In [ ]:
```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

In [ ]:
```
file = pd.read_csv('/content/sample_data/Salary_Data.csv')
```

In [ ]:
```
x = file[['YearsExperience']]
y = file['Salary']

print("X and Y : ")
print(x.shape)
print(y.shape)
```

```
X and Y :
(30, 1)
(30,)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta

print("\nX_TRAIN and Y_TRAIN : ")
print(x_train.shape)
print(y_train.shape)

print("\nX_TEST and Y_TEST : ")
print(x_test.shape)
print(y_test.shape)
```

```
X_TRAIN and Y_TRAIN :
(24, 1)
(24,)

X_TEST and Y_TEST :
(6, 1)
(6,)
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
model = LinearRegression()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print("\nModel predict y values: ")
print(y_pred)
```

```
Model predict y values:
[ 40748.96184072 122699.62295594  64961.65717022  63099.14214487
 115249.56285456 107799.50275317]
```

In [ ]:
```
mtp.scatter(x_train, y_train, color = 'red', label = 'Training Data')
mtp.plot(x_train, model.predict(x_train), color = 'blue', label = 'Regression Line'
mtp.title('Salary v/s Experience (Training Set)')
```

```
mtp.xlabel('Years of Experience')
mtp.ylabel('Salary')
mtp.legend()
mtp.show()
```

In [ ]:
```
mtp.scatter(x_test, y_test, color = 'red', label = 'Test Data')
mtp.plot(x_test, model.predict(x_test), color = 'blue', label = 'Regression Line')
mtp.title('Salary v/s Experience (Test Set)')
mtp.xlabel('Years of Experience')
mtp.ylabel('Salary')
mtp.legend()
mtp.show()
```

In [ ]:
```
# import sklearn.metrics as r2_score
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)

print("R2 Score : ")
print(r2)

c = model.intercept_
m = model.coef_

print("\nModel intercept and coef")
print(m, c)
```

```
R2 Score :
0.988169515729126

Model intercept and coef
[9312.57512673] 26780.09915062818
```

In [ ]:
```
accuracy = []

for i in range(100):
  x_train1, x_test1, y_train1, y_test1 = train_test_split(x, y, test_size = 0.8, ra
  model1 = LinearRegression()
  model1.fit(x_train1, y_train1)
  y_pred1 = model.predict(x_test1)
  r2_1 = r2_score(y_test1, y_pred1)
  accuracy.append(r2_1)
```

```
print(accuracy)
print(nm.max(accuracy))
```

```
[0.9536645755326943, 0.9587276669369498, 0.9599226033873139, 0.9474003096085318, 0.9
621063968368476, 0.9604183789115188, 0.9512638404910294, 0.9593236934450713, 0.95864
64746719688, 0.9412976253548547, 0.9596455139722875, 0.9645945039362755, 0.954645182
8306478, 0.9524283843195558, 0.9574349109881829, 0.9616217677370656, 0.9581808041466
211, 0.9583546185222949, 0.9641428142717868, 0.9616813005555789, 0.9443891385389038,
0.9578347285706635, 0.9509390773626878, 0.9627928948960339, 0.9524032048211271, 0.95
67653204443508, 0.9636873613970725, 0.9612603275880329, 0.9512660019542469, 0.962162
4564917671, 0.9526594452373836, 0.9557050174779026, 0.9638773784528749, 0.9497596777
951478, 0.9520178577454619, 0.9609223936735086, 0.9572491005122411, 0.95870004882875
36, 0.9636878446739479, 0.9422185509273002, 0.9501560722560173, 0.9510736292338142,
0.9639970519061158, 0.9662264814449496, 0.9503771843706749, 0.9473614276653719, 0.96
00741369543193, 0.9574501126663743, 0.96173672184421, 0.9600744775743649, 0.96146837
35473679, 0.950494768868645, 0.9579475923300533, 0.9570848179401535, 0.9526204223354
385, 0.9601767789630493, 0.9529857750528089, 0.938427688033362, 0.9566097445611071,
0.9695111211997747, 0.9579177248530238, 0.9529045551559473, 0.9650129321576381, 0.96
07690590975153, 0.9558401079078208, 0.953821050913366, 0.9633530805881929, 0.9525314
94217113, 0.9596164770706567, 0.9542990700558682, 0.9518009398927234, 0.962840894754
048, 0.9612347425144955, 0.9630980566538064, 0.939956073561345, 0.9663083377924078,
0.9506173252358386, 0.9522705290853881, 0.9551200629157734, 0.9534738892946083, 0.95
91085518943221, 0.9572519800510758, 0.9384132231966089, 0.9581215157168369, 0.958166
5113069906, 0.9595025129155997, 0.9549392840822342, 0.9525312590068257, 0.9539815539
857341, 0.965872795774837, 0.9494500804095631, 0.9470786930345976, 0.953711693254650
7, 0.9524522583709272, 0.9409849345516205, 0.9530882303436213, 0.9507159559904956, 0
.9563279411484026, 0.9515636676117039, 0.9534797597250289]
0.9695111211997747
```

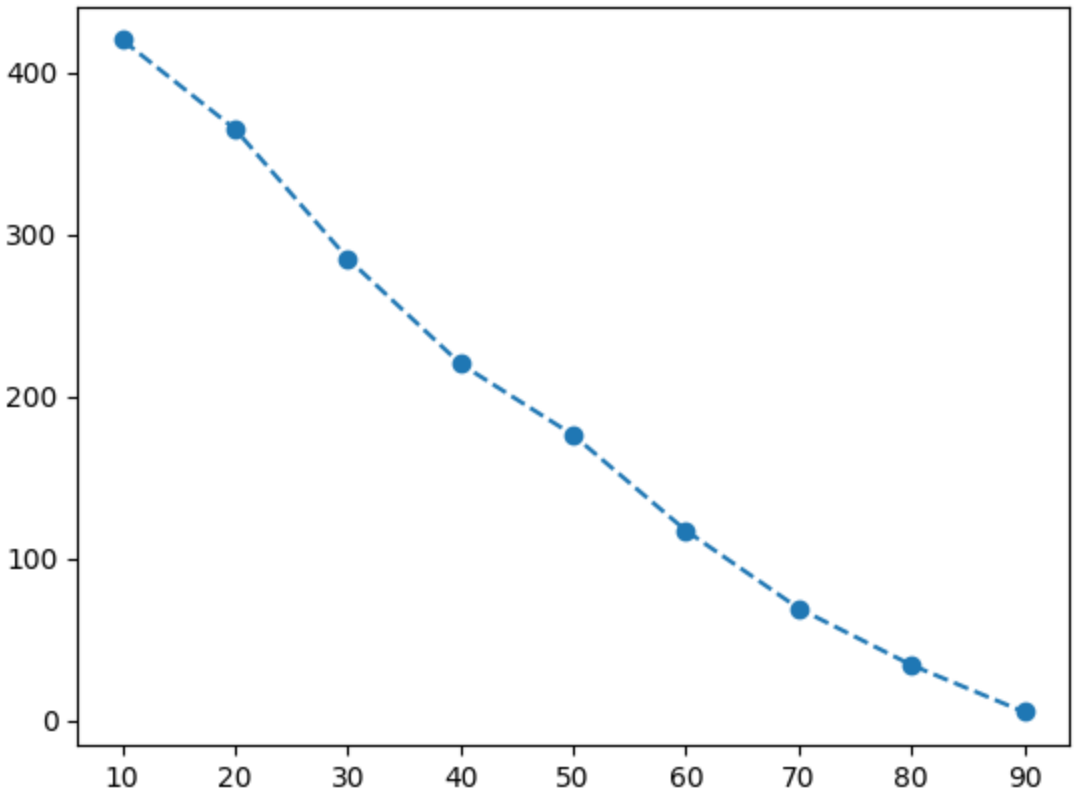
In [ ]:
```
###### POSTLAB 1 ########################################
```

In [ ]:
```
### Q.1)

import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
import scipy.stats as sp
mtp.rcParams['figure.figsize'] = (20.0, 10.0)

df = {
    'X' : [10,20,30,40,50,60,70,80,90],
    'Y' : [420,365,285,220,176,117,69,34,5]
}

data = pd.DataFrame(df)
print(data.head(9))
```

```
   X    Y
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
```

In [ ]:
```
X = data['X'].values
Y = data['Y'].values

mtp.plot(X,Y,marker='o', ls='--')
```

Out[ ]: `[<matplotlib.lines.Line2D at 0x7d407c2dfdd0>]`



In [ ]:
```
y=np.array(data['Y'], dtype=float)
x=np.array(data['X'], dtype=float)
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,"\n","intercept = ",intercept,"\n",'r = ', r_value**2, '\n',
```

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

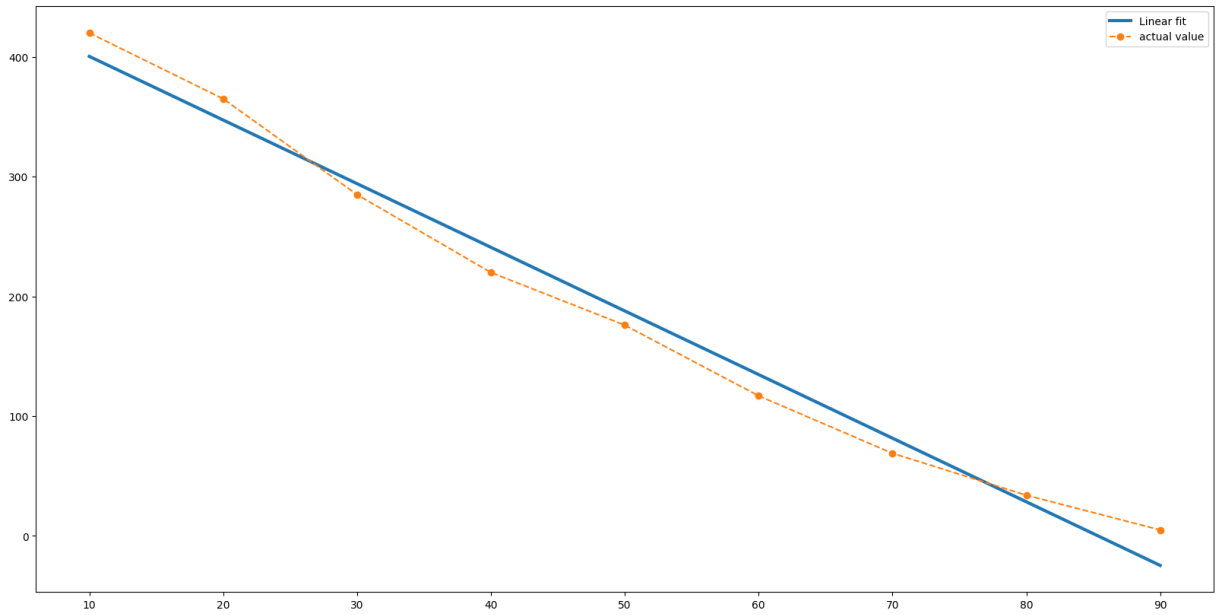
In [ ]:
```
data["expected"] = data["X"] * slope + intercept

data.head(9)
```

Out[ ]:

	X	Y	expected
0	10	420	400.422222
1	20	365	347.288889
2	30	285	294.155556
3	40	220	241.022222
4	50	176	187.888889
5	60	117	134.755556
6	70	69	81.622222
7	80	34	28.488889
8	90	5	-24.644444

In [ ]:
```
f, ax = mtp.subplots(1, 1)
ax.plot(xf, yf,label='Linear fit', lw=3)
ax.plot(X,Y,label="actual value",marker='o', ls='--')
mtp.ylabel('')
ax.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x7d407ccf8650>



In [ ]:
```
from scipy.stats import t
tinv = lambda p, df: abs(t.ppf(p/2, df))
ts = tinv(0.05, len(X)-2)
```

In [ ]:
```
print(f"slope (95%): {slope:.6f} +/- {ts*std_err:.6f}")
```

```
slope (95%): -5.313333 +/- 0.604828
```

In [ ]:
```
print(f"intercept (95%): {intercept:.6f}"f" +/- {ts*std_err:.6f}")
```

```
intercept (95%): 453.555556 +/- 0.604828
```

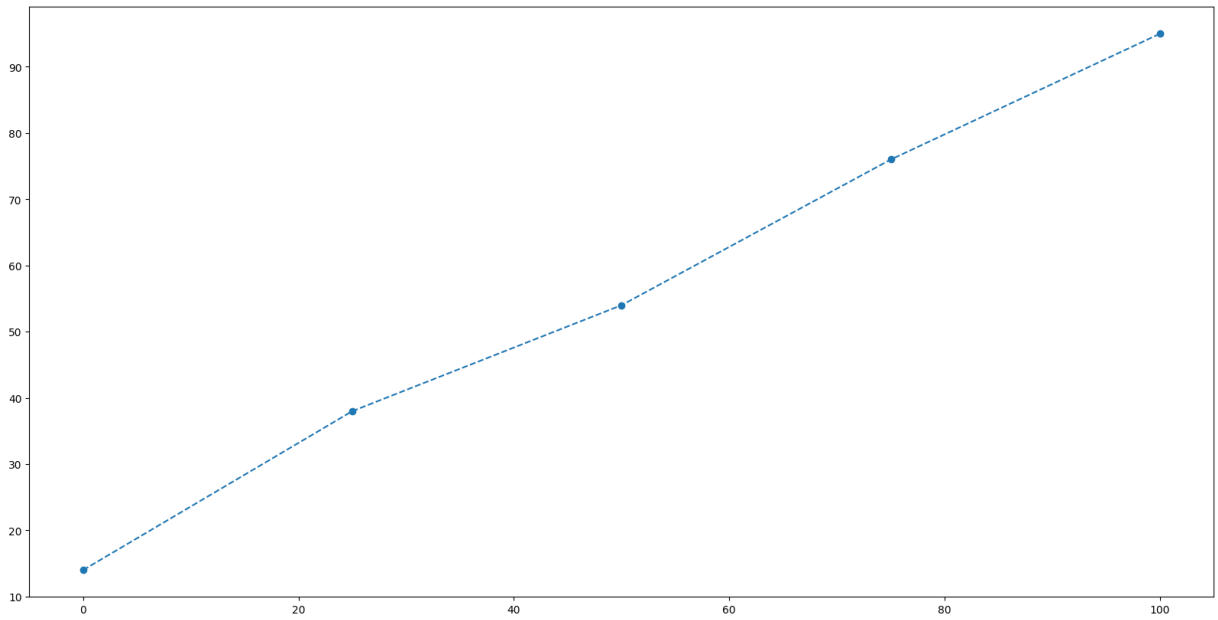
In [ ]:
```
############### Question 2 ###################
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as sp
plt.rcParams['figure.figsize'] = (20.0, 10.0)

df = {
    'X':[0,25,50,75,100],
    'Y':[14,38,54,76,95]
}

data = pd.DataFrame(df)
print(data.head(9))
```

```
     X   Y
0    0  14
1   25  38
2   50  54
3   75  76
4  100  95
```

In [ ]:
```
X = data['X'].values
Y = data['Y'].values


plt.plot(X, Y, marker = 'o', ls = '--')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7d407beefdd0>]
```



In [ ]:
```
y=np.array(data['Y'], dtype=float)
x=np.array(data['X'], dtype=float)
slope, intercept, r_value, p_value, std_err =sp.linregress(x,y)
xf = np.linspace(min(x),max(x),100)
yf = (slope*xf)+intercept
```

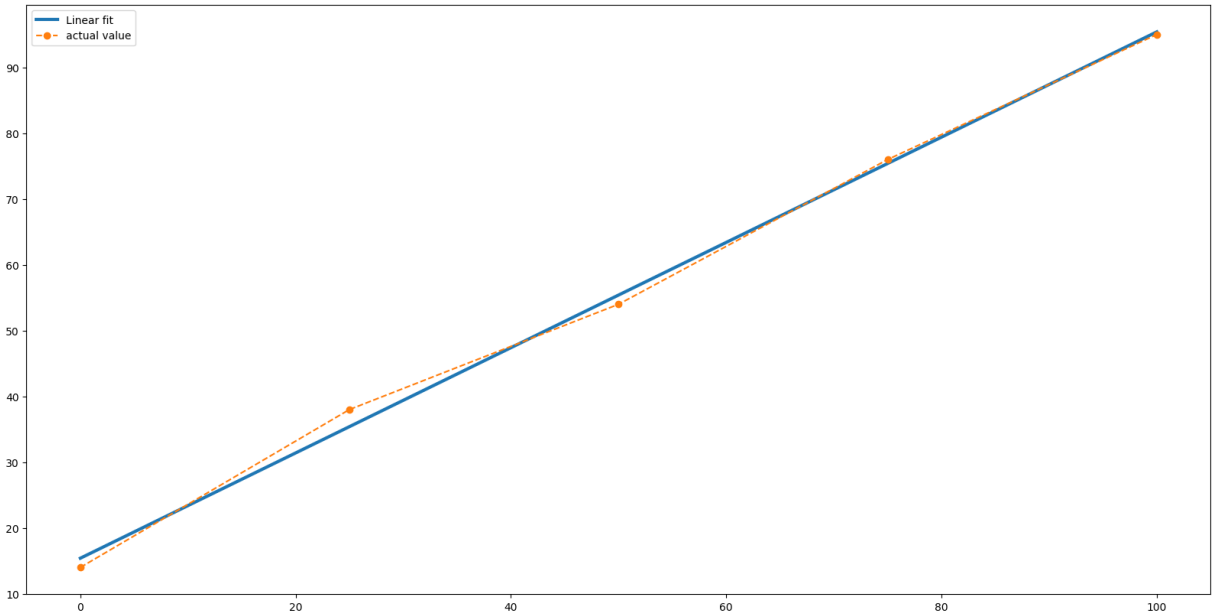
```
print("slope = ",slope,"\n","intercept = ",intercept,"\n",'r = ', r_value**2, '\n',
```

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

In [ ]:
```
data["expected"]=data["X"]*slope+intercept
print(data.head())
plt.rcParams['figure.figsize'] = (20.0, 10.0)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Linear fit', lw=3)
ax.plot(X,Y,label="actual value",marker='o', ls='--')
plt.ylabel('')
ax.legend()
```

```
     X   Y  expected
0    0  14      15.4
1   25  38      35.4
2   50  54      55.4
3   75  76      75.4
4  100  95      95.4
```

Out[ ]: <matplotlib.legend.Legend at 0x7d407e7e0650>



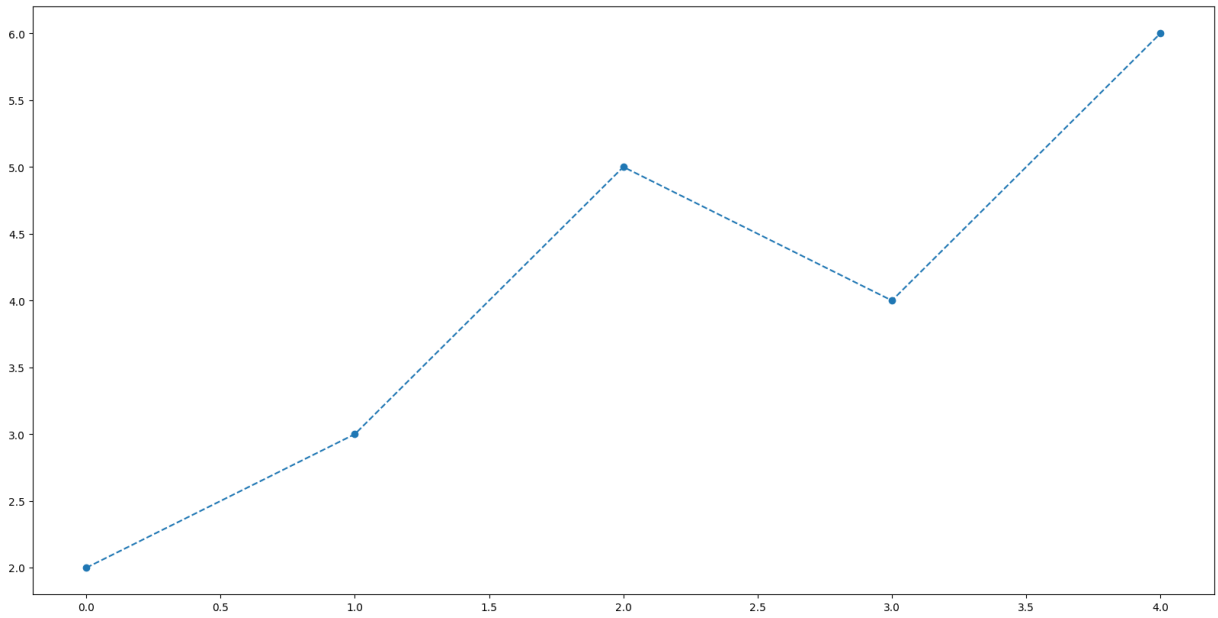
In [ ]:
```
from scipy.stats import t
tinv = lambda p, df: abs(t.ppf(p/2, df))
ts = tinv(0.05, len(X)-2)
print(f"slope (95%): {slope:.6f} +/- {ts*std_err:.6f}")
print(f"intercept (95%): {intercept:.6f}"f" +/- {ts*std_err:.6f}")
```

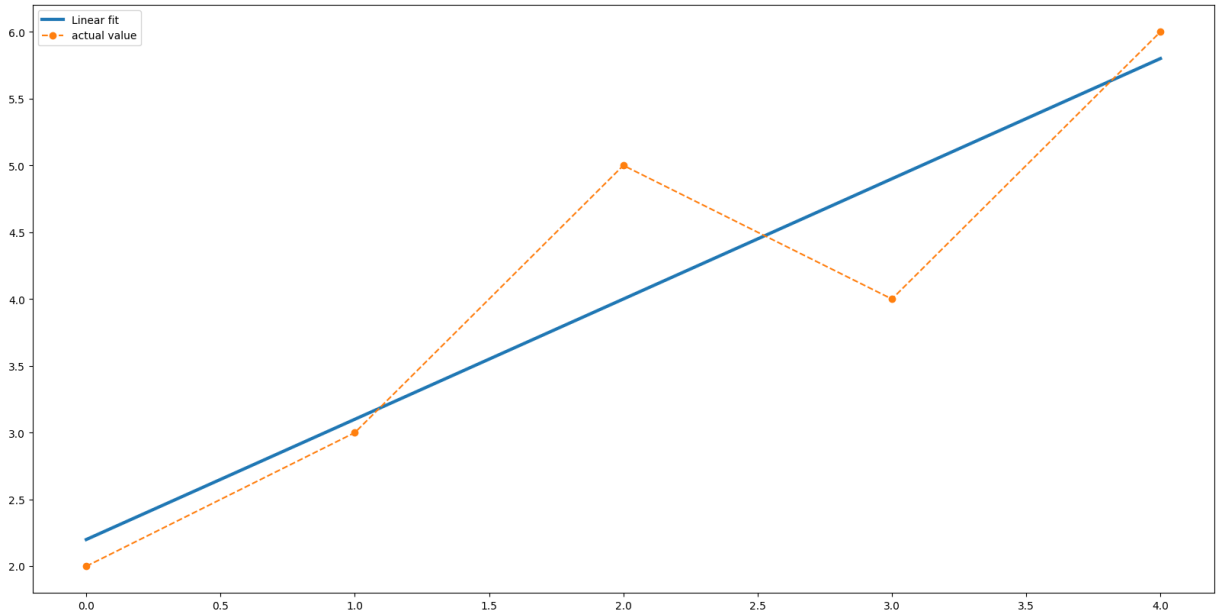
```
slope (95%): 0.800000 +/- 0.077780
intercept (95%): 15.400000 +/- 0.077780
```

In [ ]:
```
########### Question 3 ####################
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import scipy.stats as sp
plt.rcParams['figure.figsize'] = (20.0, 10.0)

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

data = pd.DataFrame(df)
print (data.head(9))

X = data['X'].values
Y = data['Y'].values
plt.plot(X,Y,marker='o', ls='--')

y=np.array(data['Y'], dtype=float)
x=np.array(data['X'], dtype=float)
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,"\n","intercept=",intercept,"\n",'r = ', r_value**2, '\n', 'p
print(f"\nequation of line: y={slope:.2f}x+{intercept:.2f}")

data["expected"]=data["X"]*slope+intercept
print(data.head())
plt.rcParams['figure.figsize'] = (20.0, 10.0)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Linear fit', lw=3)
ax.plot(X,Y,label="actual value",marker='o', ls='--')
plt.ylabel('')
ax.legend()

from scipy.stats import t
tinv = lambda p, df: abs(t.ppf(p/2, df))
pred = lambda x: slope*x+intercept
ts = tinv(0.05, len(X)-2)
print(f"slope (95%): {slope:.6f} +/- {ts*std_err:.6f}")
print(f"intercept (95%): {intercept:.6f}"f" +/- {ts*std_err:.6f}")
print("value of y at x=10=",pred(10))
```

```
   X  Y
0  0  2
1  1  3
2  2  5
3  3  4
4  4  6
slope= 0.9
 intercept= 2.2
 r =  0.81
 p =  0.03738607346849863
 s =  0.25166114784235827

equation of line: y=0.90x+2.20
   X  Y  expected
0  0  2       2.2
1  1  3       3.1
2  2  5       4.0
3  3  4       4.9
4  4  6       5.8
slope (95%): 0.900000 +/- 0.800898
intercept (95%): 2.200000 +/- 0.800898
value of y at x=10= 11.2
```

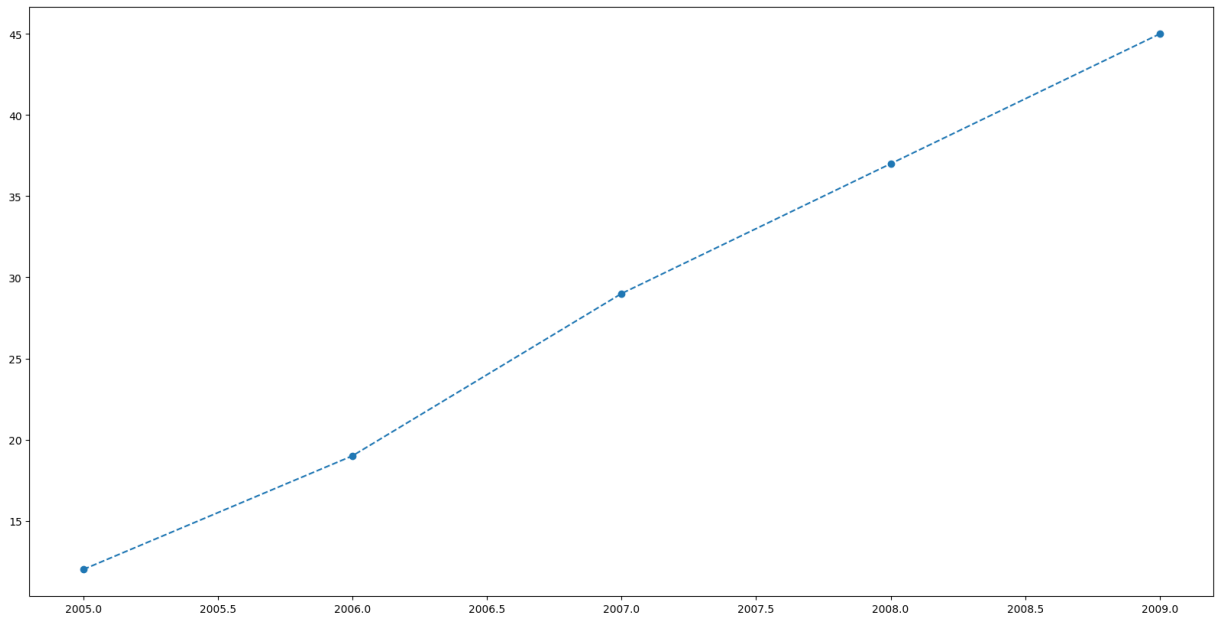


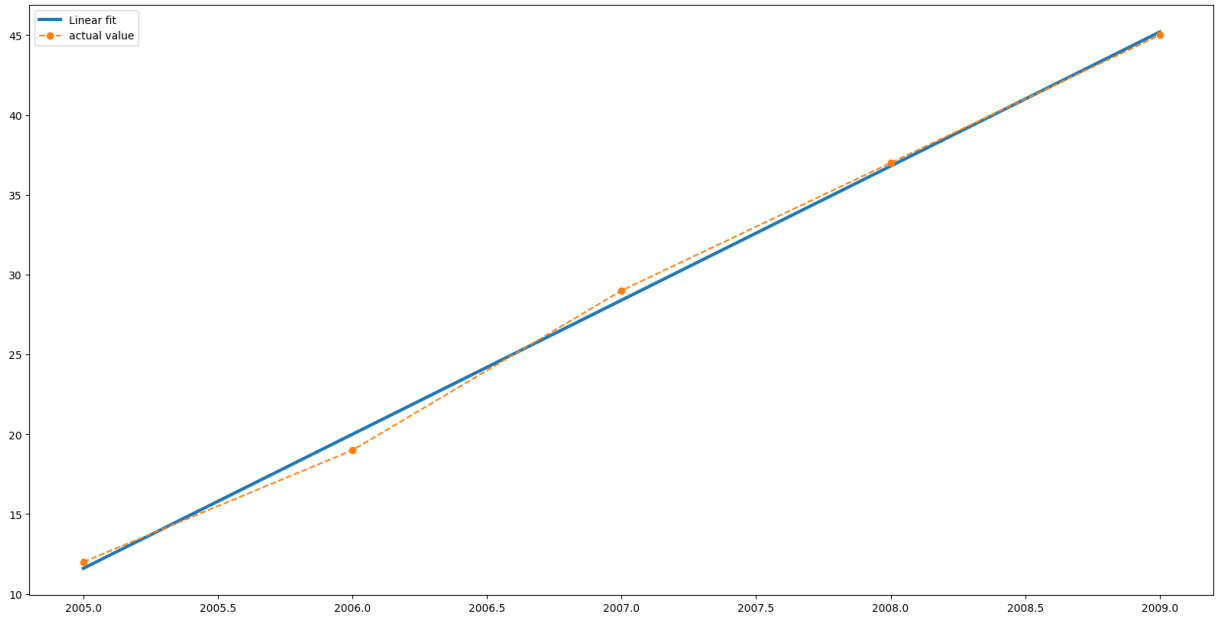


In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as sp
plt.rcParams['figure.figsize'] = (20.0, 10.0)


df = {'X':[2005,2006,2007,2008,2009],
'Y':[12,19,29,37,45]}


data = pd.DataFrame(df)
print (data.head(9))


X = data['X'].values
Y = data['Y'].values
plt.plot(X,Y,marker='o', ls='--')


y=np.array(data['Y'], dtype=float)
x=np.array(data['X'], dtype=float)
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,"\n","intercept=",intercept,"\n",'r = ', r_value**2, '\n', 'p
print(f"\nequation of line: y={slope:.2f}x+{intercept:.2f}")

data["expected"]=data["X"]*slope+intercept
print(data.head())
plt.rcParams['figure.figsize'] = (20.0, 10.0)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Linear fit', lw=3)
ax.plot(X,Y,label="actual value",marker='o', ls='--')
plt.ylabel('')
```

```
ax.legend()


from scipy.stats import t
tinv = lambda p, df: abs(t.ppf(p/2, df))
pred = lambda x: slope*x+intercept
ts = tinv(0.05, len(X)-2)
print(f"slope (95%): {slope:.6f} +/- {ts*std_err:.6f}")
print(f"intercept (95%): {intercept:.6f}"f" +/- {ts*std_err:.6f}")
print("value of y at x=2012=",pred(2012))
```

```
      X   Y
0  2005  12
1  2006  19
2  2007  29
3  2008  37
4  2009  45
slope= 8.4
 intercept= -16830.399999999998
 r =  0.9977375565610856
 p =  4.570360814080035e-05
 s =  0.23094010767586692

equation of line: y=8.40x+-16830.40
      X   Y  expected
0  2005  12      11.6
1  2006  19      20.0
2  2007  29      28.4
3  2008  37      36.8
4  2009  45      45.2
slope (95%): 8.400000 +/- 0.734954
intercept (95%): -16830.400000 +/- 0.734954
value of y at x=2012= 70.40000000000146
```

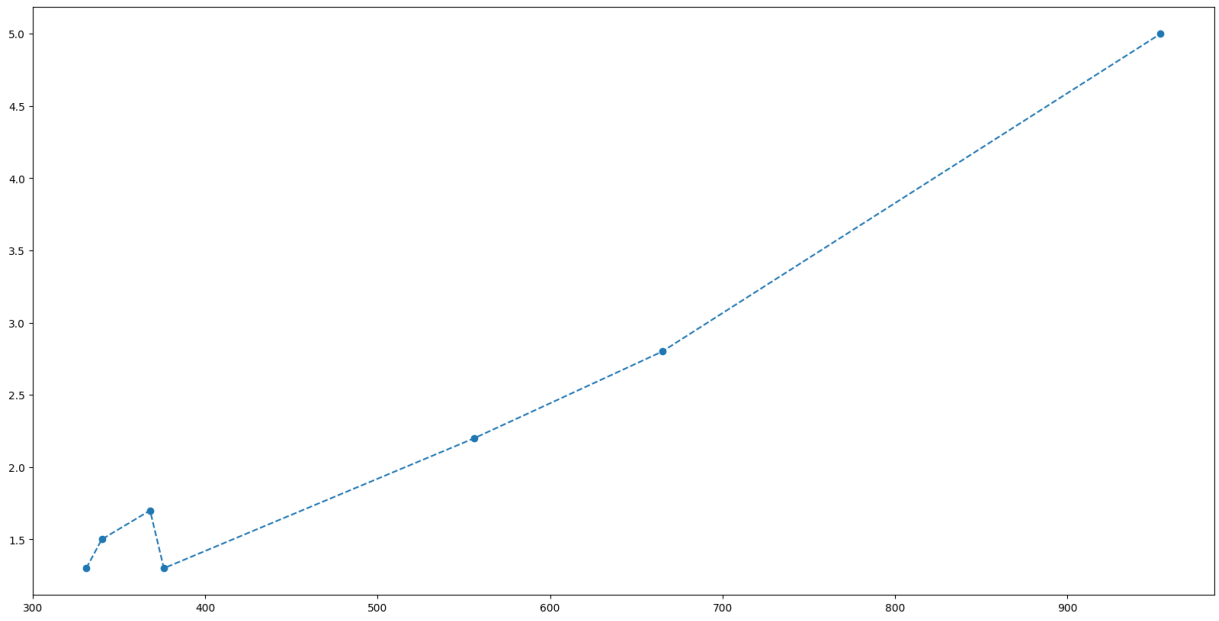


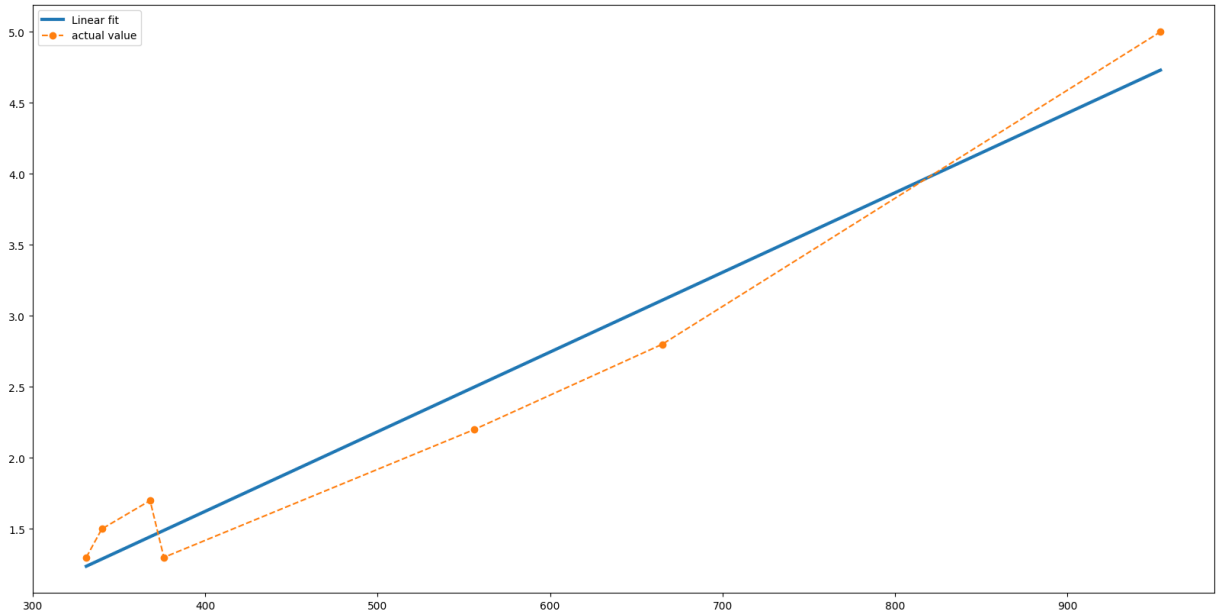


In [ ]:
```
############## Question 5 #################
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as sp
plt.rcParams['figure.figsize'] = (20.0, 10.0)


df = {'X':[368,340,665,954,331,556,376],
'Y':[1.7,1.5,2.8,5,1.3,2.2,1.3]}


data = pd.DataFrame(df)
data=data.sort_values('X')
print (data.head(9))


X = data['X'].values
Y = data['Y'].values
plt.plot(X,Y,marker='o', ls='--')


y=np.array(data['Y'], dtype=float)
x=np.array(data['X'], dtype=float)
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,"\n","intercept=",intercept,"\n",'r = ', r_value**2, '\n', 'p
print(f"\nequation of line: y={slope:.2f}x+{intercept:.2f}")

data["expected"]=data["X"]*slope+intercept
print(data.head())
plt.rcParams['figure.figsize'] = (20.0, 10.0)
f, ax = plt.subplots(1, 1)
ax.plot(xf, yf,label='Linear fit', lw=3)
```

```
ax.plot(X,Y,label="actual value",marker='o', ls='--')
plt.ylabel('')
ax.legend()


from scipy.stats import t
tinv = lambda p, df: abs(t.ppf(p/2, df))
pred = lambda x: slope*x+intercept
ts = tinv(0.05, len(X)-2)
print(f"slope (95%): {slope:.6f} +/- {ts*std_err:.6f}")
print(f"intercept (95%): {intercept:.6f}"f" +/- {ts*std_err:.6f}")
print("value of y at x=8=",pred(8))
```

```
     X    Y
4  331  1.3
1  340  1.5
0  368  1.7
6  376  1.3
5  556  2.2
2  665  2.8
3  954  5.0
slope= 0.005606157184993036
 intercept= -0.6180148991607144
 r =  0.9612629035488399
 p =  0.00010169537218360504
 s =  0.0005032951082414453

equation of line: y=0.01x+-0.62
     X    Y  expected
4  331  1.3  1.237623
1  340  1.5  1.288079
0  368  1.7  1.445051
6  376  1.3  1.489900
5  556  2.2  2.499008
slope (95%): 0.005606 +/- 0.001294
intercept (95%): -0.618015 +/- 0.001294
value of y at x=8= -0.5731656416807701
```




Linear fit
actual value
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
300
400
500
600
700
800
900