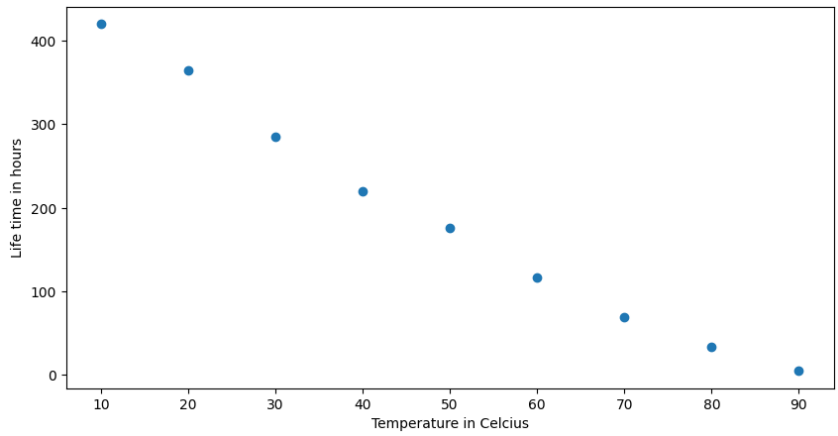
Question 1

Linear Regression Life time model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tabulate import tabulate
# import seaborn as sns


data1 = {
    'Temperature':[10,20,30,40,50,60,70,80,90],
    'Life time': [420, 365, 285, 220, 176, 117, 69, 34, 5]
}

df1 = pd.DataFrame(data1)
df1
```

	Temperature	Life time
0	10	420
1	20	365
2	30	285
3	40	220
4	50	176
5	60	117
6	70	69
7	80	34
8	90	5

```
plt.scatter(df1['Temperature'], df1['Life time'], marker='o')
plt.xlabel('Temperature in Celcius')
plt.ylabel('Life time in hours')
plt.show()
```




```
X = df1[['Temperature']]
y = df1['Life time']


# from sklearn.linear_model import LinearRegression
# from sklearn.model_selection import train_test_split

# X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=42)

# lr = LinearRegression()
# lr.fit(X_train, y_train)
```

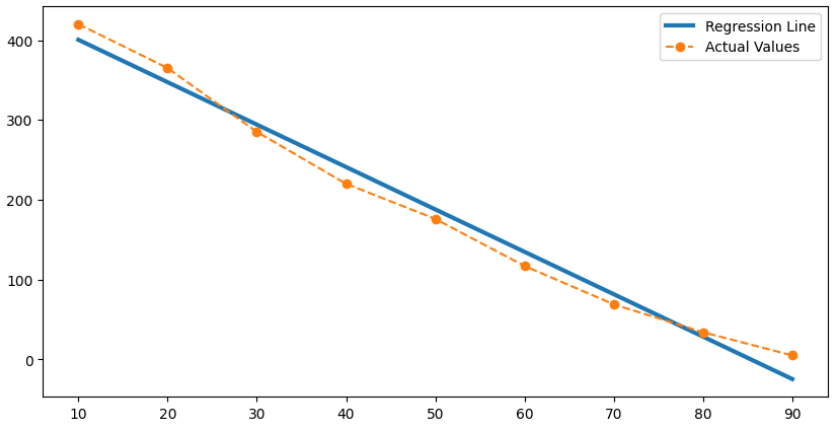
```
# print("Slope:",lr.coef_)
# print("Constant",lr.intercept_)


x = np.array(X, dtype=float).flatten()
y = np.array(y, dtype=float).flatten()


import scipy.stats as sp
slope, intercept, r_value, p_value, std_err = sp.linregress(x, y)

n=len(x)
t_crit = 2.306

margin = t_crit*std_err
lower_bound = slope-margin
upper_bound = slope+margin

xf = np.linspace(min(x), max(x), 100)
yf = slope * xf + intercept

print("slope =", slope)
print("intercept =", intercept)
print("r =", r_value**2)
print("p =", p_value)
print("s =", std_err)


print("\n95% Confidence Interval for the slope:")
print(f"Lower bound: {lower_bound:.4f}")
print(f"Upper bound: {upper_bound:.4f}")
```

```
slope = -5.3133333333333335
intercept = 453.55555555555554
r = 0.9840369938137091
p = 1.5050387692160386e-07
s = 0.2557818184006401

95% Confidence Interval for the slope:
Lower bound: -5.9032
Upper bound: -4.7235
```

```
prediction=x*slope+intercept


final1 = [["X", "Expected(Y)", "Predicted"]]

for i in range(len(prediction)):
  final1.append([x[i], y[i], prediction[i]])

t1 = tabulate(final1,headers='firstrow', tablefmt='grid')
print(t1)
```

```
+-----+---------------+-------------+
|   X |   Expected(Y) |   Predicted |
+=====+===============+=============+
|  10 |           420 |    400.422  |
+-----+---------------+-------------+
|  20 |           365 |    347.289  |
+-----+---------------+-------------+
|  30 |           285 |    294.156  |
+-----+---------------+-------------+
|  40 |           220 |    241.022  |
+-----+---------------+-------------+
|  50 |           176 |    187.889  |
+-----+---------------+-------------+
|  60 |           117 |    134.756  |
+-----+---------------+-------------+
|  70 |            69 |     81.6222 |
+-----+---------------+-------------+
|  80 |            34 |     28.4889 |
+-----+---------------+-------------+
|  90 |             5 |    -24.6444 |
+-----+---------------+-------------+
```

```
plt.rcParams['figure.figsize'] = (10, 5)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Regression Line', lw=3)
ax.plot(x,y,label="Actual Values",marker='o', ls='--')
plt.ylabel('')
ax.legend()
plt.show()
```



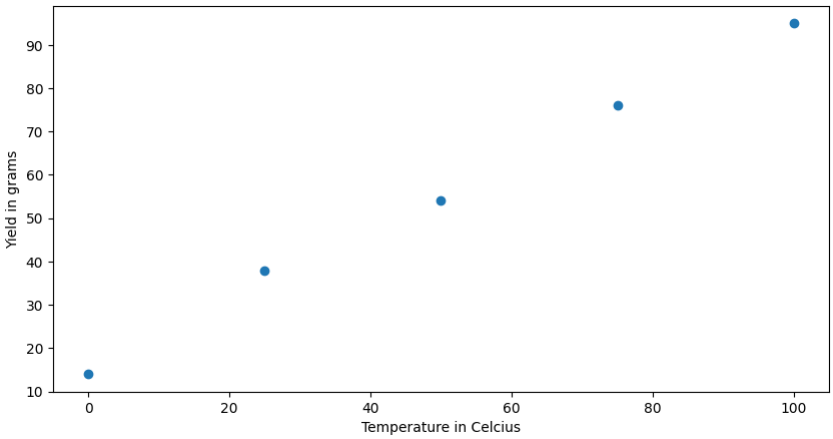
Question 2

Yield of Chemical Process

```
data2 = {
    'Temperature':[0, 25, 50, 75, 100],
    'Yield': [14, 38, 54, 76, 95]
}

df2 = pd.DataFrame(data2)
df2
```

	Temperature	Yield
0	0	14
1	25	38
2	50	54
3	75	76
4	100	95

```
plt.scatter(df2['Temperature'], df2['Yield'], marker='o')
plt.xlabel('Temperature in Celcius')
plt.ylabel('Yield in grams')
plt.show()
```




```
x = np.array(df2['Temperature'], dtype=float).flatten()
y = np.array(df2['Yield'], dtype=float).flatten()
```

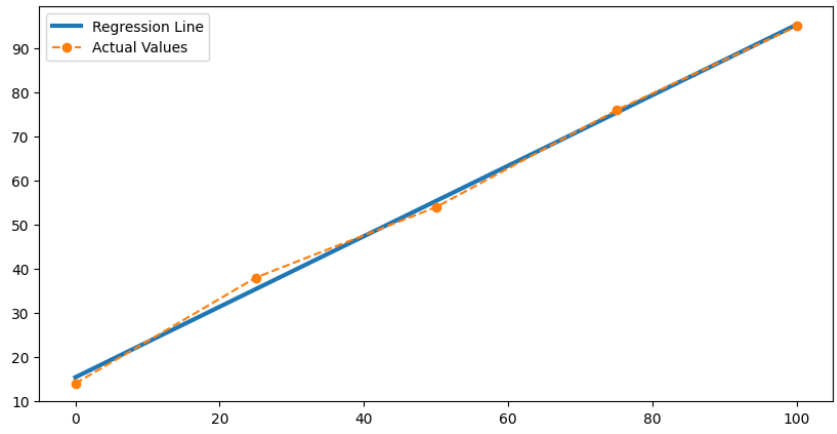
```
slope, intercept, r_value, p_value, std_err = sp.linregress(x, y)

xf = np.linspace(min(x), max(x), 100)
yf = slope * xf + intercept

print("slope =", slope)
print("intercept =", intercept)
print("r =", r_value**2)
print("p =", p_value)
print("s =", std_err)
```

```
slope = 0.8
intercept = 15.399999999999999
r = 0.9972078181092939
p = 6.267128567572262e-05
s = 0.02444040370643135
```

```
prediction=x*slope+intercept


final1 = [["X", "Expected(Y)", "Predicted"]]

for i in range(len(prediction)):
  final1.append([x[i], y[i], prediction[i]])

t1 = tabulate(final1,headers='firstrow', tablefmt='grid')
print(t1)
```

```
+-----+---------------+-------------+
|   X |   Expected(Y) |   Predicted |
+=====+===============+=============+
|   0 |            14 |        15.4 |
+-----+---------------+-------------+
|  25 |            38 |        35.4 |
+-----+---------------+-------------+
|  50 |            54 |        55.4 |
+-----+---------------+-------------+
|  75 |            76 |        75.4 |
+-----+---------------+-------------+
| 100 |            95 |        95.4 |
+-----+---------------+-------------+
```

```
plt.rcParams['figure.figsize'] = (10, 5)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Regression Line', lw=3)
ax.plot(x,y,label="Actual Values",marker='o', ls='--')
plt.ylabel('')
ax.legend()
plt.show()
```



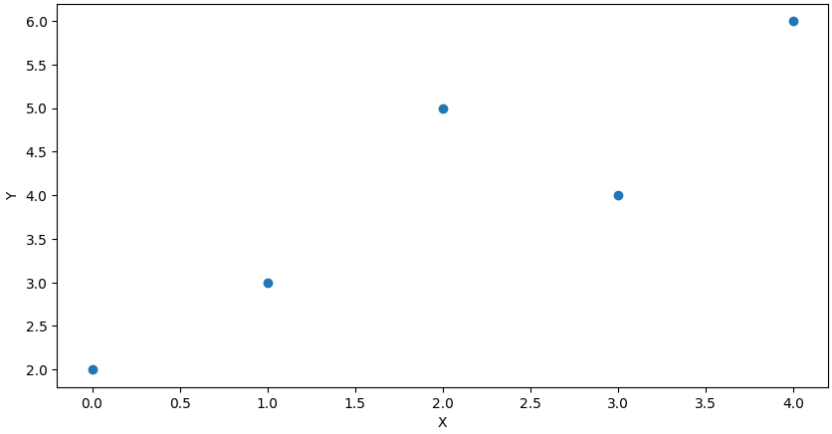
Question 3

```
data3 = {
    'X':[0, 1, 2, 3, 4],
    'Y': [2, 3, 5, 4, 6]
}
```

```
df3 = pd.DataFrame(data3)
df3
```

	X	Y
0	0	2
1	1	3
2	2	5
3	3	4
4	4	6

```
plt.scatter(df3['X'], df3['Y'], marker='o')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```




```
x = np.array(df3['X'], dtype=float).flatten()
y = np.array(df3['Y'], dtype=float).flatten()


slope, intercept, r_value, p_value, std_err = sp.linregress(x, y)

xf = np.linspace(min(x), max(x), 100)
yf = slope * xf + intercept

print("slope =", slope)
print("intercept =", intercept)
print("r =", r_value**2)
print("p =", p_value)
print("s =", std_err)
```

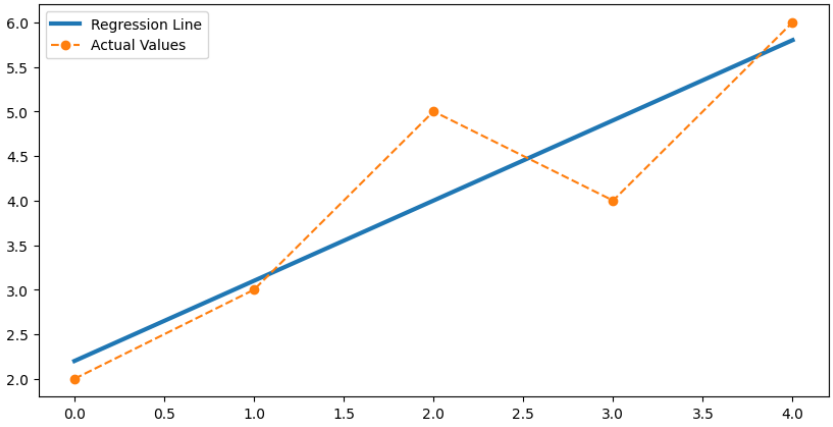
```
slope = 0.9
intercept = 2.2
r = 0.81
p = 0.03738607346849863
s = 0.25166114784235827
```

```
prediction=x*slope+intercept


final1 = [["X", "Expected(Y)", "Predicted"]]

for i in range(len(prediction)):
  final1.append([x[i], y[i], prediction[i]])

t1 = tabulate(final1,headers='firstrow', tablefmt='grid')
print(t1)
```

```
+-----+---------------+-------------+
|   X |   Expected(Y) |   Predicted |
+=====+===============+=============+
|   0 |             2 |         2.2 |
+-----+---------------+-------------+
```

```
|   1 |             3 |         3.1 |
+-----+---------------+-------------+
|   2 |             5 |         4   |
+-----+---------------+-------------+
|   3 |             4 |         4.9 |
+-----+---------------+-------------+
|   4 |             6 |         5.8 |
+-----+---------------+-------------+
```

```
plt.rcParams['figure.figsize'] = (10, 5)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Regression Line', lw=3)
ax.plot(x,y,label="Actual Values",marker='o', ls='--')
plt.ylabel('')
ax.legend()
plt.show()
```




```
print(f"Least Regression Line is:  y = {slope}*x + {intercept}")
```

```
Least Regression Line is:  y = 0.9*x + 2.2
```

```
print(f"Value of y when x = 10 is: {slope*10+intercept}")
```

```
Value of y when x = 10 is: 11.2
```

Question 4

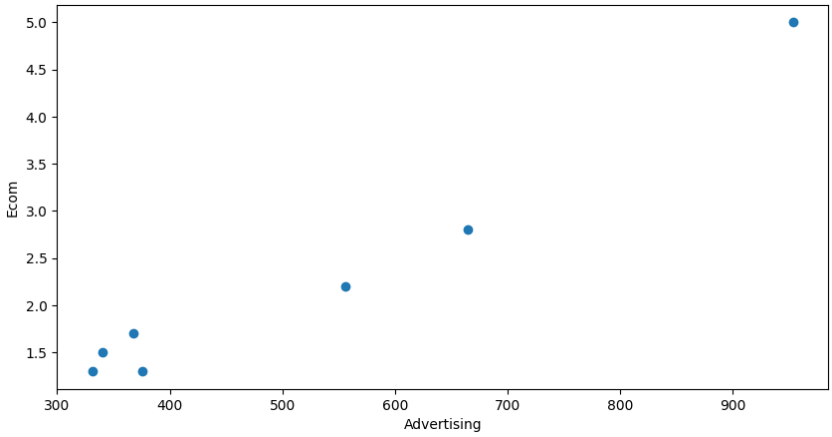
[ ] ↳ 9 *cells hidden*

Question 5

```
data5 = {
    'Advertising':[368,340,665,954,331,556,376],
    'Ecom': [1.7,1.5,2.8,5,1.3,2.2,1.3]
}

df5 = pd.DataFrame(data5)
df5
```

	Advertising	Ecom
0	368	1.7
1	340	1.5
2	665	2.8
3	954	5.0
4	331	1.3
5	556	2.2
6	376	1.3

```
plt.scatter(df5['Advertising'], df5['Ecom'], marker='o')
plt.xlabel('Advertising')
plt.ylabel('Ecom')
plt.show()
```




```
x = np.array(df5['Advertising'], dtype=float).flatten()
y = np.array(df5['Ecom'], dtype=float).flatten()


slope, intercept, r_value, p_value, std_err = sp.linregress(x, y)

xf = np.linspace(min(x), max(x), 100)
yf = slope * xf + intercept

print("slope =", slope)
print("intercept =", intercept)
print("r =", r_value**2)
print("p =", p_value)
print("s =", std_err)
```

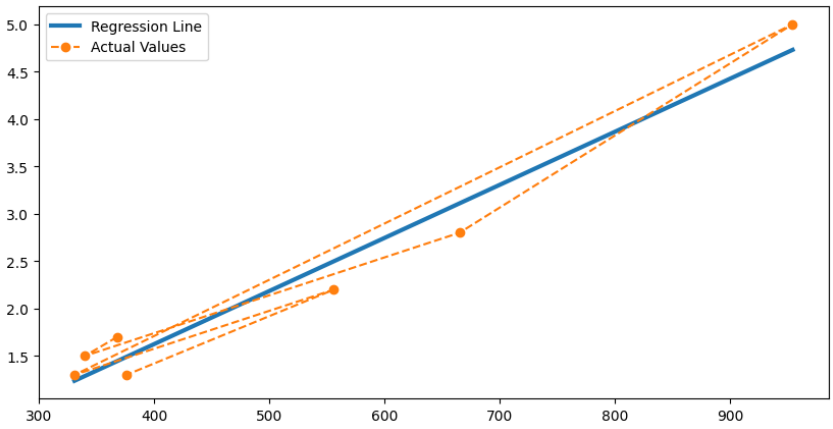
```
slope = 0.005606157184993036
intercept = -0.6180148991607144
r = 0.9612629035488399
p = 0.00010169537218360504
s = 0.0005032951082414452
```

```
prediction=x*slope+intercept


final1 = [["X", "Expected(Y)", "Predicted"]]

sorted_indices = np.argsort(x)
for i in sorted_indices:
    final1.append([x[i], y[i], prediction[i]])

t1 = tabulate(final1, headers='firstrow', tablefmt='grid')
print(t1)
```

```
+-----+---------------+-------------+
|   X |   Expected(Y) |   Predicted |
+=====+===============+=============+
| 331 |           1.3 |     1.23762 |
+-----+---------------+-------------+
| 340 |           1.5 |     1.28808 |
+-----+---------------+-------------+
| 368 |           1.7 |     1.44505 |
+-----+---------------+-------------+
| 376 |           1.3 |     1.4899  |
+-----+---------------+-------------+
| 556 |           2.2 |     2.49901 |
+-----+---------------+-------------+
| 665 |           2.8 |     3.11008 |
+-----+---------------+-------------+
| 954 |           5   |     4.73026 |
+-----+---------------+-------------+
```

```
plt.rcParams['figure.figsize'] = (10, 5)
f, ax = plt.subplots(1, 1)
```

```
ax.plot(xf, yf,label='Regression Line', lw=3)
ax.plot(df5['Advertising'],df5['Ecom'],label="Actual Values",marker='o', ls='--')
plt.ylabel('')
ax.legend()
plt.show()
```




```
print(f"Least Regression Line is:  y = {slope}*x + {intercept}")
```

```
Least Regression Line is:  y = 0.005606157184993036*x + -0.6180148991607144
```

```
print(f"Value of y when x = 8 is: {slope*8+intercept}")
```

```
Value of y when x = 8 is: -0.5731656416807701
```