```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


data = pd.read_csv("/content/ex1.csv")
data
```

	a	b
0	6.1101	17.59200
1	5.5277	9.13020
2	8.5186	13.66200
3	7.0032	11.85400
4	5.8598	6.82330
...	...	...
92	5.8707	7.20290
93	5.3054	1.98690
94	8.2934	0.14454
95	13.3940	9.05510
96	5.4369	0.61705

97 rows × 2 columns

```
#compute cost value
def computeCost(X,y,theta):
  m=len(y)
  h=X.dot(theta)
  square_err=(h - y)**2
  return 1/(2*m) * np.sum(square_err)


#computing cost value
data_n=data.values
m=data_n[:,0].size
X=np.append(np.ones((m, 1)),data_n[:,0].reshape(m, 1),axis=1)
y=data_n[:,1].reshape (m,1)
theta=np.zeros((2,1))
computeCost(X,y,theta) # Call the function
```

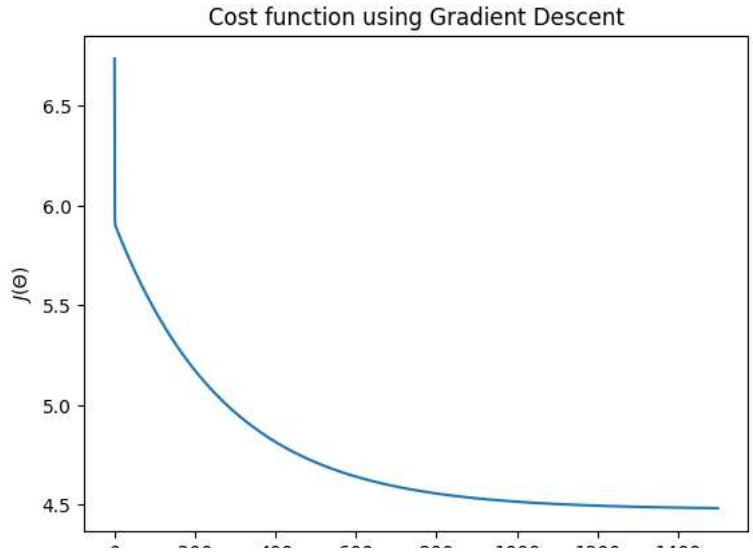
```
32.072733877455676
```

```
def gradientDescent (X,y, theta, alpha, num_iters):
  m=len (y)
  J_history=[]

  for i in range(num_iters):
    predictions = X.dot(theta)
    error = np.dot(X.transpose(), (predictions -y))
    descent=alpha * 1/m * error
    theta-=descent
    J_history.append(computeCost (X,y, theta))
  return theta, J_history


#h(x) value
theta,J_history = gradientDescent (X,y, theta, 0.01,1500)
print ("h(x) ="+str (round(theta[0,0],2))+" + "+str(round(theta[1,0],2))+"X1")
```

```
h(x) =-3.63 + 1.17X1
```

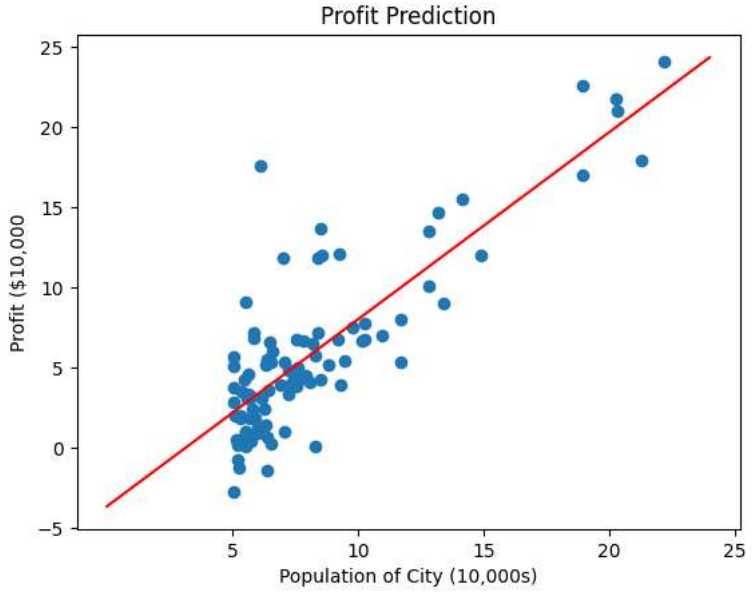
```
plt.plot(J_history)
plt.xlabel("Iteration")
plt.ylabel("$J(\Theta)$")
plt.title("Cost function using Gradient Descent")
```

```
Text(0.5, 1.0, 'Cost function using Gradient Descent')
```




```
plt.scatter(data['a'],data['b'])
x_value=[x for x in range (25)]
y_value=[y*theta[1]+theta[0] for y in x_value]
plt.plot(x_value,y_value, color="r")
plt.xticks(np.arange (5,30,step=5))
plt.yticks(np.arange(-5,30,step=5))
plt.xlabel("Population of City (10,000s)")
plt.ylabel("Profit ($10,000")
plt.title("Profit Prediction")
# Text(0.5, 1.0, 'Profit Prediction')
```

```
Text(0.5, 1.0, 'Profit Prediction')
```




```
def predict (x,theta):
# 11 11 11
# Takes in numpy array of x and theta and return the predicted value of y based on theta
  predictions= np.dot (theta.transpose (),x)
  return predictions[0]


predict1=predict(np.array([1,3.5]),theta)*10000
print("For population = 35,000, we predict a profit of $"+str(round(predict1,0)))
```

```
For population = 35,000, we predict a profit of $4520.0
```

```
predict2=predict(np.array ([1,7]), theta)*10000
print("For population = 70,000, we predict a profit of $"+str(round(predict2,0)))
```

```
For population = 70,000, we predict a profit of $45342.0
```