.replace(r'\D', '', regex=True)
```

```
df['Phone'] = df['Phone'].astype(float, errors='ignore')
```

```
df.dtypes
```

```
Index              int64
User Id           object
First Name        object
Last Name         object
Gender            object
Email             object
Phone            float64
Date of birth     object
Job Title         object
Salary             int64
dtype: object
```

```
df.info()
```

```
<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          1000 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
```

Q.8. Perform the following tasks using people dataset:

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

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

c) Display the first 10 rows of the filtered dataset.

d) Extract the ‘Salary’' column as a Series and display its last 5 values

```
df = pd.read_csv('People Data.csv')
df.iloc[50:].head()
```

{"repr_error":"0","type":"dataframe"}

```
df[["Last Name","Gender","Email","Phone","Salary"]]
```

```
{"summary":"{\n  \"name\": \"df[[\\\"Last
Name\\\",\\\"Gender\\\",\\\"Email\\\",\\\"Phone\\\",\\\"Salary\\\"]]\"
,\n  \"rows\": 1000,\n  \"fields\": [\n    {\n      \"column\": \"Last
Name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 628,\n        \"samples\": [\n
\"Mendez\",\n          \"Callahan\",\n          \"Martinez\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Email\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 1000,\
n        \"samples\": [\n          \"fernando58@example.com\",\n
\"willisannette@example.org\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Phone\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 979,\n        \"samples\":
[\n          \"(500)343-9851x714\",\n          \"703.544.7090\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 16136,\n
\"min\": 50000,\n        \"max\": 100000,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          80000,\n
60000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
df.Salary
```

```
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
```

```
df.Salary.tail(5)
```

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

Q.9.Filter and select rows from the People_Dataset, where the “Last Name' column contains the name 'Duke', 'Gender' column contains the word Female and ‘Salary’ should be less than 85000.

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

```
{"repr_error":"0","type":"dataframe"}
```

Q.10 Create a 7*5. Dataframe in Pandas using a series generated from 36 random integers between 1 to 6 ?

```
np.random.seed(42)
random_no = np.random.randint(1,7,35)

pd.DataFrame(random_no.reshape(7,5), columns=('a','b','c','d','e'))
```

```
{"summary":"{\n  \"name\": \"pd\",\n  \"rows\": 7,\n  \"fields\": [\n
{\n      \"column\": \"a\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 2,\n
\"max\": 5,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          3,\n          4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"b\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 6,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          5,\n          3,\n          2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"c\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 6,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          6,\n          1,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"d\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          3,\n          4,\n          2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"e\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 2,\n
\"max\": 6,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          2,\n          3,\n          4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

11. Create two different Series, each of length 50, with the following criteria:

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

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

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

```
series1 = np.random.randint(10,50,50)
series2 = np.random.randint(100,1000,50)

df = pd.DataFrame({'Col1':series1, 'Col2':series2})
df.head(5)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 50,\n  \"fields\": [\n
{\n      \"column\": \"Col1\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 11,\n        \"min\": 10,\n
\"max\": 49,\n        \"num_unique_values\": 29,\n        \"samples\":
[\n          33,\n          26,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Col2\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 270,\n        \"min\": 104,\n
\"max\": 971,\n        \"num_unique_values\": 49,\n
\"samples\": [\n          478,\n          894,\n          306\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

Q.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.

```
df = df.drop(['Email','Phone','Date of birth'],axis=1)
df.head(5)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Index\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 288,\n        \"min\": 1,\n
\"max\": 1000,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          522,\n          738,\n          741\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"User Id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"89FdFDb8Fa09efF\",\n          \"BBa02EC792cfFf3\",\n
\"b0E2bF69efAB9c5\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"First Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 526,\n        \"samples\":
[\n          \"Maureen\",\n          \"Breanna\",\n
\"Ernest\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Last Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 628,\n        \"samples\":
[\n          \"Mendez\",\n          \"Callahan\",\n
\"Martinez\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Job Title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 519,\n        \"samples\": [\n
\"Furniture designer\",\n          \"Field seismologist\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 16136,\n
\"min\": 50000,\n        \"max\": 100000,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          80000,\n
60000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df.isnull().sum()
```

```
Index         0
User Id       0
First Name    0
Last Name     0
Gender        0
Job Title     0
Salary        0
dtype: int64
```

```
df.dropna(inplace=True)
df.head(5)
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Index\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 288,\n        \"min\": 1,\n
\"max\": 1000,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          522,\n          738,\n          741\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"User Id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"89FdFDb8Fa09efF\",\n          \"BBa02EC792cfFf3\",\n
\"b0E2bF69efAB9c5\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"First Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 526,\n        \"samples\":
[\n          \"Maureen\",\n          \"Breanna\",\n
\"Ernest\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Last Name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 628,\n        \"samples\":
[\n          \"Mendez\",\n          \"Callahan\",\n
\"Martinez\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Job Title\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 519,\n        \"samples\": [\n
\"Furniture designer\",\n          \"Field seismologist\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 16136,\n
\"min\": 50000,\n        \"max\": 100000,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          80000,\n
60000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

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'.

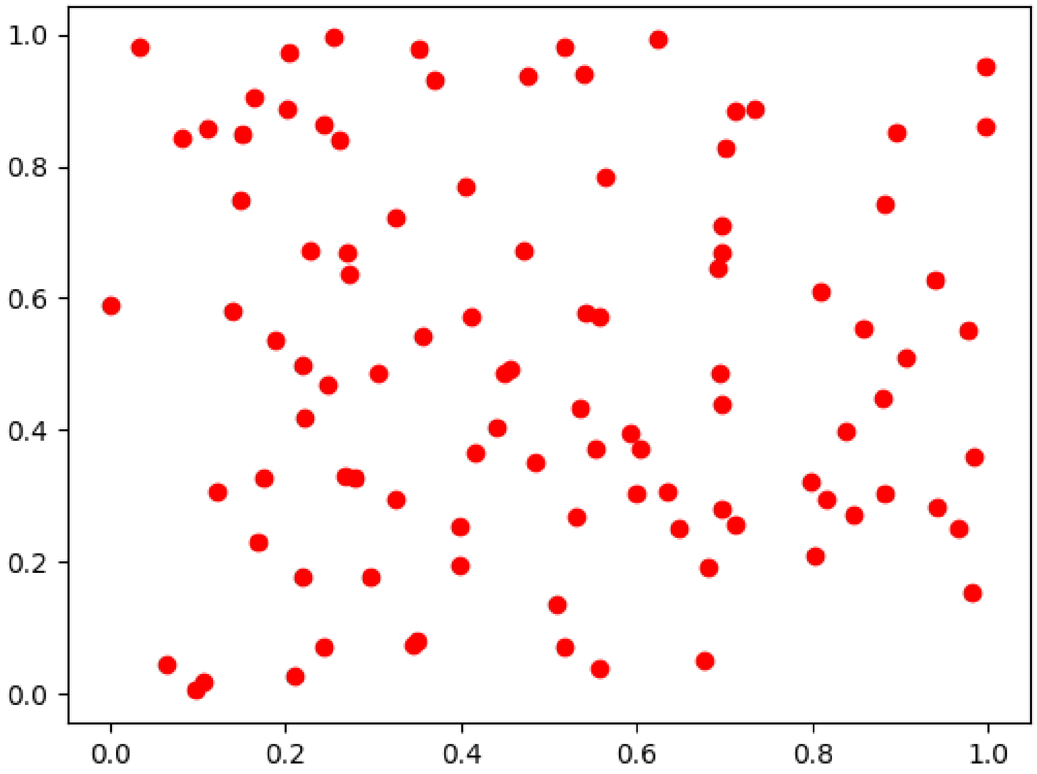
d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.

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

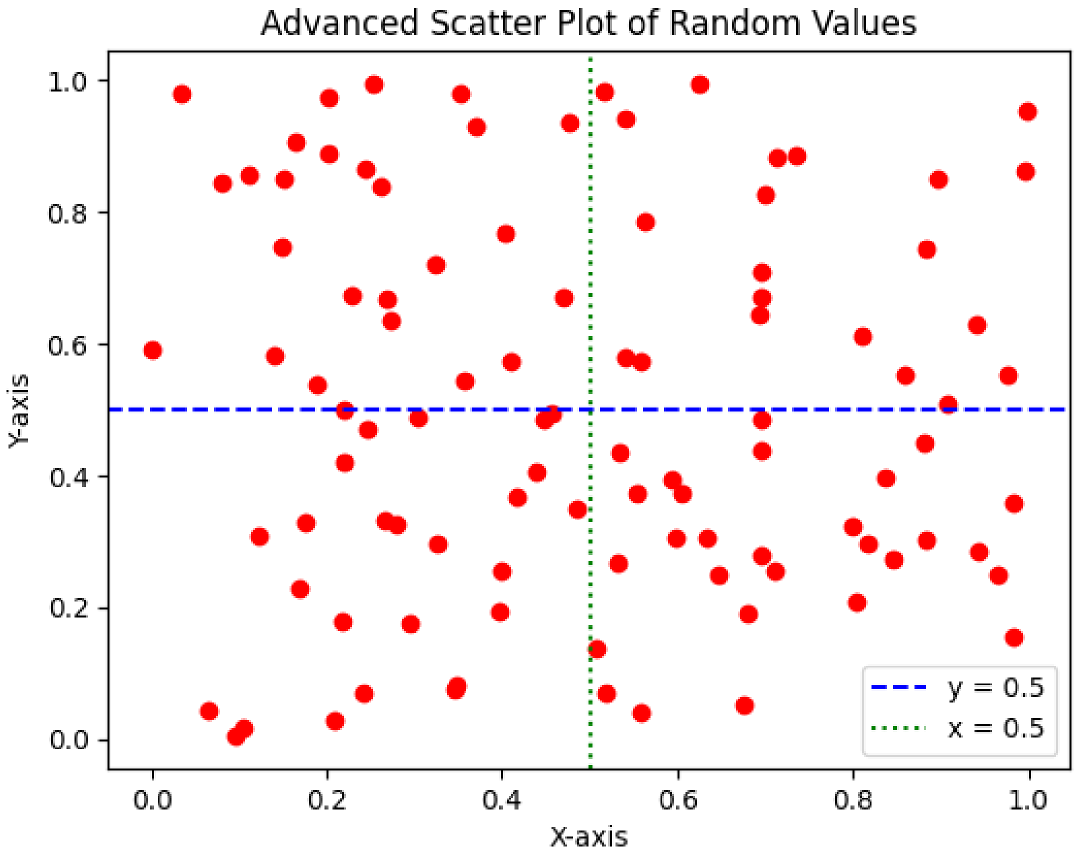
f) Display a legend for the scatter plot, the horizontal line, and the vertical line.

```python
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

x = np.random.rand(100)
y = np.random.rand(100)
plt.scatter(x,y , color = 'Red', marker ='o')
```

```
<matplotlib.collections.PathCollection at 0x7b61cbc237f0>
```

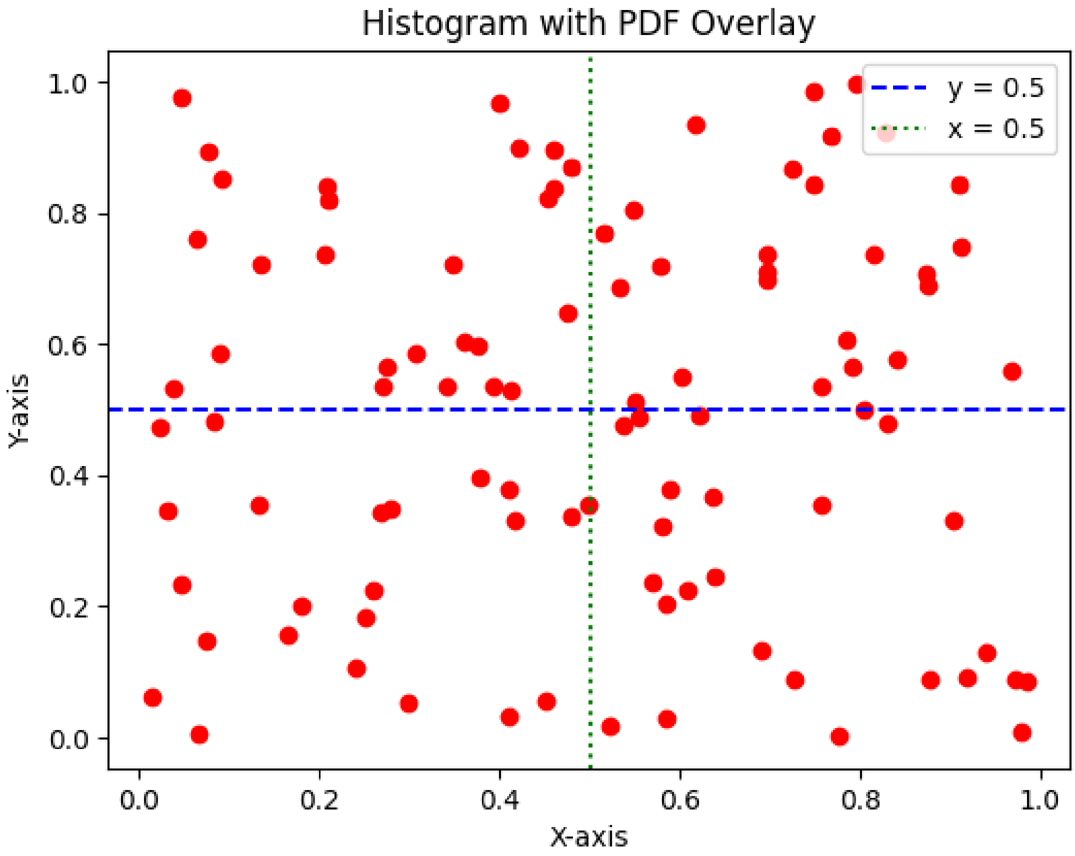


```
plt.scatter(x,y , color = 'Red', marker ='o')
plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')
plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')

plt.title('Advanced Scatter Plot of Random Values')

plt.legend()

plt.show()
```



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

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

plt.scatter(x,y , color = 'Red', marker ='o')
plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')
plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Histogram with PDF Overlay')

plt.legend()

plt.show()
```



Q.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.

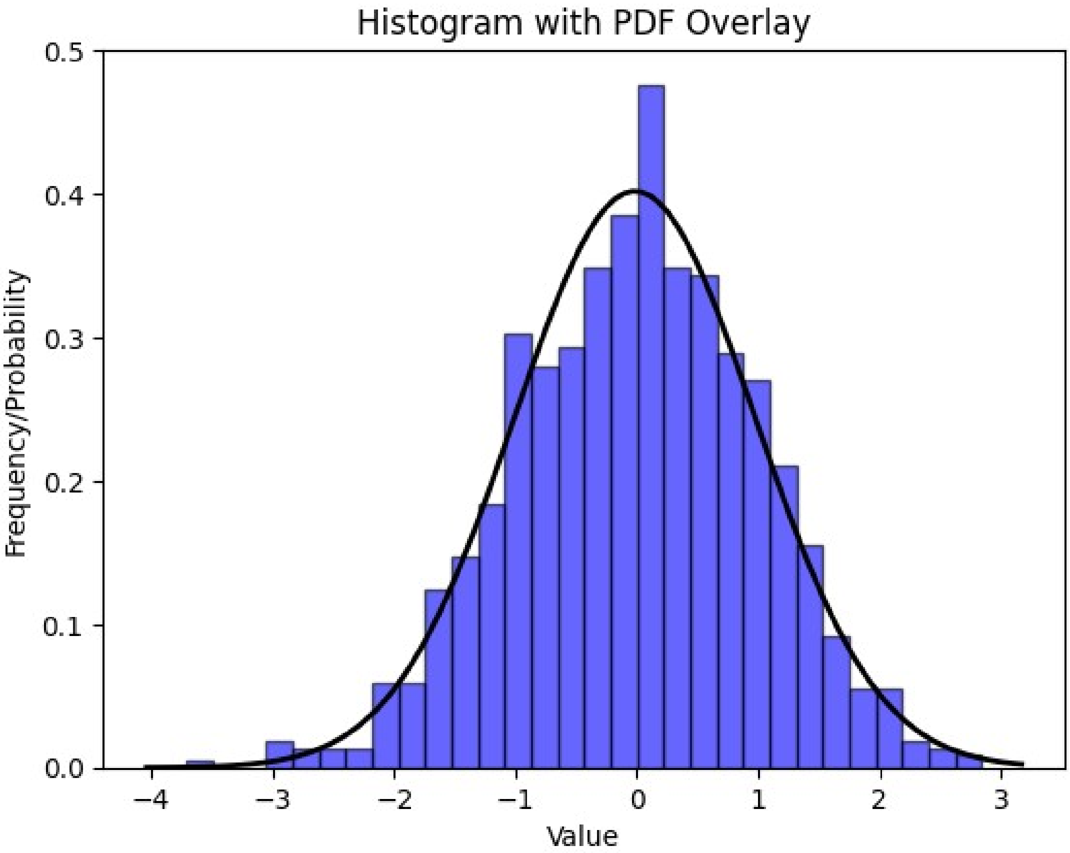
b) Overlay a line plot representing the normal distribution's probability density function (PDF).

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'.

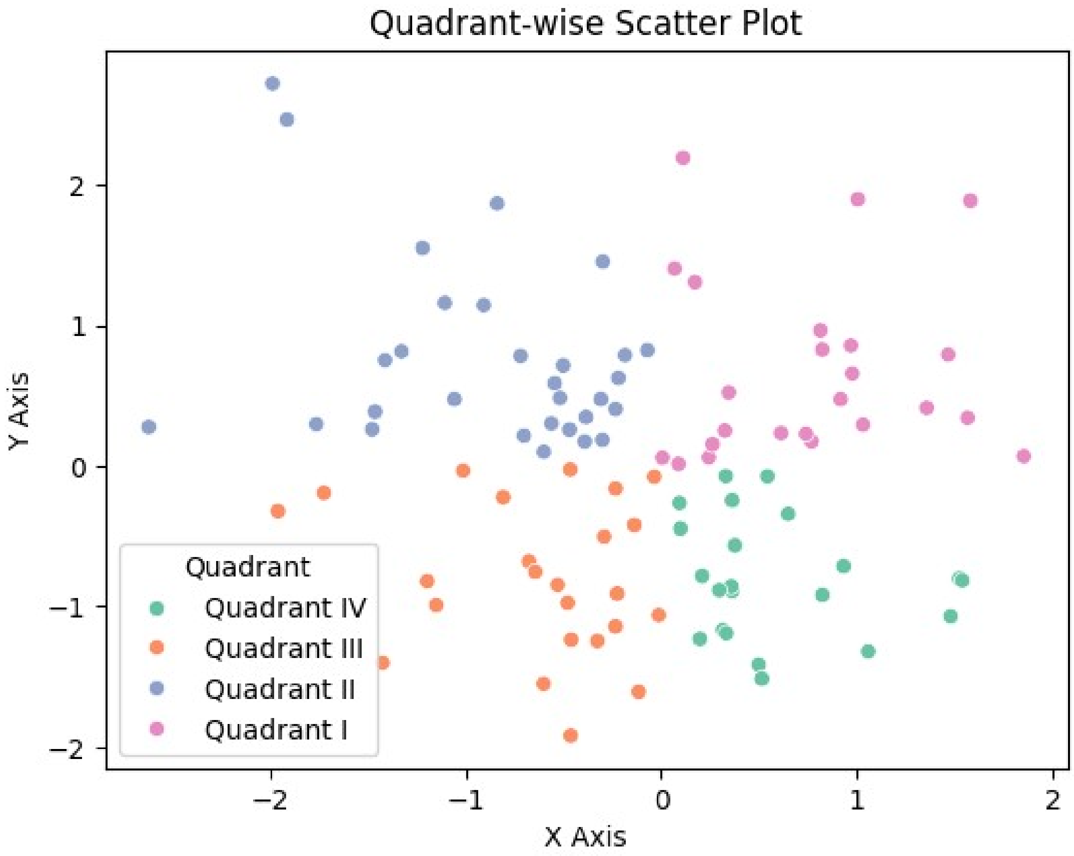
```
import scipy.stats as stats

data = np.random.normal(loc=0, scale=1, size=1000)

plt.hist(data, bins=30, density=True, alpha=0.6, color='b',
edgecolor='black')
```

```python
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, loc=np.mean(data), scale=np.std(data))
plt.plot(x, p, 'k', linewidth=2)

plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

plt.title('Histogram with PDF Overlay')

plt.show()
```



16.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'.

```python
import seaborn as sns

np.random.seed(42)
x = np.random.randn(100)
y = np.random.randn(100)

def get_quadrant(x, y):
    if x > 0 and y > 0:
        return 'Quadrant I'
    elif x < 0 and y > 0:
        return 'Quadrant II'
    elif x < 0 and y < 0:
        return 'Quadrant III'
    elif x > 0 and y < 0:
        return 'Quadrant IV'
    else:
        return 'Origin'

quadrants = [get_quadrant(x_val, y_val) for x_val, y_val in zip(x, y)]

df = pd.DataFrame({'x': x, 'y': y, 'Quadrant': quadrants})

sns.scatterplot(x='x', y='y', hue='Quadrant', palette='Set2', data=df)

plt.legend(title='Quadrant')

plt.xlabel('X Axis')
plt.ylabel('Y Axis')

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

plt.show()
```



17.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'.

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

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

p = figure(title="Sine Wave Function",
           x_axis_label='X Axis',
           y_axis_label='Y Axis',
           width=700, height=400)

p.line(x, y, line_width=2, color="blue")

p.grid.grid_line_color = "gray"
```

```
output_notebook()
show(p)
```

```
"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
```

```
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""
```

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

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import HoverTool

categories = ['Category A', 'Category B', 'Category C', 'Category D',
'Category E']
values = np.random.randint(10, 100, size=len(categories))

p = figure(x_range=categories,
           title="Random Categorical Bar Chart",
           x_axis_label='Category',
           y_axis_label='Value',
           height=400, width=600,
           tooltips=[("Category", "@x"), ("Value", "@top")])

p.vbar(x=categories, top=values, width=0.8, color="skyblue")

output_notebook()
show(p)

"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
```

```
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""
```

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

```
import plotly.graph_objects as go

np.random.seed(42)
x = np.linspace(0, 10, 100)
y = np.random.randn(100).cumsum()
```

```python
fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Random Data'))

fig.update_layout(
    title='Simple Line Plot',
    xaxis_title='X Axis',
    yaxis_title='Y Axis'
)
fig.show()
```

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

```python
import plotly.graph_objects as go
import random

# Generate random data
labels = ['Category A', 'Category B', 'Category C', 'Category D']
values = [random.randint(10, 100) for _ in labels]

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

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

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