```
import pandas as pd
```

```
df=pd.read_csv('/Salary_dataset.csv')
```

```
df.head()
```

	Unnamed: 0	YearsExperience	Salary
0	0	1.2	39344
1	1	1.4	46206
2	2	1.6	37732
3	3	2.1	43526
4	4	2.3	39892

Next steps: Generate code with df | New interactive sheet

```
df.tail()
```

	Unnamed: 0	YearsExperience	Salary
25	25	9.1	105583
26	26	9.6	116970
27	27	9.7	112636
28	28	10.4	122392
29	29	10.6	121873

```
x=df['YearsExperience']
x
```

	YearsExperience
0	1.2
1	1.4
2	1.6
3	2.1
4	2.3
5	3.0
6	3.1
7	3.3
8	3.3
9	3.8
10	4.0
11	4.1
12	4.1
13	4.2
14	4.6
15	5.0
16	5.2
17	5.4
18	6.0
19	6.1
20	6.9
21	7.2
22	8.0
23	8.3
24	8.8
25	9.1
26	9.6
27	9.7
28	10.4
29	10.6

dtype: float64

```
y=df['Salary']
y
```

	Salary
0	39344
1	46206
2	37732
3	43526
4	39892
5	56643
6	60151
7	54446
8	64446
9	57190
10	63219
11	55795
12	56958
13	57082
14	61112
15	67939
16	66030
17	83089
18	81364
19	93941
20	91739
21	98274
22	101303
23	113813
24	109432
25	105583
26	116970
27	112636
28	122392
29	121873

dtype: int64

```
X=df.iloc[:,-1].values
print(x)
```

```
[ 39344  46206  37732  43526  39892  56643  60151  54446  64446  57190
  63219  55795  56958  57082  61112  67939  66030  83089  81364  93941
  91739  98274 101303 113813 109432 105583 116970 112636 122392 121873]
```

```
y=df.iloc[:,-1].values
print(y)
```

```
[ 39344  46206  37732  43526  39892  56643  60151  54446  64446  57190
  63219  55795  56958  57082  61112  67939  66030  83089  81364  93941
  91739  98274 101303 113813 109432 105583 116970 112636 122392 121873]
```

```
import numpy as np
```

```
X=np.array(x)
Y=np.array(y)
```

```
X=x.reshape(-1,1)
print(x)
```

```
[[ 39344]
 [ 46206]
 [ 37732]
```

```
 [ 43526]
 [ 39892]
 [ 56643]
 [ 60151]
 [ 54446]
 [ 64446]
 [ 57190]
 [ 63219]
 [ 55795]
 [ 56958]
 [ 57082]
 [ 61112]
 [ 67939]
 [ 66030]
 [ 83089]
 [ 81364]
 [ 93941]
 [ 91739]
 [ 98274]
 [101303]
 [113813]
 [109432]
 [105583]
 [116970]
 [112636]
 [122392]
 [121873]]
```

```
X_min = X.min()
X_max = X.max()
X_norm = (X - X_min) / (X_max - X_min)
X_norm
```

```
array([[0.01904087],
       [0.1000945 ],
       [0.        ],
       [0.06843846],
       [0.02551382],
       [0.22337586],
       [0.26481219],
       [0.19742499],
       [0.31554453],
       [0.229837  ],
       [0.30105126],
       [0.21335932],
       [0.22709662],
       [0.2285613 ],
       [0.27616348],
       [0.35680369],
       [0.33425467],
       [0.53575478],
       [0.51537916],
       [0.66393811],
       [0.63792818],
       [0.7151193 ],
       [0.75089771],
       [0.89866525],
       [0.84691708],
       [0.80145287],
       [0.93595559],
       [0.88476258],
       [1.        ],
       [0.9938696 ]])
```

```
m = np.random.randn()
c = np.random.randn()

print("Randomly initiated slope:",m)
print("Randomly initiated intercept:",c)
```

```
Randomly initiated slope: 0.743029172804261
Randomly initiated intercept: -0.25748264521537717
```

```
def predict(X,m,c):
  y_pred = m*X + c
  return y_pred
```

```
y_pred = predict(X_norm,m,c)
```

```
#Display first five Predictions
print("First 5 predicted values:")
print(y_pred[:5])
```

```
First 5 predicted values:
[[-0.24333472]
 [-0.18310951]
 [-0.25748265]
 [-0.20663087]
 [-0.23852513]]
```

```
#Mean Square Error (cost function)
def compute_cost(y, y_pred):
  n = len(y)
  cost = (1/(2*n)) * np.sum((y_pred - y) ** 2)
  return cost
```

```
# Compute initial cost
cost = compute_cost(Y, y_pred)
print("Initial cost (MSE):", cost)
```

```
Initial cost (MSE): 97546609166.0
```

```
# Given Parameters
m = 0.0
c = 0.0

#Prediction
y_pred = predict(X_norm, m, c)

# Compute initial cost
cost = compute_cost(Y, y_pred)
print("cost for m =", m, "and c =", c, "is", cost)
```

```
cost for m = 0.0 and c = 0.0 is 97546609166.0
```

```
# Define Learning Rate
learning_rate = 0.01
epochs = 1000

print("Learning Rate:", learning_rate)
print("Number of Epochs:", epochs)
```

```
Learning Rate: 0.01
Number of Epochs: 1000
```

```
n = len(Y)
```

```
cost_history = []
```

```
for epoch in range(epochs):
  y_pred = predict(X_norm, m, c)

  dm = (1 / n) * np.sum((y_pred - Y) * X_norm)
  dc = (1 / n) * np.sum(y_pred - Y)

  m = m - learning_rate * dm
  c = c - learning_rate * dc

  cost = compute_cost(Y, y_pred)
  cost_history.append(cost)

  if epoch % 100 == 0:
    print(f"Epoch {epoch}: Cost = {cost}")
```

```
Epoch 0: Cost = 10897488926.0
Epoch 100: Cost = 10897488926.0
Epoch 200: Cost = 10897488926.0
Epoch 300: Cost = 10897488926.0
Epoch 400: Cost = 10897488926.0
Epoch 500: Cost = 10897488926.0
Epoch 600: Cost = 10897488926.0
Epoch 700: Cost = 10897488926.0
Epoch 800: Cost = 10897488926.0
Epoch 900: Cost = 10897488926.0
```

```
final_slope = m
final_intercept = c

print("Final Model Parameters:")
print("Slope (m):", final_slope)
print("Intercept (c):", final_intercept)
```

```
Final Model Parameters:
Slope (m): 3.0767872484987113e-07
Intercept (c): 76003.99999984834
```

```
def predict_salary(years_expereince, m, c, X_min, X_max):
  # Convert input into NumPy array
  X = np.array(years_expereince).reshape(-1, 1)

  #Normalize input using training parameters
  X_norm = (X - X_min) / (X_max - X_min)

  #Predict salary
  salary_pred = m * X_norm + c
  return salary_pred
```

```
# Example: Predict salary for 5 years of Expereince
years = 5

predicted_salary = predict_salary(
    years,
    final_slope,
    final_intercept,
    X_max,
    X_norm
)

print(f"predicted salary for {years} years of expereince:")
print(predicted_salary[0][0])
```

```
predicted salary for 5 years of expereince:
76004.00000015601
```

```
y_pred_final = predict(X_norm, final_slope, final_intercept)

mse = np.mean((y - y_pred_final) ** 2)

print("Final Mean Squared Error (MSE):", mse)
```