NumPy

```
# Import the NumPy library, conventionally aliased as 'np'
import numpy as np

# Create a 1D array of integers from 0 up to (but not including) 10
my_array = np.arange(10)

# Print the array to the console
print("The created array is:")
print(my_array)

# Display the fundamental attributes of the array
print("\nArray Attributes:")
print(f"Shape: {my_array.shape}")
print(f"Size: {my_array.size}")
print(f"Data Type (dtype): {my_array.dtype}")
```

```
The created array is:
[0 1 2 3 4 5 6 7 8 9]

Array Attributes:
Shape: (10,)
Size: 10
Data Type (dtype): int32
```

```
import numpy as np

# Create two 1D arrays of the same size
array_a = np.array([1,2,3,4,5])
array_b = np.array([6,7,8,9,10])

# Perform element-wise operations
addition = array_a + array_b
subtraction = array_a - array_b
multiplication = array_a * array_b
division = array_a / array_b

# Print the results
print(f"Array A: {array_a}")
print(f"Array B: {array_b}\n")
print(f"A + B = {addition}")
print(f"A - B = {subtraction}")
print(f"A * B = {multiplication}")
print(f"A / B = {division}")
```

```
Array A: [1 2 3 4 5]
Array B: [ 6  7  8  9 10]

A + B = [ 7  9 11 13 15]
A - B = [-5 -5 -5 -5 -5]
A * B = [ 6 14 24 36 50]
A / B = [0.16666667 0.28571429 0.375      0.44444444 0.5       ]
```

```
import numpy as np

# Create a 3x3 matrix using arange() and reshape()
matrix = np.arange(9).reshape(3, 3)
print("Original 3x3 Matrix:")
print(matrix)

# 1. Print the second row
# The index for the second row is 1 (since indexing is 0-based)
# The ':' for the column means "select all columns"
second_row = matrix[1]
print("\nSecond Row:")
print(second_row)

# 2. Print the third column
# The ':' for the row means "select all rows"
# The index for the third column is 2
third_column = matrix[:,2]
print("\nThird Column:")
```

```
print(third_column)

# 3. Print a subarray (the top-left 2x2 corner)
# Row slice ':2' selects rows from index 0 up to (but not including) 2
# Column slice ':2' selects columns from index 0 up to (but not including) 2
subarray = matrix[:2, :2]
print("\nTop-left 2x2 Subarray:")
print(subarray)
```

```
Original 3x3 Matrix:
[[0 1 2]
 [3 4 5]
 [6 7 8]]

Second Row:
[3 4 5]

Third Column:
[2 5 8]

Top-left 2x2 Subarray:
[[0 1]
 [3 4]]
```

```
import numpy as np

# Create a 1D array with 12 elements
array_1d = np.arange(12)
print("Original 1D array:")
print(array_1d)

# Reshape the 1D array into a 3x4 2D array (3 rows, 4 columns)
array_2d = array_1d.reshape(3, 4)
print("\nReshaped 3x4 2D array:")
print(array_2d)

# Flatten the 2D array back to a 1D array
flattened_array = array_2d.flatten()
print("\nFlattened array:")
print(flattened_array)
```

```
Original 1D array:
[ 0  1  2  3  4  5  6  7  8  9 10 11]

Reshaped 3x4 2D array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

Flattened array:
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

```
import numpy as np

# Create two 2x2 matrices
matrix_A = np.array([[1,2],
                     [3,4]])

matrix_B = np.array([[5,6],
                     [7,8]])

print("Matrix A:")
print(matrix_A)
print("\nMatrix B:")
print(matrix_B)

# Perform matrix multiplication using np.dot()
result_matrix = np.dot(matrix_A, matrix_B)

print("\nResult of np.dot(A, B):")
print(result_matrix)
```

```
Matrix A:
[[1 2]
 [3 4]]

Matrix B:
[[5 6]
```

```
 [7 8]]

Result of np.dot(A, B):
[[19 22]
 [43 50]]
```

```
import numpy as np

# Set a seed for reproducibility of random numbers
np.random.seed(42)

# Generate an array of 20 random integers between 1 and 100 (inclusive)
# The 'high' parameter in randint is exclusive, so we use 101.
random_integers = np.random.randint(low=1, high=101, size=20)

print("Generated Random Array:")
print(random_integers)

# Calculate and print individual statistical measures
mean_val = np.mean(random_integers)
median_val = np.median(random_integers)
std_dev = np.std(random_integers)
min_val = np.min(random_integers)
max_val = np.max(random_integers)

print("\nNumPy Statistical Summary:")
print(f"Mean: {mean_val:.2f}")
print(f"Median: {median_val}")
print(f"Standard Deviation: {std_dev:.2f}")
print(f"Minimum: {min_val}")
print(f"Maximum: {max_val}")
```

```
Generated Random Array:
[ 52  93  15  72  61  21  83  87  75  75  88 100  24   3  22  53   2  88
  30  38]

NumPy Statistical Summary:
Mean: 54.10
Median: 57.0
Standard Deviation: 31.47
Minimum: 2
Maximum: 100
```

```
# Import the pandas library
import pandas as pd

# Convert the NumPy array to a Pandas Series
data_series = pd.Series(random_integers)

# Use the.describe() method to get a statistical summary
description = data_series.describe()

print("\nPandas.describe() Summary:")
print(description)
```

```
Pandas.describe() Summary:
count     20.000000
mean      54.100000
std       32.289643
min        2.000000
25%       23.500000
50%       57.000000
75%       84.000000
max      100.000000
dtype: float64
```

```
import numpy as np

# 1. Create a 4x4 identity matrix
# np.identity() creates a square matrix with 1s on the main diagonal.
identity_matrix = np.identity(4)
print("4x4 Identity Matrix:")
print(identity_matrix)

# 2. Create a diagonal matrix with values
# np.diag() takes a 1D array and places its elements on the main
```

```
# diagonal of a new 2D array, with zeros elsewhere.
diagonal_values = [1,2,3,4]
diagonal_matrix = np.diag(diagonal_values)
print("\nDiagonal Matrix with  on the diagonal:")
print(diagonal_matrix)
```

```
4x4 Identity Matrix:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

Diagonal Matrix with  on the diagonal:
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
```

Pandas

```
import pandas as pd

# Create a dictionary containing student data
student_data = {
    'Name':['Rohit','Ash',"Dipsi","Rick"],
    'Age': [19, 20, 19, 21],
    'Marks': [85,92,96,76],
}

# Create a DataFrame from the dictionary
df_students = pd.DataFrame(student_data)

print("Full DataFrame:")
print(df_students)

# Display the first row of the DataFrame
print("\nFirst row:")
print(df_students.head(1))

# Display the last row of the DataFrame
print("\nLast row:")
print(df_students.tail(1))
```

```
Full DataFrame:
    Name  Age  Marks
0  Rohit   19     85
1    Ash   20     92
2  Dipsi   19     96
3   Rick   21     76

First row:
    Name  Age  Marks
0  Rohit   19     85

Last row:
   Name  Age  Marks
3  Rick   21     76
```

```
import pandas as pd

# Create a sample DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [24, 27, 22, 32, 29],
    'City': ['Delhi', 'Mumbai', 'Bangalore', 'Chennai', 'Kolkata']
}

df = pd.DataFrame(data)
print(df)
# Access the row with label/index 2
print(df.loc[2])

# Access multiple rows by label
print(df.loc[[1, 3]])

# Access a specific value (e.g., Age of person at index 1)
```

```
print(df.loc[1, 'Age'])

# Access rows 1 to 3 and only 'Name' and 'City' columns
print(df.loc[1:3, ['Name', 'City']])
print('\n')

# Access the row at position 2
print(df.iloc[2])

# Access multiple rows by index position
print(df.iloc[[0, 4]])

# Access specific cell (e.g., row 1, column 1 => Age of Bob)
print(df.iloc[1, 1])

# Access rows 0 to 2 and first two columns
print(df.iloc[0:3, 0:2])
```

```
      Name  Age       City
0    Alice   24      Delhi
1      Bob   27     Mumbai
2  Charlie   22  Bangalore
3    David   32    Chennai
4      Eve   29    Kolkata
Name      Charlie
Age            22
City    Bangalore
Name: 2, dtype: object
    Name  Age     City
1    Bob   27   Mumbai
3  David   32  Chennai
27
      Name       City
1      Bob     Mumbai
2  Charlie  Bangalore
3    David    Chennai


Name      Charlie
Age            22
City    Bangalore
Name: 2, dtype: object
    Name  Age     City
0  Alice   24    Delhi
4    Eve   29  Kolkata
27
      Name  Age
0    Alice   24
1      Bob   27
2  Charlie   22
```

```
import pandas as pd

# Sample DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [24, 27, 22, 32, 29],
    'Marks': [85, 67, 90, 45, 76]
}

df = pd.DataFrame(data)
print("Original DataFrame:\n", df)

# Add a new column "Grade" based on Marks
def assign_grade(mark):
    if mark >= 85:
        return 'A'
    elif mark >= 70:
        return 'B'
    elif mark >= 50:
        return 'C'
    else:
        return 'D'

df['Grade'] = df['Marks'].apply(assign_grade)
print("\nAfter Adding 'Grade' Column:\n", df)
```

```
# Delete the "Age" column
df.drop(columns='Age', inplace=True)
print("\nAfter Deleting 'Age' Column:\n", df)
```

```
Original DataFrame:
      Name  Age  Marks
0    Alice   24     85
1      Bob   27     67
2  Charlie   22     90
3    David   32     45
4      Eve   29     76

After Adding 'Grade' Column:
      Name  Age  Marks Grade
0    Alice   24     85     A
1      Bob   27     67     C
2  Charlie   22     90     A
3    David   32     45     D
4      Eve   29     76     B

After Deleting 'Age' Column:
      Name  Marks Grade
0    Alice     85     A
1      Bob     67     C
2  Charlie     90     A
3    David     45     D
4      Eve     76     B
```

```
import pandas as pd

# Step 1: Read CSV into a DataFrame
df = pd.read_csv('student.csv')
print("Original DataFrame:\n", df)

# Step 2: Filter students with Marks >= 60
filtered_df = df[df['Marks'] >= 60]
print("\nFiltered DataFrame (Marks >= 60):\n", filtered_df)

# Step 3: Write the filtered DataFrame to a new CSV
filtered_df.to_csv('filtered_students.csv', index=False)
print("\nFiltered data written to 'filtered_students.csv'")
```

```
Original DataFrame:
     Name  Age   Marks     City
0  Rohit    20     92  Kolkata
1   Rick    21     56   Mumbai
2    Ash    19     96    Delhi
3  Dipsi    20     59  Chennai

Filtered DataFrame (Marks >= 60):
     Name  Age   Marks     City
0  Rohit    20     92  Kolkata
2    Ash    19     96    Delhi

Filtered data written to 'filtered_students.csv'
```

```
import pandas as pd

# Sample DataFrame with Grades
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace'],
    'Marks': [85, 67, 90, 45, 76, 55, 30]
}

# Add a 'Grade' column based on marks
def assign_grade(m):
    if m >= 85:
        return 'A'
    elif m >= 70:
        return 'B'
    elif m >= 50:
        return 'C'
    else:
        return 'D'

df = pd.DataFrame(data)
df['Grade'] = df['Marks'].apply(assign_grade)
```

```
# Filter students with marks > 75
filtered_df = df[df['Marks'] > 75][['Name', 'Grade']]

# Display the result
print("Students with Marks > 75 (Name and Grade):\n")
print(filtered_df)
```

```
Students with Marks > 75 (Name and Grade):

      Name Grade
0    Alice     A
2  Charlie     A
4      Eve     B
```

```
import pandas as pd

# Sample DataFrame with employee data
data = {
    'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Helen'],
    'Department': ['HR', 'IT', 'Finance', 'HR', 'IT', 'Finance', 'IT', 'HR'],
    'Salary': [50000, 60000, 55000, 52000, 62000, 58000, 61000, 53000]
}

df = pd.DataFrame(data)
print("Original DataFrame:\n", df)

# Group by Department and calculate average salary
avg_salary = df.groupby('Department')['Salary'].mean().reset_index()

# Rename column for clarity
avg_salary.columns = ['Department', 'Average_Salary']

print("\nAverage Salary by Department:\n", avg_salary)
```

```
Original DataFrame:
   Employee Department  Salary
0    Alice         HR   50000
1      Bob         IT   60000
2  Charlie    Finance   55000
3    David         HR   52000
4      Eve         IT   62000
5    Frank    Finance   58000
6    Grace         IT   61000
7    Helen         HR   53000

Average Salary by Department:
   Department  Average_Salary
0    Finance    56500.000000
1         HR    51666.666667
2         IT    61000.000000
```

```
import pandas as pd
import numpy as np

# Step 1: Create a DataFrame with missing values (NaN)
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, np.nan, 30, 22, np.nan],
    'Marks': [85, 67, np.nan, 90, 76]
}

df = pd.DataFrame(data)
print("Original DataFrame with Missing Values:\n", df)
```

```
Original DataFrame with Missing Values:
      Name   Age  Marks
0    Alice  25.0   85.0
1      Bob   NaN   67.0
2  Charlie  30.0    NaN
3    David  22.0   90.0
4      Eve   NaN   76.0
```

```
# Fill missing values with a default value (e.g., 0 or a placeholder)
filled_df = df.fillna({'Age': df['Age'].mean(), 'Marks': 0})
```

```
print("\nDataFrame after fillna():\n", filled_df)
```

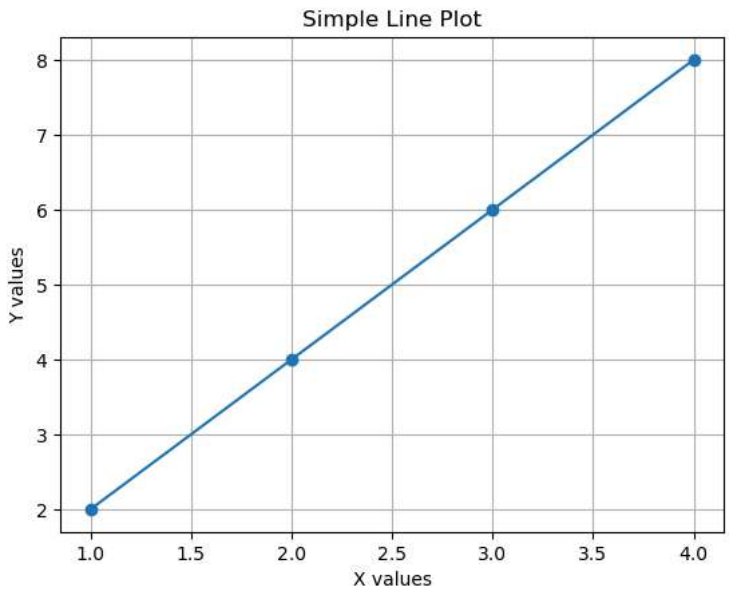
```
DataFrame after fillna():
       Name        Age  Marks
0    Alice  25.000000   85.0
1      Bob  25.666667   67.0
2  Charlie  30.000000    0.0
3    David  22.000000   90.0
4      Eve  25.666667   76.0
```

```
# Drop rows where any value is missing
dropped_df = df.dropna()
print("\nDataFrame after dropna():\n", dropped_df)
```

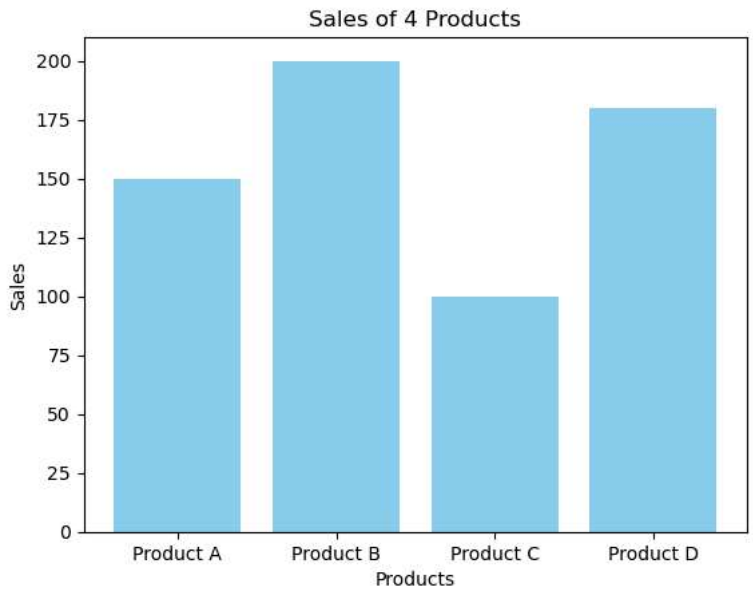
```
DataFrame after dropna():
     Name   Age  Marks
0  Alice  25.0   85.0
3  David  22.0   90.0
```

MatplotLib

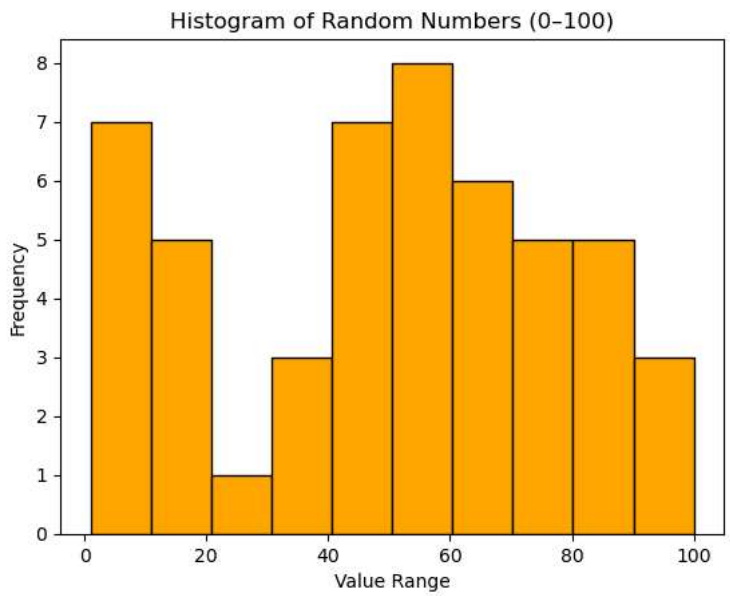
```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [2, 4, 6, 8]

plt.plot(x, y, marker='o')
plt.title('Simple Line Plot')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.grid(True)
plt.show()
```



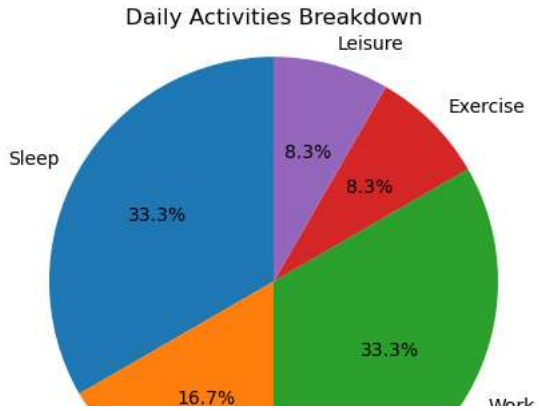

```
products = ['Product A', 'Product B', 'Product C', 'Product D']
sales = [150, 200, 100, 180]

plt.bar(products, sales, color='skyblue')
plt.title('Sales of 4 Products')
plt.xlabel('Products')
plt.ylabel('Sales')
plt.show()
```



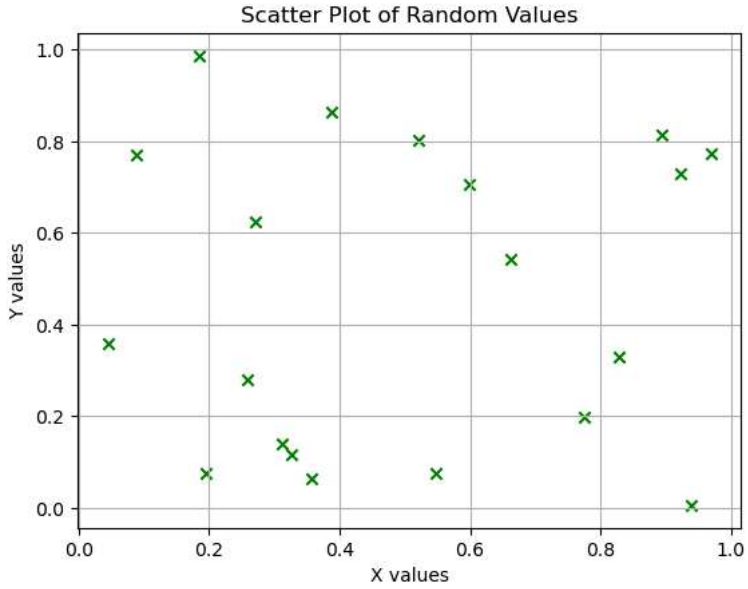

```
import numpy as np

data = np.random.randint(0, 101, size=50)

plt.hist(data, bins=10, color='orange', edgecolor='black')
plt.title('Histogram of Random Numbers (0–100)')
plt.xlabel('Value Range')
plt.ylabel('Frequency')
plt.show()
```




```
activities = ['Sleep', 'Study', 'Work', 'Exercise', 'Leisure']
time_spent = [8, 4, 8, 2, 2]  # hours per day

plt.pie(time_spent, labels=activities, autopct='%1.1f%%', startangle=90)
plt.title('Daily Activities Breakdown')
plt.axis('equal')  # Makes the pie chart circular
plt.show()
```




```
x = np.random.rand(20)
y = np.random.rand(20)

plt.scatter(x, y, color='green', marker='x')
plt.title('Scatter Plot of Random Values')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.grid(True)
plt.show()
```



Start coding or generate with AI.