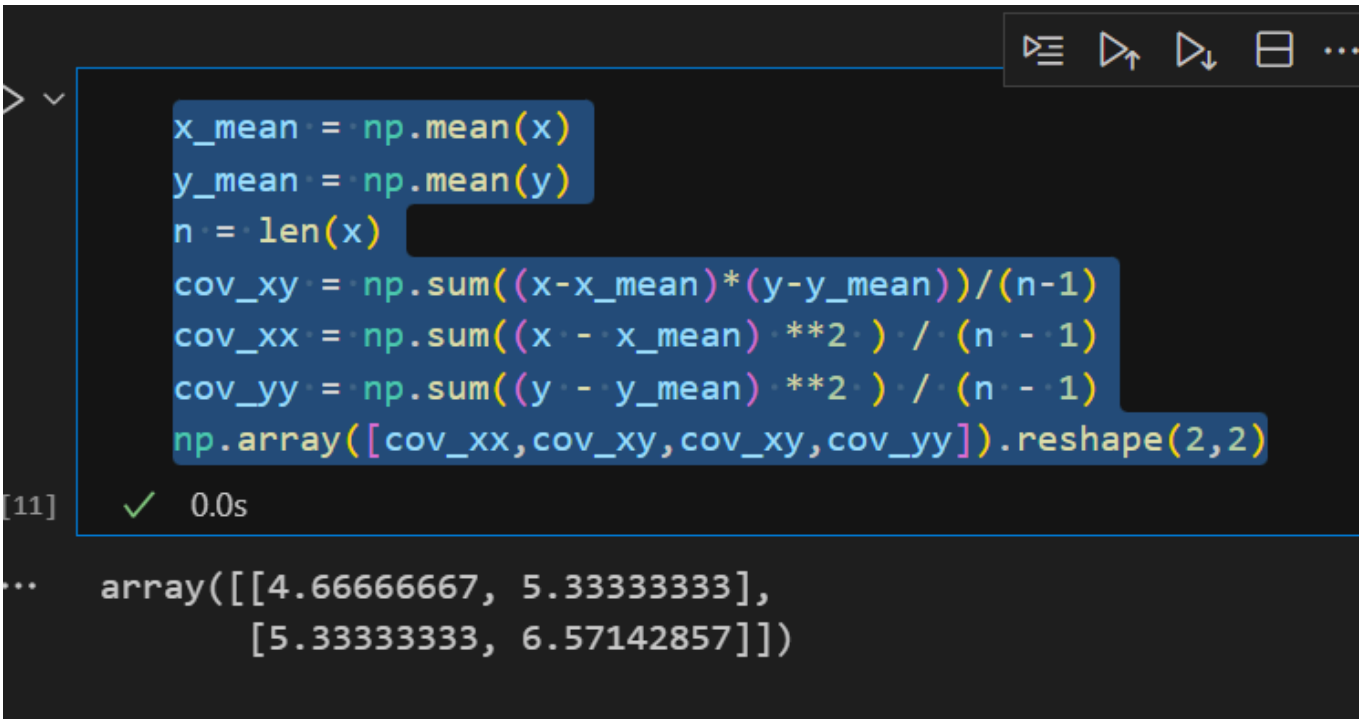
```
import numpy as np



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

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



np.cov(x, y)
```

```
x_mean = np.mean(x)

y_mean = np.mean(y)

n = len(x)

cov_xy = np.sum((x-x_mean)*(y-y_mean))/(n-1)

cov_xx = np.sum((x - x_mean) **2 ) / (n - 1)

cov_yy = np.sum((y - y_mean) **2 ) / (n - 1)

np.array([cov_xx,cov_xy,cov_xy,cov_yy]).reshape(2,2)
```



From Regression

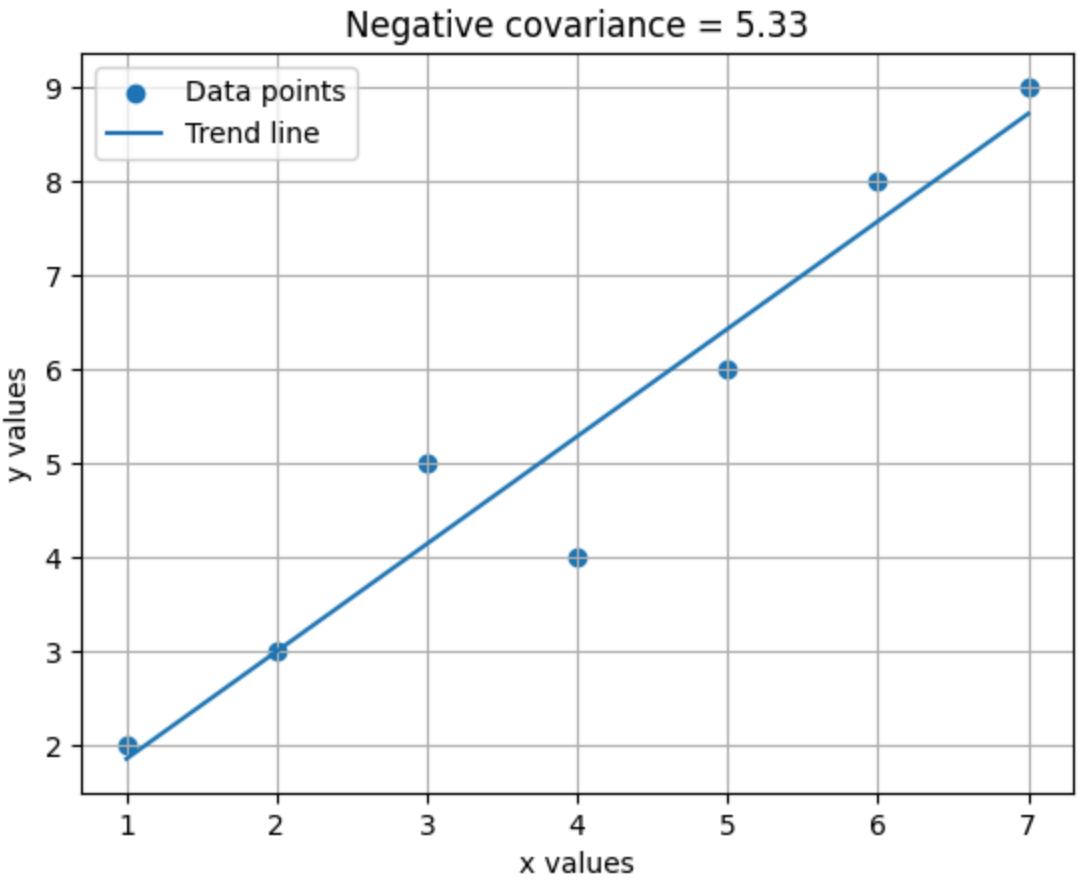
```
m = cov_xy / cov_xx


b = y_mean - m * x_mean


print(m,b)
```

From Polyfit (OLS)

```


slope, intercept = np.polyfit(x, y, 1)

y_line = slope * x + intercept

slope,intercept,y_line
```

```


plt.scatter(x, y, label="Data points")

plt.plot(x, y_line, label="Trend line")
```

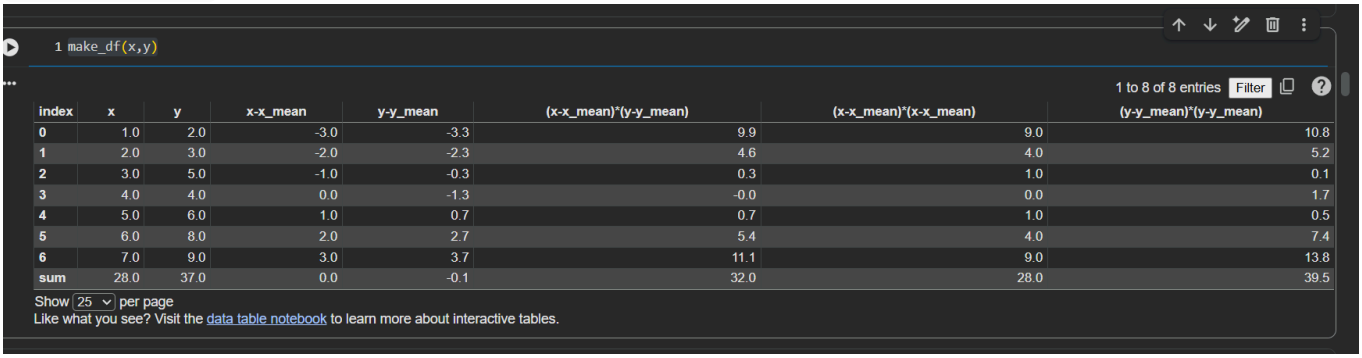
```
plt.title(f"Negative covariance = {cov_xy:.2f}")

plt.xlabel("x values")

plt.ylabel("y values")

plt.legend()

plt.grid(True)


plt.show()
```




```
plt.scatter(x, y, label="Data points")

plt.plot(x, y_line, label="Trend line")
```

```
plt.title(f" covariance = {cov_xy:.2f}")

plt.xlabel("x values")

plt.ylabel("y values")

plt.legend()

plt.grid(True)



plt.show()
```

`make_df(x,y)`



```
1 make_df(x,y)
```

1 to 8 of 8 entries Filter

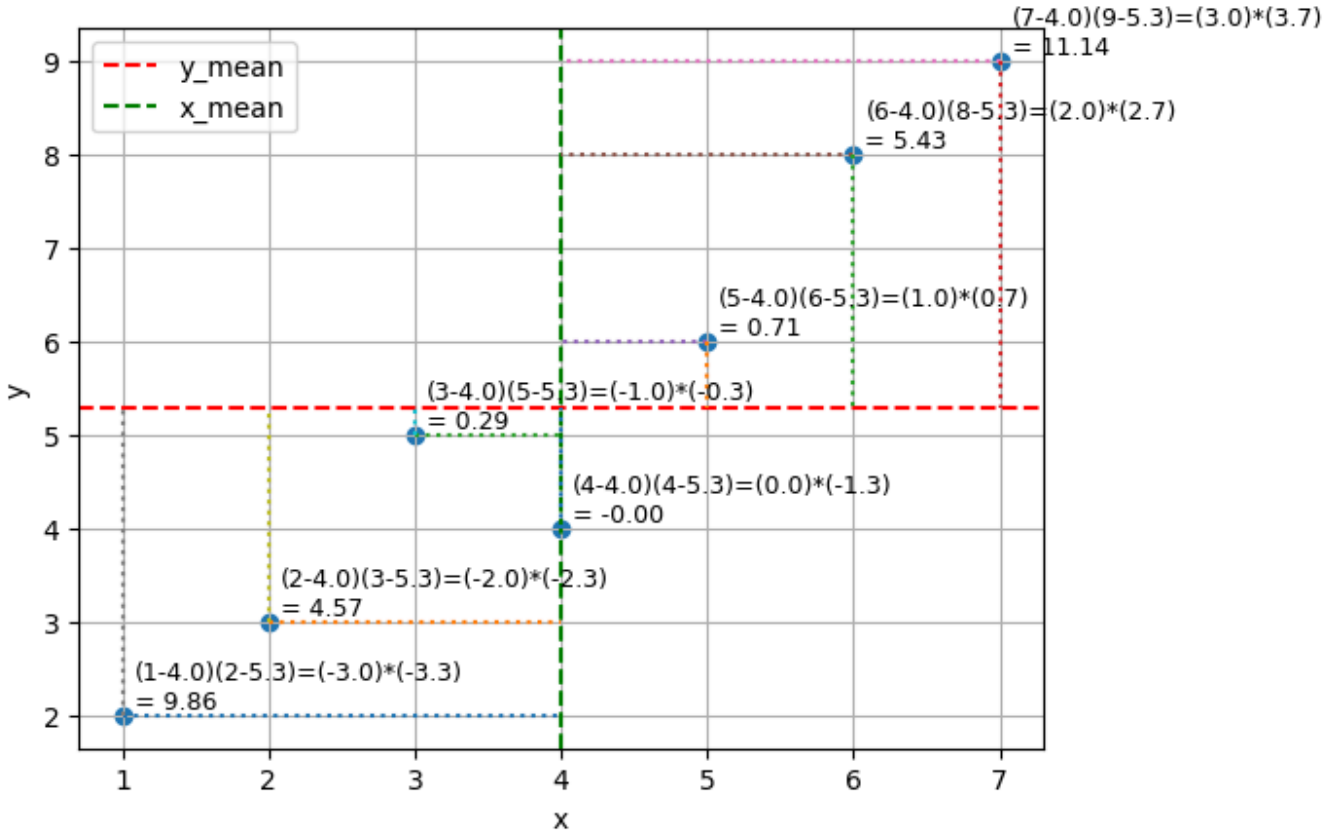
index	x	y	x-x_mean	y-y_mean	(x-x_mean)*(y-y_mean)	(x-x_mean)*(x-x_mean)	(y-y_mean)*(y-y_mean)
0	1.0	2.0	-3.0	-3.3	9.9	9.0	10.8
1	2.0	3.0	-2.0	-2.3	4.6	4.0	5.2
2	3.0	5.0	-1.0	-0.3	0.3	1.0	0.1
3	4.0	4.0	0.0	-1.3	-0.0	0.0	1.7
4	5.0	6.0	1.0	0.7	0.7	1.0	0.5
5	6.0	8.0	2.0	2.7	5.4	4.0	7.4
6	7.0	9.0	3.0	3.7	11.1	9.0	13.8
sum	28.0	37.0	0.0	-0.1	32.0	28.0	39.5

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

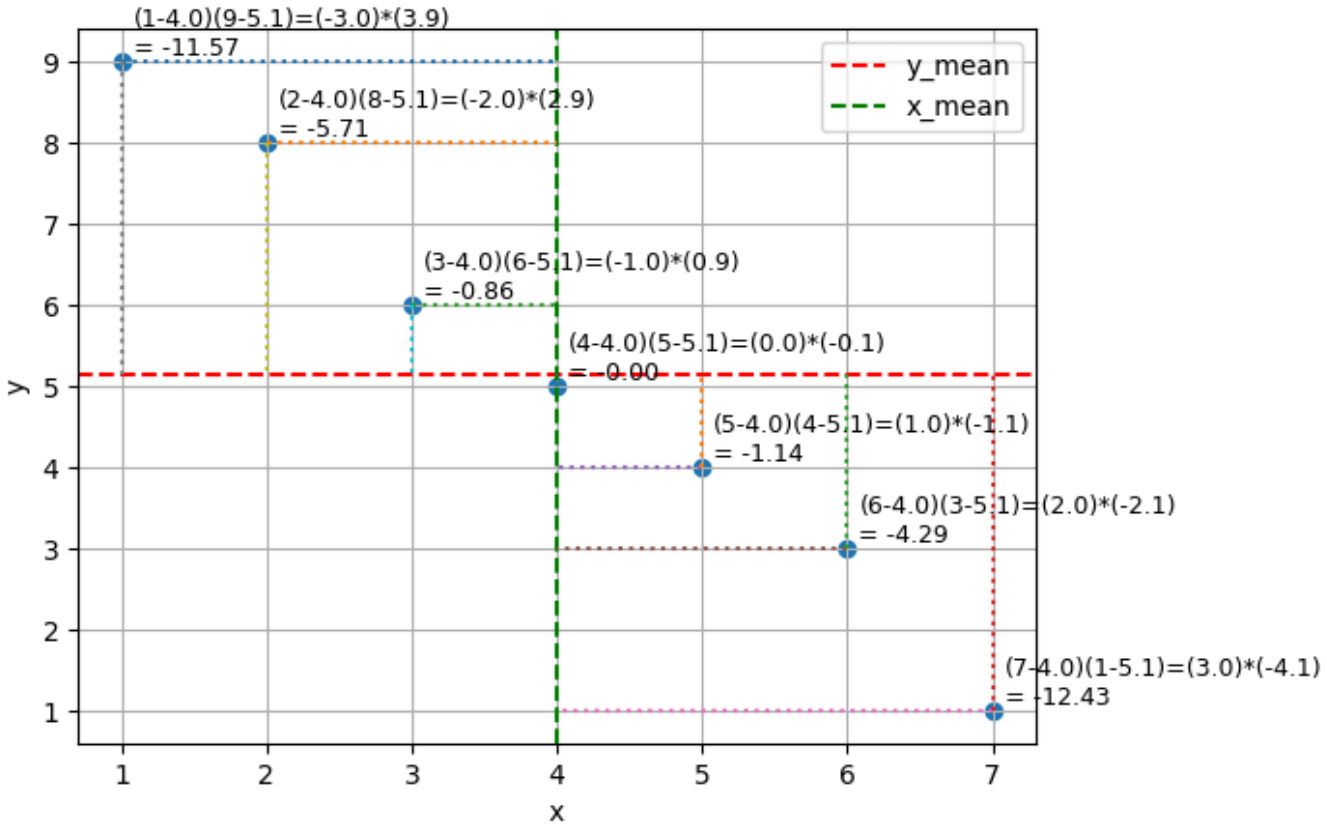
```
def plot_cov_contributions(x, y):

  x_mean = np.mean(x)

  y_mean = np.mean(y)

  x_minus_x_mean = x - x_mean

  y_minus_y_mean = y - y_mean

  contributions = x_minus_x_mean * y_minus_y_mean

  plt.xlabel('x')

  plt.ylabel('y')

  plt.axhline(y_mean,linestyle = '--',c='r',label='y_mean')

  plt.axvline(x_mean,linestyle = '--',c='g',label='x_mean')
```

```
  # Annotate each point with contribution

  for xi, yi, contribution in zip(x, y, contributions):

      plt.text(

          xi + 0.08,

          yi + 0.08,

          f"({xi:.0f}-{x_mean:.1f})({yi:.0f}-{y_mean:.1f})=({xi-x_mean})*({yi-
y_mean:.1f})\n= {contribution:.2f}",

          fontsize=9

      )



  # distance from x mean: horizontal distance

  for xi, yi in zip(x, y):

      plt.plot([xi, x_mean], [yi, yi], linestyle=":")




  # distance from y mean: vertical distance

  for xi, yi in zip(x, y):

      plt.plot([xi, xi], [yi, y_mean], linestyle=":")

  plt.grid(True)

  plt.legend()

  plt.scatter(x,y)
```

```
plot_cov_contributions(x,y)
```




```
y_negative = np.array([9, 8, 6, 5, 4, 3, 1])


plot_cov_contributions(x,y_negative)
```



`make_df(x,y_negative)`

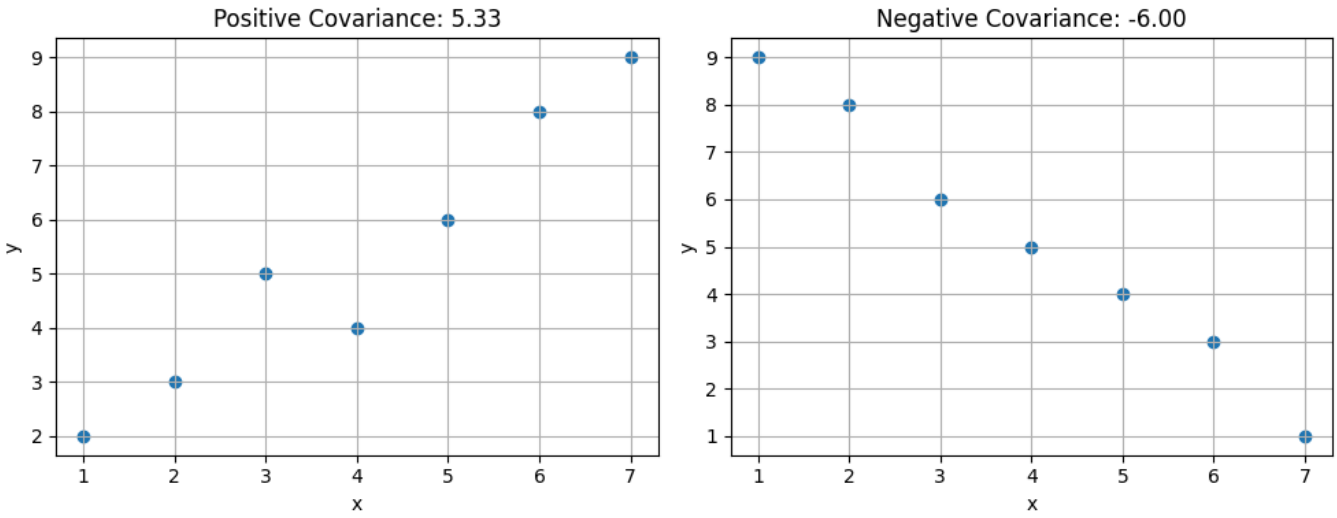
```
1 make_df(x,y_negative)
```

	x	y	x-x_mean	y-y_mean	(x-x_mean)*(y-y_mean)	(x-x_mean)*(x-x_mean)	(y-y_mean)*(y-y_mean)
0	1.0	9.0	-3.0	3.9	-11.6	9.0	14.9
1	2.0	8.0	-2.0	2.9	-5.7	4.0	8.2
2	3.0	6.0	-1.0	0.9	-0.9	1.0	0.7
3	4.0	5.0	0.0	-0.1	-0.0	0.0	0.0
4	5.0	4.0	1.0	-1.1	-1.1	1.0	1.3
5	6.0	3.0	2.0	-2.1	-4.3	4.0	4.6
6	7.0	1.0	3.0	-4.1	-12.4	9.0	17.2
sum	28.0	36.0	0.0	0.3	-36.0	28.0	46.9

```
1
```

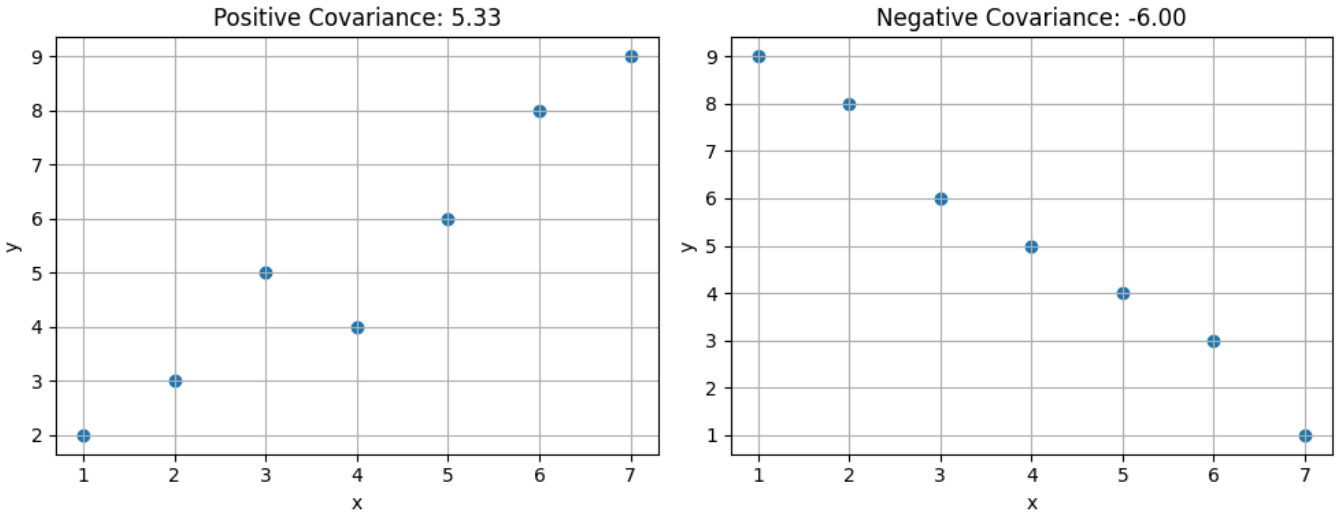
```
import numpy as np

import matplotlib.pyplot as plt



cov_positive = np.cov(x, y)[0, 1]
```

```
cov_negative = np.cov(x, y_negative)[0, 1]



plt.figure(figsize=(10, 4))



plt.subplot(1, 2, 1)

plt.scatter(x, y)

plt.title(f"Positive Covariance: {cov_positive:.2f}")

plt.xlabel("x")

plt.ylabel("y")

plt.grid(True)



plt.subplot(1, 2, 2)

plt.scatter(x, y_negative)

plt.title(f"Negative Covariance: {cov_negative:.2f}")

plt.xlabel("x")

plt.ylabel("y")

plt.grid(True)



plt.tight_layout()

plt.show()
```




```
import numpy as np

import matplotlib.pyplot as plt



cov_xy = np.cov(x, y)[0, 1]



slope, intercept = np.polyfit(x, y, 1)

y_line = slope * x + intercept



plt.scatter(x, y, label="Data points")

plt.plot(x, y_line, label="Trend line")



plt.xlabel("x values")

plt.ylabel("y values")

plt.legend()

plt.grid(True)
```

```
plt.show()
```



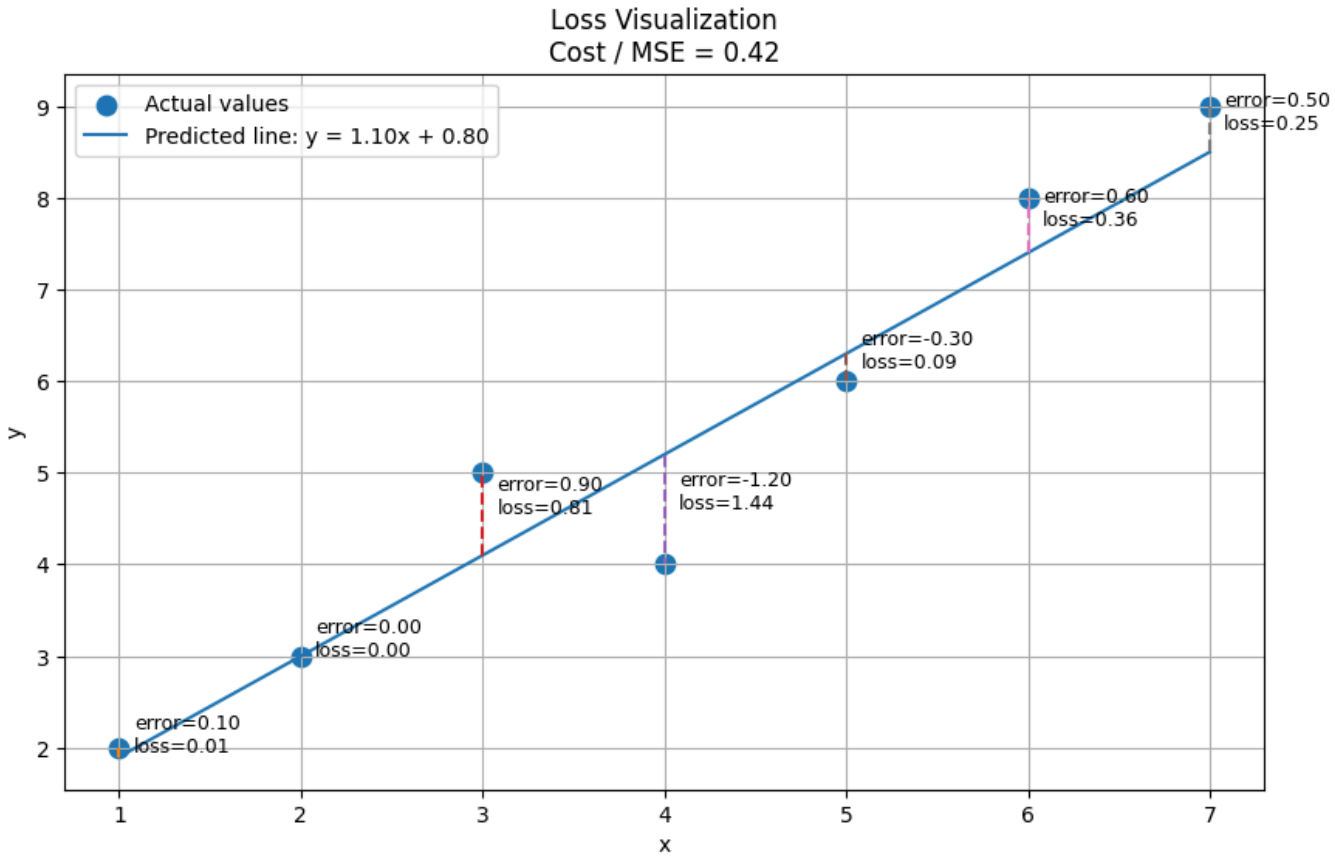

```
import numpy as np

import matplotlib.pyplot as plt



cov_xy = np.cov(x, y)[0, 1]



slope, intercept = np.polyfit(x, y, 1)

y_line = slope * x + intercept



plt.scatter(x, y, label="Data points")

plt.plot(x, y_line, label="Trend line")



plt.xlabel("x values")

plt.ylabel("y values")

plt.legend()

plt.grid(True)
```

```
plt.show()
```

Cost

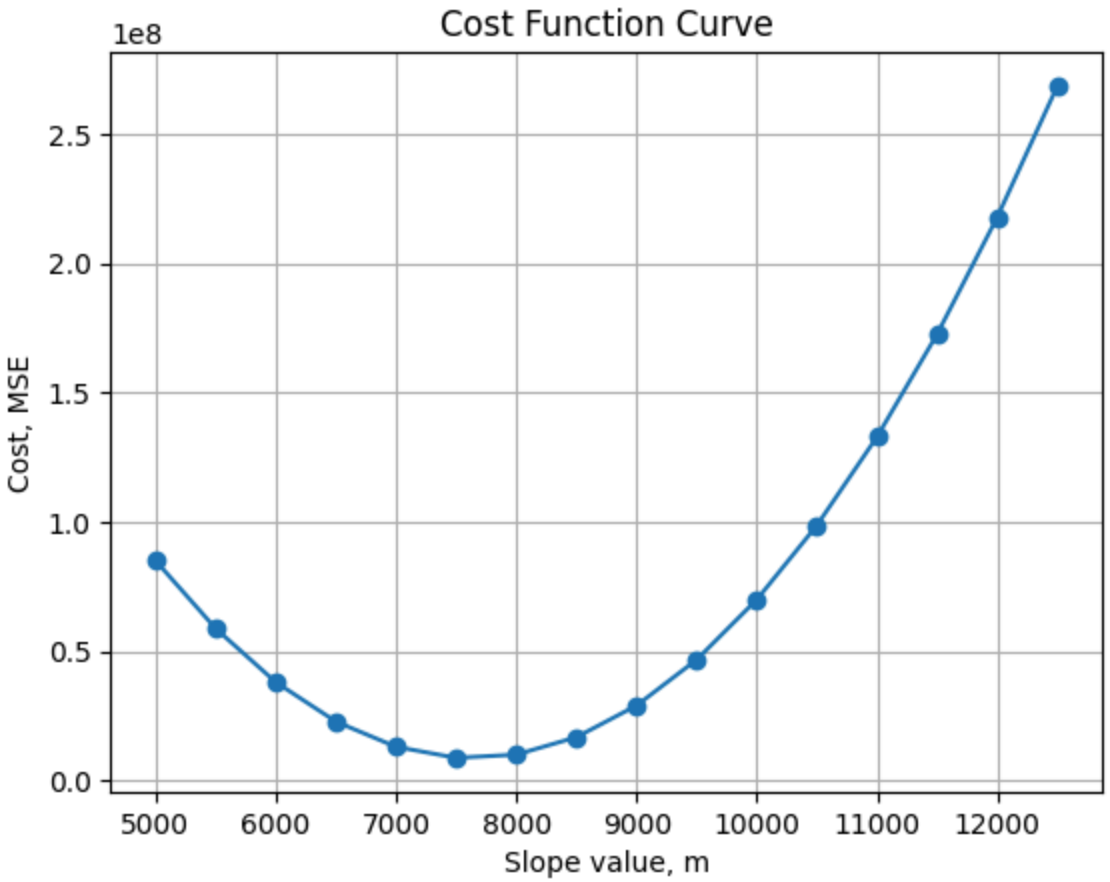
```
import numpy as np

import matplotlib.pyplot as plt




# A simple prediction line: y_pred = mx + b

m = 1.1

b = 0.8

y_pred = m * x + b



# Error and squared loss

errors = y - y_pred

squared_losses = errors ** 2



# Cost function: Mean Squared Error

cost = np.mean(squared_losses)



plt.figure(figsize=(10, 6))



# Actual data points
```

```
plt.scatter(x, y, s=80, label="Actual values")



# Predicted line

plt.plot(x, y_pred, label=f"Predicted line: y = {m:.2f}x + {b:.2f}")



# Draw vertical loss/error lines from predicted value to actual value

for xi, actual, pred, error, loss in zip(x, y, y_pred, errors,
squared_losses):

    plt.plot([xi, xi], [pred, actual], linestyle="--")



    mid_y = (actual + pred) / 2

    plt.text(

        xi + 0.08,

        mid_y,

        f"error={error:.2f}\nloss={loss:.2f}",

        fontsize=9

    )



plt.title(f"Loss Visualization\nCost / MSE = {cost:.2f}")

plt.xlabel("x")

plt.ylabel("y")

plt.grid(True)

plt.legend()

plt.show()
```




```
import numpy as np

import matplotlib.pyplot as plt


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

actual_salary = np.array([30000, 35000, 50000, 55000, 65000])


b = 25000

m_values = np.arange(5000, 13000, 500)


costs = []


for m in m_values:
```

```
    predicted_salary = m * experience + b
    mse = np.mean((actual_salary - predicted_salary) ** 2)
    costs.append(mse)


print(costs)


plt.plot(m_values, costs, marker='o')


plt.xlabel('Slope value, m')
plt.ylabel('Cost, MSE')
plt.title('Cost Function Curve')
plt.grid(True)


plt.show()


best_index = np.argmin(costs)


print("Best m:", m_values[best_index])
print("Lowest cost:", costs[best_index])
```




```
import numpy as np

import matplotlib.pyplot as plt



m = 8000

b = 25000


predicted_salary = m * experience + b


mse = np.mean((actual_salary - predicted_salary) ** 2)


plt.scatter(experience, actual_salary, label='Actual Salary')
```

```
plt.plot(experience, predicted_salary, marker='*',c='r', label='Predicted
Salary')



plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.title('Linear Regression Prediction')

plt.legend()

plt.grid(True)



plt.show()



print("Predicted Salary:", predicted_salary)

print("Cost using MSE:", mse)
```