import numpy as np import matplotlib.pyplot
as plt

Step 1: Input x and y values x = np.array(list(map(float, input("Enter the values of x (separated by spaces): ").split()))) y = np.array(list(map(float, input("Enter the values of y (separated by spaces): ").split())))

Step 2: Calculate the slope (m) and intercept (b) n = len(x)

# Calculate sums sum_x = np.sum(x)
sum_y = np.sum(y) sum_x_squared
= np.sum(x**2) sum_xy = np.sum(x
* y)

Calculate slope (m) m = (n * sum_xy - sum_x * sum_y) / (n * sum_x_squared
- sum_x**2)

Calculate intercept (b) b =
(sum_y - m * sum_x) / n

Step 3: Print intermediate values and final results print(f"Sum of x:
{sum_x}")

print(f"Sum of y: {sum_y}") print(f"Sum of x^2: {sum_x_squared}") print(f"Sum of xy: {sum_xy}") print(f"Slope (m) = (n * sum(xy) - sum(x) * sum(y)) / (n * sum(x^2) - (sum(x))^2)") print(f"Slope (m) = ({n} * {sum_xy} - {sum_x} * {sum_y}) / ({n} * {sum_x_squared} - {sum_x}^2)")

print(f"Slope

(m) = {m:.2f}")

print(f"Intercept (b) = (sum(y) - m * sum(x)) / n") print(f"Intercept

(b) = ({sum_y} - {m:.2f} * {sum_x}) / {n}") print(f"Intercept (b) =

{b:.2f}")

Step 4: Print the equation of the line print(f"The equation of the regression

line is: y = {m:.2f}x + {b:.2f}")

Step 5: Plot the data points and the regression line plt.scatter(x, y, color='blue',

label='Data points') # Scatter plot for the data points plt.plot(x, m * x + b, color='red',

label='Regression line') # Plot the regression line plt.xlabel('x') plt.ylabel('y')

plt.legend() plt.title('Linear Regression: Data Points & Regression Line') plt.grid(True)

plt.show()

Enter the values of x (separated by spaces): 1 2 3 4 5 6 7 8 9

Enter the values of y (separated by spaces): 9 8 7 6 5 4 3 2 1

Sum of x: 45.0

Sum of y: 45.0

Sum of x^2: 285.0

Sum of xy: 165.0

Slope (m) = (n * sum(xy) - sum(x) * sum(y)) / (n * sum(x^2) - (sum(x))^2)

Slope (m) = (9 * 165.0 - 45.0 * 45.0) / (9 * 285.0 - 45.0^2)

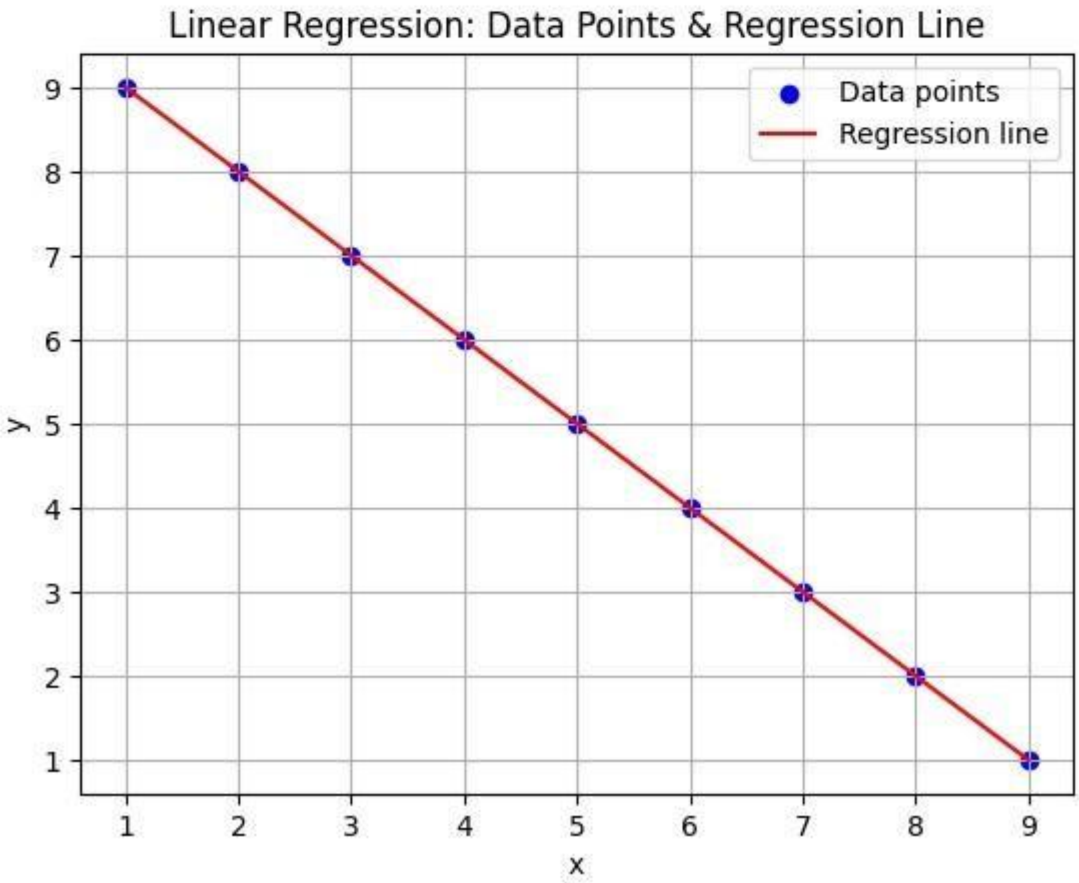
Slope (m) = -1.00

Intercept (b) = (sum(y) - m * sum(x)) / n

Intercept (b) = (45.0 - -1.00 * 45.0) / 9

Intercept (b) = 10.00

The equation of the regression line is: y = -1.00x + 10.00



import numpy as np import matplotlib.pyplot

as plt

Step 1: Input the number of independent variables (features) num_features =

int(input("Enter the number of features (independent variables): "))

Step 2: Input x values (features) and y values (dependent variable) x

= [] for i in range(num_features):

```
    x_i = list(map(float, input(f"Enter the values for x{i+1} (separated by spaces): ").split()))
x.append(x_i)

x = np.array(x).T  # Transpose the input to make each row a data point y =
np.array(list(map(float, input("Enter the values of y (dependent variable, separated by
spaces): ").split())))

# Step 3: Add a column of ones to x for the intercept term (bias)
X = np.c_[np.ones(x.shape[0]), x]

# Step 4: Calculate the coefficients using the Normal Equation
# The normal equation is: theta = (X^T * X)^-1 * X^T * y
X_transpose = X.T
X_transpose_X = X_transpose @ X
X_transpose_X_inv = np.linalg.inv(X_transpose_X)
X_transpose_y = X_transpose @ y theta =
X_transpose_X_inv @ X_transpose_y

# Step 5: Print the intermediate calculations  print("\nIntermediate
Calculations:") print(f"X^T
(Transposed X): \n{X_transpose}") print(f"X^T *
X: \n{X_transpose_X}") print(f"(X^T * X)^-1:
\n{X_transpose_X_inv}") print(f"X^T * y:
\n{X_transpose_y}") print(f"Coefficients (theta):
\n{theta}")

# Step 6: Print the regression equation
print("\nRegression Equation:") print("The
```

```
regression equation is: y = ", end="") for i in range(1,
len(theta)):
    print(f"{theta[i]:.2f}*x{i} + ", end="") print(f"{theta[0]:.2f}")
# Intercept term

# Step 7: Plotting
# Since it's difficult to visualize in higher dimensions, we'll just plot the first two features for
demonstration if num_features >= 2:
    fig = plt.figure()     ax = fig.add_subplot(111, projection='3d')     ax.scatter(x[:,
0], x[:, 1], y, color='blue', label='Data points')

    # Create a meshgrid for prediction surface     x0_vals = np.linspace(np.min(x[:,
0]), np.max(x[:, 0]), 10)     x1_vals = np.linspace(np.min(x[:, 1]), np.max(x[:, 1]),
10)
    X0, X1 = np.meshgrid(x0_vals, x1_vals)
    Z = theta[0] + theta[1]*X0 + theta[2]*X1

    # Plot the regression plane     ax.plot_surface(X0, X1, Z, color='red', alpha=0.5)
ax.set_xlabel('Feature 1')     ax.set_ylabel('Feature 2')     ax.set_zlabel('y')
plt.title('Multivariable Regression: Data points and Regression Plane')
plt.show()
```

Intermediate Calculations:

X^T (Transposed X):

[[1. 1. 1. 1. 1. 1. 1. 1. 1.]

[1. 2. 3. 4. 5. 6. 7. 8. 9.] [2.

3. 5. 6. 7. 8. 5. 6. 3.]] X^T *

X:

[[9. 45. 45.]

 [45. 285. 240.]

[45. 240. 257.]]

(X^T * X)^-1:

[[1.02556539 -0.05014749 -0.13274336]

 [-0.05014749 0.01887906 -0.00884956] [-0.13274336 -

0.00884956 0.03539823]]

X^T * y:

[84. 464. 452.]

Coefficients (theta):

[2.87905605 0.54749263 0.74336283]

Regression Equation:

The regression equation is: y = 0.55*x1 + 0.74*x2 + 2.88