#111111111111111111

def fun(name):

print(f"Hi, {name}")

fun("Ali")

def equation(x, y):

return (x \* 2 + y \* 2 + 2 \* x \* y)

equation(3, 4)

equation(-3, 2)

def sq(x):

return x \* x

sq(5)

def cal(x, y):

return x + y, x \* y

cal(10, 15)

def rect\_a\_p(length, width):

area = length \* width

perimeter = 2 \* (length + width)

return area, perimeter

rect\_a\_p(2, 6)

length = 5

width = 2

area, perimeter = rect\_a\_p(length, width)

print(f"area : {area}")

print(f"perimeter : {perimeter}")

import math

def area\_circle(radius):

return math.pi \* radius \* 2

area\_circle(3)

def even(n):

return n % 2 == 0

print(even(4))

print(even(7))

def m(a, b):

return a if a > b else b

print(m(7, 12))

def fact(n):

if n == 0:

return 1

return n \* fact(n - 1)

print(fact(5))

def count\_vowels(text):

vowels = "aeiouAEIOU"

return sum(1 for char in text if char in vowels)

print(count\_vowels("Statistics and Data Science"))

print(count\_vowels("Jahangirnagar University"))

def rev\_string(st):

return st[::-1]

print(rev\_string("python"))

def count\_words(sentence):

return len(sentence.split())

print(count\_words("More para in the Dept of SDS"))

def count\_charecter(text, ch):

return text.count(ch)

print(count\_charecter("mango", "a"))

print(count\_charecter("Jahangirnagar", "a"))

def sum\_list(lst):

return sum(lst)

print(sum\_list([1, 2, 3, 4, 5]))

def largest(lst):

return max(lst)

print(largest([4, 10, 20, 50, 34]))

def ave(nums):

return sum(nums) / len(nums)

print(ave([10, 20, 30]))

def check\_number(n):

if n > 0:

return "Positive"

elif n < 0:

return "Negaive"

else:

return "Zero"

print(check\_number(-5))

print(check\_number(5))

print(check\_number(0))

#2222222222222222222222222

import numpy as np

data=np.array([5,15,10,20,25,30,25,35])

#Mean

mean=np.mean(data)

print('Mean:',mean)

#Median

median=np.median(data)

print('Median:',median)

#Mode

from scipy import stats

mode=stats.mode(data)

print("Mode:",mode[0])

#Show dictionary

dir(np)

import scipy as sc

dir(sc)

#SD

std\_dev=np.std(data)

print("Standard Deviation:",std\_dev)

#Variance

variance=np.var(data)

print('Variance',variance)

#Skewnwess ,Kurtosis

from scipy.stats import skew,kurtosis

Skewness=stats.skew(data)

Kurtosis=stats.kurtosis(data)

print('Skewness:',Skewness)

print('Kurtosis:',Kurtosis)

X=np.array([1,2,3,4,5])

Y=np.array([2,4,5,4,5])

correlation\_matrix=np.corrcoef(X,Y)

correlation= correlation\_matrix[0,1]

print("Pearson correlation coefficient:",correlation)

mean\_X= np.mean(X)

mean\_Y= np.mean(Y)

#333333333333333333333333333333333

import matplotlib.pyplot as plt

# Sample data

x = [1, 2, 3, 4, 5]

y = [5, 7, 4, 6, 8]

# Create scatter plot

plt.scatter(x, y, color='blue', marker='v', label='Data Points')

# Connect points with a line

plt.plot(x, y, color='red', linestyle='-', linewidth=1, label='Connecting Line')

# Add value labels for each point

for i in range(len(x)):

plt.text(x[i], y[i], f'({x[i]},{y[i]})', fontsize=9, ha='left', va='top')

# Add labels, title, and legend

plt.xlabel('X Axis')

plt.ylabel('Y Axis')

plt.title('Scatter Plot with Connecting Line')

plt.legend()

# Show plot

plt.show()

#####two scatter line in a plot

import matplotlib.pyplot as plt

# Data

x = [1, 2, 3, 4, 5]

y = [5, 7, 4, 6, 8]

z = [2, 4, 6, 7, 8]

# Plot x vs y

plt.scatter(x, y, color='blue', marker='o', label='x vs y')

plt.plot(x, y, color='blue', linestyle='-', linewidth=1)

# Plot x vs z

plt.scatter(x, z, color='red', marker='^', label='x vs z')

plt.plot(x, z, color='red', linestyle='--', linewidth=1)

# Labels and title

plt.xlabel('x')

plt.ylabel('Values')

plt.title('Scatter Plot with Two Lines')

plt.legend()

plt.show()

###Histogram

import matplotlib.pyplot as plt

import numpy as np

import math

# Sample data

data = [12, 15, 13, 17, 19, 22, 22, 23, 25, 26, 26, 29, 30, 33, 35]

# Calculate bins using Sturges' formula

n = len(data)

bins = math.ceil(np.log2(n) + 1)

# Plot histogram and get counts and bin edges

counts, bin\_edges, patches = plt.hist(data, bins=bins, color='skyblue', edgecolor='black')

# Calculate bin midpoints for labeling

bin\_mids = (bin\_edges[:-1] + bin\_edges[1:]) / 2

# Add axis labels and title

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title(f'Histogram with bin midpoints labeled (bins={bins})')

# Set new x-axis labels

plt.xticks(bin\_mids, [f'{mid:.1f}' for mid in bin\_mids])

plt.show()

#### Bar Diagram

import matplotlib.pyplot as plt

from collections import Counter

# Your categorical data

data = ['A', 'A', 'B', 'A', 'C', 'A', 'B', 'C', 'D', 'E', 'D', 'F', 'E', 'F', 'C', 'D', 'A', 'F', 'A']

# Count frequency of each category

freq = Counter(data)

# Separate the categories and their counts

categories = list(freq.keys())

values = list(freq.values())

# Create bar plot

plt.bar(categories, values, color='skyblue', edgecolor='red')

# Add labels and title

plt.xlabel('Category')

plt.ylabel('Frequency')

plt.title('Bar Diagram of Categorical Data')

# Show plot

plt.show()

#### Pie chart

import matplotlib.pyplot as plt

from collections import Counter

# Your categorical data

data = ['A', 'A', 'B', 'A', 'C', 'A', 'B', 'C', 'D', 'E', 'D', 'F', 'E', 'F', 'C', 'D', 'A', 'F', 'A']

# Count frequency of each category

freq = Counter(data)

# Separate the categories and their counts

labels = list(freq.keys())

sizes = list(freq.values())

# Create pie chart

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=plt.cm.Pastel1.colors)

# Equal aspect ratio ensures pie is a circle

plt.axis('equal')

plt.title('Pie Chart of Categorical Data')

# Show plot

plt.show()

import matplotlib.pyplot as plt

# Data

x = [1, 2, 3, 4, 5]

y = [5, 7, 4, 6, 8]

z = [2, 4, 6, 7, 8]

# Plot x vs y

plt.scatter(x, y, color='blue', marker='o', label='x vs y')

plt.plot(x, y, color='blue', linestyle='-', linewidth=1)

# Plot x vs z

plt.scatter(x, z, color='red', marker='^', label='x vs z')

plt.plot(x, z, color='red', linestyle='--', linewidth=1)

# Labels and title

plt.xlabel('x')

plt.ylabel('Values')

plt.title('Scatter Plot with Two Lines')

plt.legend()

plt.show()

#444444444444444444

#####Mean by Class

import pandas as pd

# Sample data

data = {

'Class': ['A', 'A', 'A', 'B', 'B', 'C', 'C'],

'Student': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7'],

'Score': [85, 90, 88, 75, 78, 92, 95]

}

df = pd.DataFrame(data)

print("Original Data:\n", df)

df['Mean by Group'] = df.groupby('Class')['Score'].transform(lambda x: x.mean())

print("\nMean within each class:\n", df)

######Standard deviation by Class

import pandas as pd

# Sample data

data = {

'Class': ['A', 'A', 'A', 'B', 'B', 'C', 'C'],

'Student': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7'],

'Score': [85, 90, 88, 75, 78, 92, 95]

}

df = pd.DataFrame(data)

print("Original Data:\n", df)

df['Standard deviation by Group'] = df.groupby('Class')['Score'].transform(lambda x: x.std())

print("\nSD within each class:\n", df)

######Mean and Standard deviation by Class

import pandas as pd

# Sample data

data = {

'Class': ['A', 'A', 'A', 'B', 'B', 'C', 'C'],

'Student': ['S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7'],

'Score': [85, 90, 88, 75, 78, 92, 95]

}

df = pd.DataFrame(data)

print("Original Data:\n", df)

df['Standard deviation by Group'] = df.groupby('Class')['Score'].transform(lambda x: x.std())

df['Mean by Class'] = df.groupby('Class')['Score'].transform(lambda x: x.mean())

print("\nSD and Mean within each class:\n", df)

######Mean by Class and Student Section

import pandas as pd

data = {

'Class': ['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'B', 'C', 'C'],

'section': ['S1', 'S2','S1', 'S2', 'S3', 'S4', 'S5', 'S3', 'S4','S5', 'S7','S7'],

'Score': [85, 90, 88, 75, 78, 92, 95, 85, 90, 88, 75, 78]

}

df = pd.DataFrame(data)

print("Original Data:\n", df)

df['Mean by Class and section'] = df.groupby(['Class', 'section'])['Score'].transform(lambda x: x.mean())

print("\nSD within each class:\n", df)

##### Contigency table

import pandas as pd

# Sample categorical data

data = {

'Gender': ['Male', 'Female', 'Female', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female', 'Male'],

'Class': ['A', 'A', 'B', 'B', 'A', 'B', 'A', 'A', 'B', 'B'],

'Passed': ['Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes']

}

df = pd.DataFrame(data)

print(" Sample Data:\n", df)

contingency\_table = pd.crosstab([df['Class'], df['Gender']], df['Passed'], margins=True)

print("\n Contingency Table (Class + Gender vs Passed):\n", contingency\_table)

#### Pivot table

import pandas as pd

data = {

'Class': ['A', 'A', 'B', 'B', 'A', 'B'],

'Gender': ['Male', 'Female', 'Female', 'Male', 'Male', 'Female'],

'Score': [85, 90, 88, 75, 78, 92]

}

df = pd.DataFrame(data)

pivot = pd.pivot\_table(df, values='Score', index='Class', columns='Gender', aggfunc='sum')

print("Pivot Table:\n", pivot)

##### Cross tab

crosstab = pd.crosstab(df['Gender'], df['Class'])

print("\nCross-Tabulation:\n", crosstab)

data2 = {

'Gender': ['Male', 'Female', 'Female', 'Male', 'Male', 'Female'],

'Passed': ['Yes', 'No', 'Yes', 'No', 'Yes', 'Yes']

}

df2 = pd.DataFrame(data2)

contingency = pd.crosstab(df2['Gender'], df2['Passed'])

print("\nContingency Table:\n", contingency)

###### Chi\_square table

import pandas as pd

from scipy.stats import chi2\_contingency

# Data

data2 = {

'Gender': ['Male', 'Female', 'Female', 'Male', 'Male', 'Female','Female', 'Male', 'Male', 'Female'],

'Passed': ['Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No','No','Yes','No']

}

df2 = pd.DataFrame(data2)

# Step 1: Create contingency table

contingency = pd.crosstab(df2['Gender'], df2['Passed'])

print("\n Contingency Table:\n", contingency)

# Step 2: Apply Chi-Square test

chi2, p, dof, expected = chi2\_contingency(contingency)

# Step 3: Print results

print("\n Chi-Square Test Results:")

print(f"Chi-square statistic = {chi2:.4f}")

print(f"Degrees of freedom = {dof}")

print(f"P-value = {p:.4f}")

print("\nExpected Frequencies:\n", pd.DataFrame(expected, index=contingency.index, columns=contingency.columns))

row\_percent = contingency.div(contingency.sum(axis=1), axis=0) \* 100

print("\nRow-wise Percentage (%):\n", row\_percent.round(2))

# Assuming 'contingency' is your contingency table

column\_percent = contingency.div(contingency.sum(axis=0), axis=1) \* 100

# Print the column-wise percentage, rounded to 2 decimal places

print("\nColumn-wise Percentage (%):\n", column\_percent.round(2))

combined = contingency.astype(str) + " (" + row\_percent.round(1).astype(str) + "%" ")"

print(" Contingency Table with Row-wise Percentages:\n", )

print(combined)