VINAY A N

1NT22ME444

QUANTITATIVE AND SOFTWARE PROGRAMMING (21MEA753) LA - 01

UNIT-01

1) LINE PLOT

Write a Python script to plot the population growth of a country over the years using a line plot. The years are from 2010 to 2020 and the population data is given as:

Years: [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

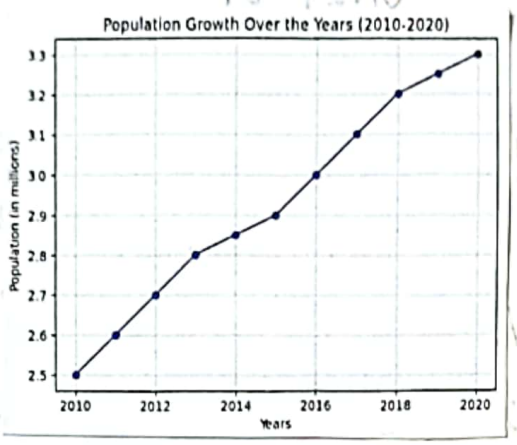
Population (in millions): [2.5, 2.6, 2.7, 2.8, 2.85, 2.9, 3.0, 3.1, 3.2, 3.25, 3.3]

Include appropriate labels for the X & Y axis, title, & grid lines for better readability.

->

```
import matplotlib.Pyplot as plt
Years = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
         2020]
Population = [2.5, 2.6, 2.7, 2.8, 2.85, 2.9, 3.0, 3.1,
              3.2, 3.25, 3.3]
plt.plot(years, Population, marker='o' linestyle='-' color='b')
plt.xlabel('Years')
plt.ylabel('Population (in millions)')
```

plt.title ('Population Growth over the Years (2010-2020)')
plt.grid (True)
plt.show().



2) Bar Chart

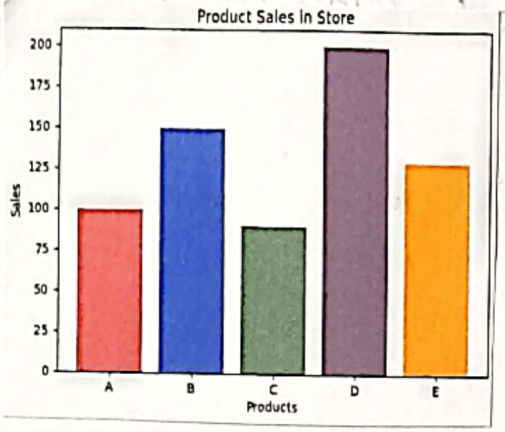
Write a Python script to create a bar chat showing the sales of different products in a store. The products and their corresponding sales are as follow

Products : ['A', 'B', 'C', 'D', 'E']

Sales : [100, 150, 90, 200, 130]

Ensure that each bar has a unique color, label X-axis as 'Products', the Y-axis as 'Sales' & give the chart a title.

```
import matplotlib.pyplot as plt
products = ['A', 'B', 'C', 'D', 'E']
Sales = [100, 150, 90, 200, 130]
plt.bar(products, Sales, color = ['red', 'blue', 'green', 'purple',
                                  'orange'])
plt.xlabel('Products')
plt.ylabel('Sales')
plt.title('Product Sales in Store')
plt.show()
```



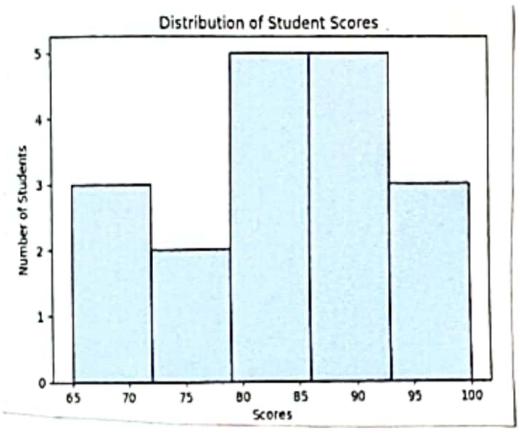
3) HISTOGRAM

Write a Python Script to plot a histogram of student scores in a test. The scores are as follow:

Scores : [88, 92, 100, 67, 85, 79, 95, 80, 78, 90, 85, 99, 76, 65, 70, 91, 87, 82]

Set the number of bins to 5 and make sure to label the X-axis as 'Score' & the Y axis as 'Number of Students'. Give the plot an appropriate title.

```
import matplotlib.pyplot as plt
Scores = [88, 92, 100, 67, 85, 79, 95, 80, 78, 90, 85, 99, 76, 65,
          70, 91, 87, 82]
plt.hist(Scores, bins = 5, edgecolor = 'black', color = 'skyblue')
plt.xlabel('Scores')
plt.ylabel('Number of Student Scores')
plt.show()
```

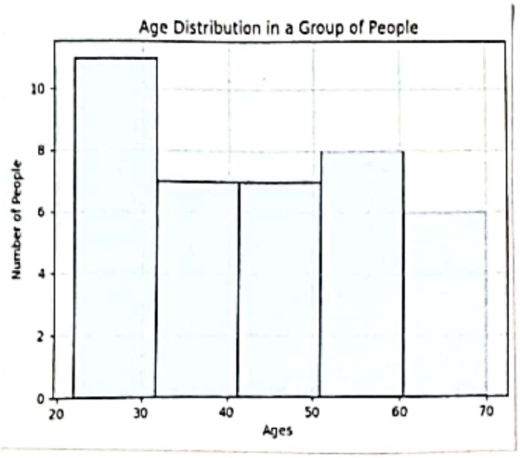


Write a Python Script to create a histogram that shows the distribution of ages in a group of 40 people.
The age data is as follow:

Ages: [22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35
40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 30, 55,
60, 65, 70, 22, 30, 40, 45, 55, 60]

Create a histogram with 5 bins.

```
import matplotlib.pyplot as plt
Ages = [22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35
        40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55,
        60, 65, 70, 22, 30, 40, 45, 55, 60]
plt.hist(Ages, bins=5, edgecolor='black', color='lightblue')
plt.xlabel('Ages')
plt.ylabel('Number of People')
plt.title('Age Distribution in a Group of People')
plt.grid(True)
plt.show()
```



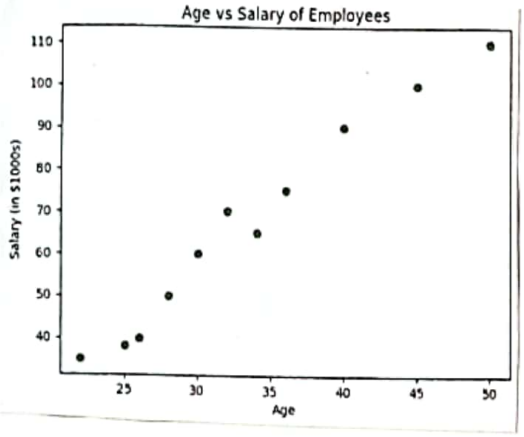
5) SCATTER PLOT

Write a Python Script to create a scatter plot for the relationship between the age of employees and their Salaries in a company. The data is given as :

Ages : [22, 25, 26, 28, 30, 32, 34, 36, 40, 45, 50]

Salaries (in $1000s) : [35, 38, 40, 50, 60, 70, 65, 75, 90, 100, 110]

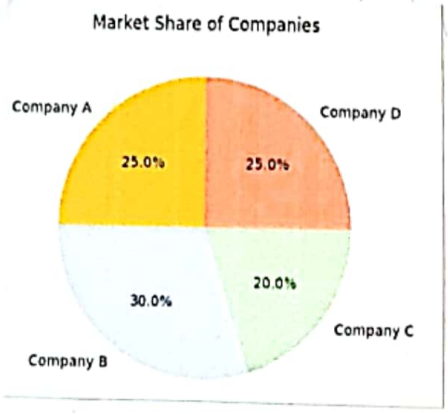
```
import matplotlib.Pyplot as plt.
ages = [22, 25, 26, 28, 30, 32, 34, 36, 40, 45, 50]
Salaries = [35, 38, 40, 50, 60, 70, 65, 75, 90, 100, 110]
plt.Scatter (ages, Salaries, color = 'green')
plt.xlabel ('Age')
plt.ylabel ('Salary (in $1000s)')
plt.title ('Age Vs Salary of Employees')
plt.show ()
```



6) PIE CHART

Write a Python Program to create a pie chart representing the market share of different companies in a specific industry. The companies and their market share are given. Make sure to label each side/slice with the percentage & give the chart a title.

→
```
import matplotlib.pyplot as plt
Companies = ['Company A', 'Company B', 'Company C', 'Company D']
Market_Share = [25, 30, 20, 25]
plt.pie(market_shares, labels = companies, autopct = '%.1.1f%%',
        startangle = 90, colors = ['gold', 'lightblue', 'lightgreen',
                                   'coral'])
plt.title('Market Share of Companies')
plt.show()
```

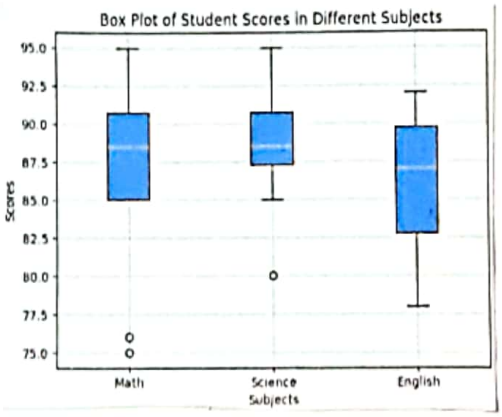


7) BOX PLOT

Write a Python script to create a box plot for the test scores of students in three different subjects: Math, Science and English. Plot a box plot for each subject & label the X-axis as 'Subjects' and the Y-axis as 'Scores'. Give the plot a title and display the outliers, if any.

→

```
import matplotlib.pyplot as plt
math_scores = [88, 92, 75, 85, 90, 95, 89, 91, 76, 85]
science_scores = [80, 85, 88, 92, 91, 87, 90, 95, 89, 88]
english_scores = [78, 82, 80, 85, 88, 90, 91, 86, 89, 92]

data = [math_scores, science_scores, english_scores]
plt.boxplot(data, patch_artist = True, labels = ['Math',
                'Science', 'English'])
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.title('Box plot of Student Scores in Different Subjects')
plt.grid(True)
plt.show()
```

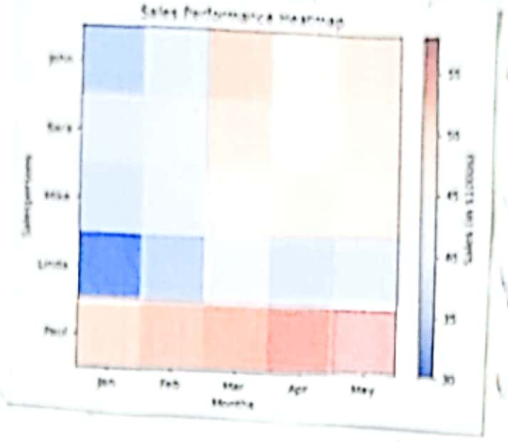


8) HEAT MAP

Write a Python script to create a heatmap representing the performance of different salespersons over five months. The sale data is given as.

⇒

```
import matplotlib.pyplot as plt
import numpy as np.
Salespersons = ['John', 'Sara', 'Mike', 'Linda', 'Paul']
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
Sales_data = [
    [35, 40, 50, 45, 48], # John
    [40, 42, 48, 44, 47], # Sara
    [38, 41, 45, 47, 46], # Mike
    [30, 35, 42, 38, 40], # Linda
    [50, 52, 53, 55, 58], # Paul
    ]
plt.imshow(Sales_data, cmap = 'coolwarm', interpolation =
            'nearest')
plt.colorbar(label = 'Sales (in $1000s)')
plt.xticks(ticks = np.arange(len(months)), labels = months)
plt.yticks(ticks = np.arange(len(Salespersons)), labels = Salesperson)
plt.xlabel('Months')
plt.ylabel('Sales Performance Heatmap')
plt.title('Salespersons')
plt.show()
```



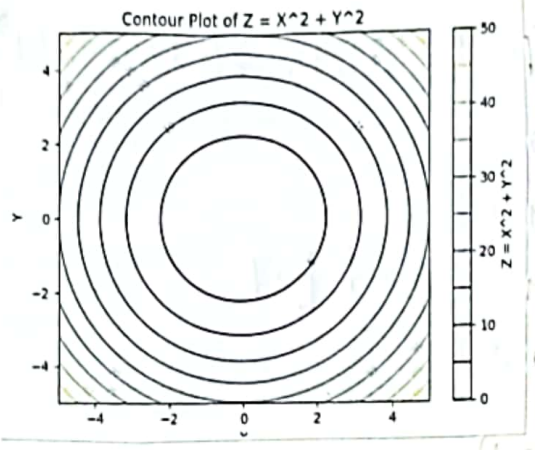
9) CONTOUR PLOT

Write a Python Script to create a contour plot of a function

$Z = f(X, Y)$, where $Z = X2 + Y2$. The X & Y values should range from -5 to 5. Generate a contour plot with 10 contour level & label both the axes and the contour lines.

```
import matplotlib.pyplot as plt
import numpy as np.
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + Y**2
plt.contour(X, Y, Z, levels = 10, cmap = 'viridis')
plt.colorbar(label = 'Z = X^2 + Y^2')
```

```
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Contour Plot of Z = X^2 + Y^2')
plt.clabel(plt.contour(X, Y, Z, levels=10, cmap='viridis'),
           inline=True, fontsize=8)
plt.show()
```



UNIT - 02

1) Calculate the mean, median, mode & standard deviation for the following array using Numpy & Scipy:

arr = np.array([15, 18, 20, 18, 21, 17, 20, 19, 15, 16])

→

```
import numpy as np
from scipy import stats
arr = np.array([15, 18, 20, 18, 21, 17, 20, 19, 15, 16])
mean = np.mean(arr)
median = np.median(arr)
mode = stats.mode(arr).mode[0]
std_dev = np.std(arr)

print('mean = {mean}')
print('Median = {median}')
print('Mode = {mode}')
print('Standard Deviation = {std_dev}')
```

Output:

Mean = 17.9
Median = 18.0
Mode = 15
Standard Deviation = 2.0223

2) Create a 2D Numpy array as follows:

arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

Calculate the mean, median, mode & Standard deviation of the array along both axes.

```
import numpy as np
from scipy import stats
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
mean_rows = np.mean(arr_2d, axis=1)
mean_columns = np.mean(arr_2d, axis=0)
median_rows = np.median(arr_2d, axis=1)
median_columns = np.median(arr_2d, axis=0)
mode_rows = stats.mode(arr_2d, axis=1)
mode_columns = stats.mode(arr_2d, axis=0)
std_rows = np.std(arr_2d, axis=1)
std_columns = np.std(arr_2d, axis=0)

print("Mean Rows:", mean_rows)
print("Mean Columns:", mean_columns)
print("Median Rows:", median_rows)
print("Median Columns:", median_columns)
print("Mode Rows:", mode_rows)
print("Mode Columns:", mode_columns)
print("Standard Deviation:", std_rows)
```

print("Standard Deviation Columns", std_columns).

Output:

Mean Rows : [2. 5. 8.]

Mean Columns : [4. 5. 6.]

Median Rows : [2. 5. 8.]

Median Columns : [4. 5. 6.]

Mode Rows : [1. 4. 7.]

Mode Columns : [1. 2. 3.]

Standard Deviation Rows : [0.8164 0.8164 0.8164]

Standard Deviation Columns : [2.449 2.449 2.449]

Suppose you have the following 2D array where some data points are missing (np.nan):

data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])

Calculate the mean, median, mode & SD. Explain how missing data can affect statistical analysis.

→ Missing values are handled with np.nanmean, np.nanmedian etc. Missing data can skew results; it's crucial to address it appropriately by imputing, removing or using functions that ignore NaN.

```
import numpy as np
from scipy import stats
data = np.array([[1, 2, np.nan], [4, np.nan, 6],
                 [7, 8, 9]])
mean = np.nanmean(data)
Median = np.nanmedian(data)
mode = stats.mode(data[~np.isnan(data)])[0]
std_dev = np.nanstd(data)
Print(Mean : {mean})
Print(Median : {Median})
print(Mode : {mode})
print(Standard Deviation : {std_dev})
```

Output:

Mean : 5.28571

Median : 6.0

Mode : ~~Med~~ nan

Standard Deviation : 2.8139

Two Groups of students took the same test, & their Scores given by the following 1D arrays.

group_A = np.array([85, 87, 90, 92, 88, 86, 84, 91, 89, 93])

group_B = np.array([78, 85, 92, 90, 83, 87, 95, 70, 91, 88])

→ import numpy as np

group_A = np.array([85, 87, 90, 92, 88, 86, 84, 91, 89, 93])

group_B = np.array([78, 85, 92, 90, 83, 87, 95, 70, 91, 88])

mean_A = np.mean(group_A)

median_A = np.median(group_A)

mode_A = stats.mode(group_A)

std_dev_A = np.std(group_A)

mean_B = np.mean(group_B)

median_B = np.median(group_B)

mode_B = stats.mode(group_B)

std_dev_B = np.std(group_B).

print(f"Mean_A: {mean_A}, Mean_B: {mean_B}")

print(f"Median_A: {median_A}, Median_B: {median_B}")

print(f"Mode_A: {mode_A}, Mode_B: {mode_B}")

print(f"Standard_Deviation_A: {std_dev_A}, Standard_Deviation_B: {std_dev_B}")

Output:

Mean_A : 88.5 , Mean_B : 85.9

Median_A : 88.5 , Median_B : 87.5

Mode_A : 84 , Mode - B : 70

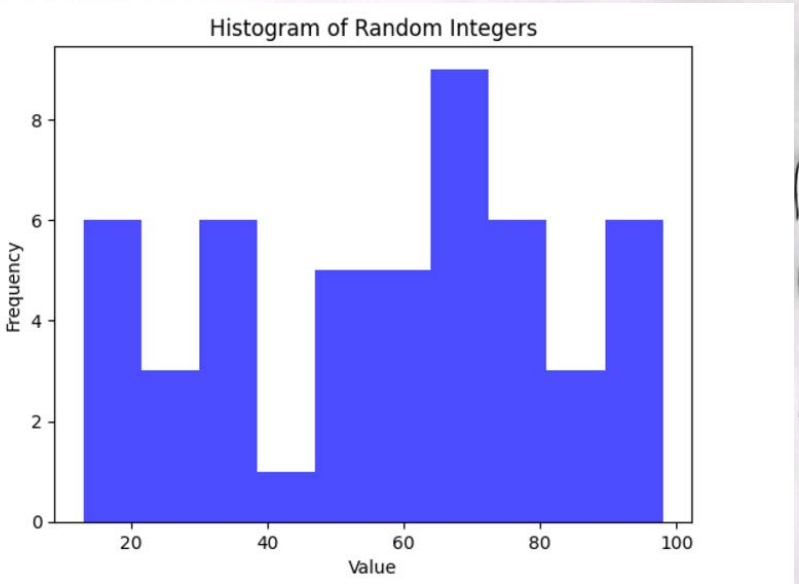
Standard_Deviation_A : 2.8722 , Standard_Deviation_B : 7.0206

Create a 1D Numpy array with 50 random integers between 10 & 100 using python code

arr = np.random.randint(10, 100, size = 50).

Using Python code calculate the mean, median, mode & standard deviation of the array. Additionally, plot a histogram of the array to visualize its distribution.

```
arr = np.random.randint(10, 100, size=50)
mean = np.mean(arr)
median = np.median(arr)
mode = stats.mode(arr).mode
std_dev = np.std(arr)
Print(f"Mean : {mean}, Median : {median}, Mode : {mode},
      Standard Deviation : {std_dev}")
plt.hist(arr, bins=10, alpha=0.7, color='blue')
plt.title("Histogram of Random Integers")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



Pandas Question

Create a Pandas Series as follows:

data = pd.Series([10, 12, 14, 13, 15, 16, 14, 18, 19, 20])

Write a Python program that calculates the mean, median, mode & standard deviation for this Series.

```
import pandas as pd
data = pd.Series([10, 12, 14, 13, 15, 16, 14, 18, 19, 20])
mean = data.mean()
median = data.median()
mode = data.mode()
std_dev = data.std()
variance = data.var()
print(f"Mean: {mean}, Median: {median}, Mode: {mode},
      Standard Deviation: {std_dev}, Variance: {variance}")
```

Output:

Mean: 15.1, Median: 14.5, Standard Deviation: 3.17804,
Variance: 10.100

Basic Statistical Measures on a pandas DataFrame:

Given the following pandas DataFrame representing the ages and heights of 5 individuals:

data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170]}

```
import pandas as pd
data = {'Age':[23,25,31,22,35], 'Height':[167,175,180,165,170]}
df = pd.DataFrame(data)
mean = df.mean()
median = df.median()
mode = df.mode().iloc[0]
std_dev = df.std()
print("Mean:", mean)
print()
print("Median:", median)
print()
print("Mode:", mode)
print()
print("Standard Deviation:", std_dev)
```

Output

Mean:

Age 27.2
Height 171.6
dtype float64

Median:

Age 25.0
Height 170.0
dtype float64

Mode:

Age 27
Height 165
dtype int64

Standard Deviation

Age 5.585696
Height 6.1073
dtype float64

Handling Missing Data in Pandas Data Frame.

Consider the following pandas Data Frame with some missing data (NaN values):

data = {'Age': [23, 25, None, 22, 35], 'Height': [167, 175, 180, None, 170], 'Weight': [65, 70, None, 68, 72]}

→ data = {'Age': [23, 25, None, 22, 35], 'Height': [167, 175, 180, None, 170], 'Weight': [65, 70, None, 68, 72]}

```
df = pd.DataFrame(data)
mean = df.mean()
median = df.median()
std_dev = df.std()
print("Mean:", mean)
print("Median:", median)
print("Standard Deviation:", std_dev)
```

Output

Mean:

Age 26.25
Height 173.00
dtype: float64

Median :

Age	24.0
Height	172.5
Weight	69.0
dtype :	float64

Standard Deviation.

Age	5.9651
Height	5.7154
Weight	2.9860
dtype :	float64

Column - Wise & Row - Wise Statistical Calculations :

Consider the following Dataframe representing the marks scored by 3 students in 4 subjects.

```
data = {'Math': [85, 92, 78], 'English': [88, 79, 85], 'Science':
        [90, 87, 92], 'History': [80, 70, 88]}.
df = pd.DataFrame(data, index = ['Student 1', 'Student 2',
                                 'Student 3'])
```

```
import Pandas as pd
data = {
    'Math': [85, 92, 78],
    'English': [88, 879, 83],
    'Science': [90, 87, 97],
    'History': [80, 70, 88]
}
df = pd.DataFrame(data, index=['Student1', 'Student2',
                               'Student3'])
Print(df)
Print()
col_mean = df.mean()
col_median = df.median()
col_std = df.std()

row_mean = df.mean(axis=1)
row_Median = df.median(axis=1)
row_Std = df.std(axis 1)

print("Column-Wise Statistics:")
print()
print("Column Mean:", col_mean)
print("Column Median:", col_median)
```

```
print ("Column Standard Deviation:", col_std)
print ()
print ("Row-Wise Statistics:")
print ("Row Mean:", row_mean)
print ("Row Median:", row_median)
print ("Row Standard Deviation:", row_std)
```

Output:

	Math	English	Science	History
Student1	85	88	90	80
Student2	92	79	87	70
Student3	78	85	92	88

Column-Wise Statistics:

Mean:

Math	85
English	84
Science	89.667
History	79.333

dtype: float64

Median

Math	85
English	86
Science	90
History	80

dtype : float64

Standard Deviation

Math	7.00
English	4.5825
Science	2.5166
History	9.0185.

dtype : float64

Row-Wise Statistics:

Mean:

Student 1	85.75
Student 2	82.00
Student 3	85.75

dtype: float64

Median:

Student 1	86.5
student 2	83.0
Student 3	86.5

dtype : float64

~~Student~~

Standard Deviation:

Student 1 4.349
Student 2 9.626
Student 3 5.909
dtype: float64

Understanding Bernoulli Trials:

Define a Bernoulli trial and give an example of a real world scenario where Bernoulli trials can be applied. Write a Python program that simulates 10 independent Bernoulli trials with a success probability of 0.6 & computes the mean & variance of the outcomes.

→
```
import numpy as np
P_success = 0.6
n_trials = 10
trials = np.random.binomial(1, P_success, n_trials)
mean_outcome = np.mean(trials)
Variance_outcome = np.var(trials)
print(trials)
print(mean_outcome)
print(Variance_outcomes)
```

[1 1 0 1 0 0 1 0 0 0]

0.4

0.24005

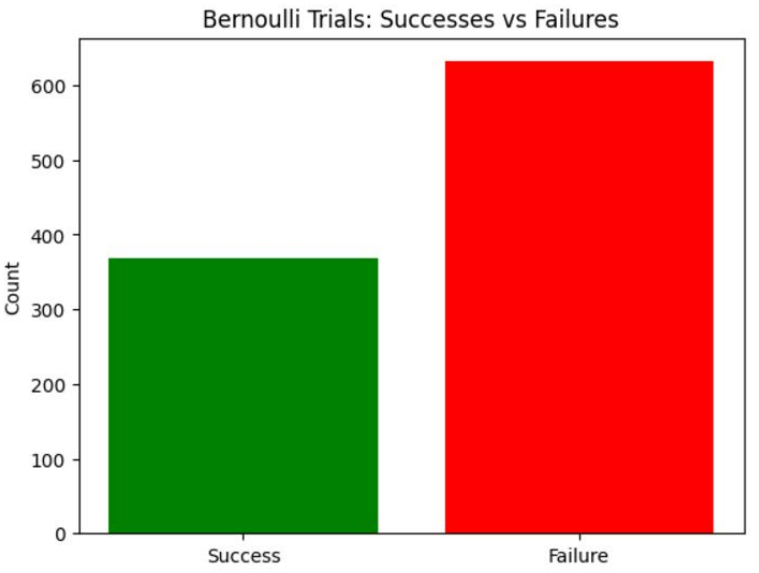
Write a Python Program that simulates 1000 Bernoulli trials with a success probability of 0.4 using Numpy. Calculate the proportion of successes and compare it with the theoretical probability. Plot a bar chart to visualize the number of successes & failures. Explain the relationship b/w the simulated results & the theoretical probability of success.

```
import numpy as np
import matplotlib.pyplot as plt

p_success = 0.4
n_trials = 1000
trials = np.random.binomial(1, p_success, n_trials)
proportion_success = np.mean(trial)

theoretical_prob = p_success
n_success = np.sum(trials)
n_failure = n_trials - n_success

print(f"Number of successes: {n_success}")
print(f"Number of failures: {n_failure}")
print(f"Number of
```

```
print(f"Proportion of successes (Simulated): {proportion_success
                                                    ")
print(f"Theoretical probability of success:
                                    {theoretical_prob}")
# plot the bar chard
labels = ['Success', 'Failure']
counts = [n_success, n_failure]
plt.bar(labels, counts, color = ['green', 'red'])
plt.ylabel('Count')
plt.title('Bernoulli Trials: Successes Vs Failures')
plt.show()
```

Output

Number of successes: 368
Number of failures: 632
Proportion of successes (Simulated): 0.368
Theoretical probability of success: 0.4


Bernoulli Trials: Successes vs Failures
Count
0
100
200
300
400
500
600
Success
Failure