```
#Importing the required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
x_train = np.array([1,2])

y_train = np.array([3,5])

print(f"x_train = {x_train}")

print(f"y_train = {y_train}")
# m is the number of training examples

print(f"x_train.shape: {x_train.shape}")

m = x_train.shape[0]

print(f"Number of training examples is: {m}")


for i in range(2):

x_i = x_train[i]

y_i = y_train[i]

print(f"(x^({i}), y^({i})) = ({x_i}, {y_i})")

# Plot the data points

plt.scatter(x_train, y_train, marker='x', c='r')

# Set the title

plt.title("Crush in each semester")

# Set the y-axis label

plt.ylabel('Number of Cursh')
```

```
# Set the x-axis label

plt.xlabel('Semester')

plt.show()
```

```
def compute_model_output(x, w, b):
    """
    Computes the prediction of a linear model    Args:      x (ndarray
(m,)): Data, m examples      w,b (scalar)    : model parameters
    Returns
      y (ndarray (m,)): target values    """    m = x.shape[0]
    f_wb = np.zeros(m)
    for i in range(m):
        f_wb[i] = w * x[i] + b

    return f_wb

def plot_graph(x_train,y_train,w,b):
    tmp_f_wb = compute_model_output(x_train, w, b)

    # Plot our model prediction
    plt.plot(x_train, tmp_f_wb, c='b',label='Our Prediction')

    # Plot the data points
    plt.scatter(x_train, y_train, marker='x', c='r',label='Actual Values')

    # Set the title
    plt.title("Crush in each semester")
    # Set the y-axis label
    plt.ylabel('Number of Cursh')
    # Set the x-axis label
    plt.xlabel('Semester')

    plt.legend()
    plt.show()


def weight_bias(x, y):
    cov_xy = np.mean((x - np.mean(x)) * (y - np.mean(y)))
    cov_xx = np.mean((x - np.mean(x)) * (x - np.mean(x)))

    m = cov_xy / cov_xx
    b = np.mean(y) - m * np.mean(x)
```

```
    return m, b
```

```
w = 2

b=3

plot_graph(x_train,y_train,w,b)
```

```
w,b= weight_bias(x_train,y_train)

plot_graph(x_train,y_train,w,b)
x_train = np.array([1.0,2.0,3.0,4,5,6])

y_train = np.array([3,5,6,7,8,9])

w,b= weight_bias(x_train,y_train)

plot_graph(x_train,y_train,w,b)
```

At 7th semester my number of crushes will be

`w*7+b`

```
from sklearn.linear_model import LinearRegression
# 2. Create and train the model
model = LinearRegression().fit(x_train.reshape(-1,1), y_train)
```

```
# 3. Get results
print(f"Intercept: {model.intercept_}")
print(f"Slope: {model.coef_}")
```

data_path = "https://www.statlearning.com/s/Advertising.csv"

```
import pandas as pd

data_path = "https://www.statlearning.com/s/Advertising.csv"



# Read the CSV data from the link
```

```
data_df = pd.read_csv(data_path,index_col=0)



# Print out first 5 samples from the DataFrame

data_df.head()
```

```
import matplotlib.pyplot as plt

import matplotlib as mpl

from sklearn import linear_model

fig = plt.figure(figsize=(15,4))

gs = mpl.gridspec.GridSpec(1,3)



# Plot of sales vs TV

ax = fig.add_subplot(gs[0])

ax.scatter(data_df["TV"], data_df["sales"], color="red", marker=".")

ax.set_xlabel("TV")

ax.set_ylabel("sales")



# Plot of sales vs radio

ax = fig.add_subplot(gs[1])

ax.scatter(data_df["radio"], data_df["sales"], color="green", marker=".")

ax.set_xlabel("radio")

ax.set_ylabel("sales")



# Plot of sales vs newspaper
```

```
ax = fig.add_subplot(gs[2])

ax.scatter(data_df["newspaper"], data_df["sales"], color="blue", marker=".")

ax.set_xlabel("newspaper")

ax.set_ylabel("sales")



plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn import linear_model

fig = plt.figure(figsize=(15, 4))
gs = mpl.gridspec.GridSpec(1, 3)

def train_plot(data_df, feature, ax, col):
    X = data_df[[feature]].values
    Y = data_df[['sales']].values

    cov_XY = np.mean((X - np.mean(X)) * (Y - np.mean(Y)))
    cov_XX = np.mean((X - np.mean(X)) * (X - np.mean(X)))

    m = cov_XY / cov_XX
    b = np.mean(Y) - m * np.mean(X)


    y_hat = m * X + b

    print(feature)
    print("Manual:", m, b)

    reg = linear_model.LinearRegression()
    reg.fit(X, Y)

    print("Sklearn:", reg.coef_, reg.intercept_)
    print()

    ax.scatter(data_df[feature], data_df["sales"], color=col, marker=".")
```

```
    ax.plot(X, y_hat, color="black")
    ax.set_xlabel(feature)
    ax.set_ylabel("sales")
    ax.set_title("$y$ = %.3f + %.3f$x$" % (b, m))

ax0 = fig.add_subplot(gs[0])
train_plot(data_df, "TV", ax0, "red")

ax1 = fig.add_subplot(gs[1])
train_plot(data_df, "radio", ax1, "green")

ax2 = fig.add_subplot(gs[2])
train_plot(data_df, "newspaper", ax2, "blue")

plt.tight_layout()
plt.show()
```