In [60]:

```
#Kevin Hagler
#Student ID: 801197095
#Homework 0: Linear Regressioin
```

In [61]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
```

In [62]:

```
housing = pd.read_csv("housing.csv")
housing.head()
```

Out[62]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheatin
0	13300000	7420	4	2	3	yes	no	no	n
1	12250000	8960	4	4	4	yes	no	no	n
2	12250000	9960	3	2	2	yes	no	yes	n
3	12215000	7500	4	2	2	yes	no	yes	n
4	11410000	7420	4	1	2	yes	yes	yes	n

In [63]:

```
m = len(housing)
m
```

Out[63]:

```
545
```

In [64]:

```
# All needed functions
def compute_cost (x, y, theta):
    predictions = x.dot(theta)
    errors = np.subtract(predictions, y)
    J = 1 / (2 * m) * np.sum(np.square(errors))
    return J

def gradientDescent(x, y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        predictions = x.dot(theta)
        errors = np.subtract(predictions, y)
        sum_delta = (alpha / m) * x.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history[i] = compute_cost(x, y, theta)
    return theta, cost_history

def gradientDescentProb3(x, y, theta, alpha, iterations, regRate):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        predictions = x.dot(theta)
        errors = np.subtract(predictions, y)
        sum_delta = (alpha / m) * x.transpose().dot(errors);
        theta = [element * (1 - ((alpha*regRate)/m)) for element in theta] - sum_delta
        cost_history[i] = compute_cost(x, y, theta)
    return theta, cost_history
```

In [65]:

```
# map functioin for 1 being yes and 0 being no.
varList = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
def binaryMap(x):
    return x.map({'yes': 1, 'no': 0})

housing[varList] = housing[varList].apply(binaryMap)
housing[:8]
```

Out[65]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheati
0	13300000	7420	4	2	3	1	0	0	
1	12250000	8960	4	4	4	1	0	0	
2	12250000	9960	3	2	2	1	0	1	
3	12215000	7500	4	2	2	1	0	1	
4	11410000	7420	4	1	2	1	1	1	
5	10850000	7500	3	3	1	1	0	1	
6	10150000	8580	4	3	4	1	0	0	
7	10150000	16200	5	3	2	1	0	0	

In [66]:

```
# Slpitting the data into 80% training and 20% testing randomly
np.random.seed(0)
housingTrain, housingTest = train_test_split(housing, train_size = 0.8, test_size = 0.
print(housingTrain.shape)
housingTest.shape
```

```
(436, 13)
```

Out[66]:

```
(109, 13)
```

In [67]:

```
###########
#Problem 1a
###########
```

In [68]:

```
# Training set
price = housingTrain.values[:,0]  # This will be the y value
area = housingTrain.values[:,1]
bedrooms = housingTrain.values[:,2]
bathrooms = housingTrain.values[:,3]
stories = housingTrain.values[:, 4]
parking = housingTrain.values[:, 10]
m = len(housingTrain)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x10 = parking.reshape(m,1)
```

```
# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x10))
print("x values for training :")
print(x[:5])
print ("")
print("y values for training :")
print(y[:5])
```

```
x values for training :
[[1.0 3620 2 1 1 0]
 [1.0 4000 2 1 1 0]
 [1.0 3040 2 1 1 0]
 [1.0 3600 2 1 1 0]
 [1.0 9860 3 1 1 0]]

y values for training :
[[1750000]
 [2695000]
 [2870000]
 [2590000]
 [4515000]]
```

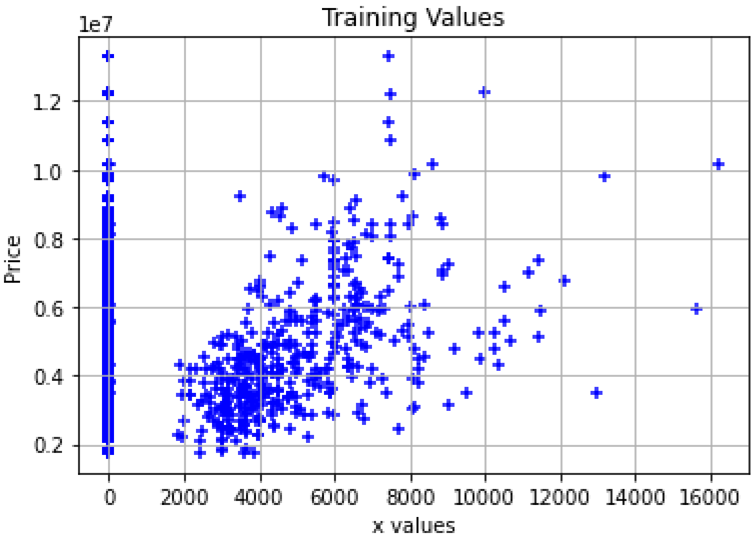
In [69]:

```
theta = np.zeros((6,1))
iterations = 2000
alpha = 0.0000000001
thetaTrain, costTrain = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for training data:")
print(thetaTrain)
print()
print("The final cost for training data:")
print(costTrain)
```

```
Theta for training data:
[[0.2107068750792625]
 [859.3424779371372]
 [0.695613564073667]
 [0.33736375603857444]
 [0.49487481703598357]
 [0.1855985183422298]]

The final cost for training data:
[1.31633712e+13 1.30921973e+13 1.30214655e+13 ... 1.70465861e+12
 1.70465832e+12 1.70465804e+12]
```

In [70]:

```
# Plotting training values
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')
plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [71]:

```
#Testing set for 1a:
price = housingTest.values[:,0]  # This will be the y value
area = housingTest.values[:,1]
bedrooms = housingTest.values[:,2]
bathrooms = housingTest.values[:,3]
stories = housingTest.values[:, 4]
parking = housingTest.values[:, 10]
m = len(housingTest)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x10 = parking.reshape(m,1)

# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x10))
print("x values for testing :")
print(x[:5])
print ("")
print("y values for testing :")
print(y[:5])
```

```
x values for testing :
[[1.0 4000 3 1 2 1]
 [1.0 9620 3 1 1 2]
 [1.0 3460 4 1 2 0]
 [1.0 13200 2 1 1 1]
 [1.0 3660 4 1 2 0]]

y values for testing :
[[4585000]
 [6083000]
 [4007500]
 [6930000]
 [2940000]]
```

In [72]:

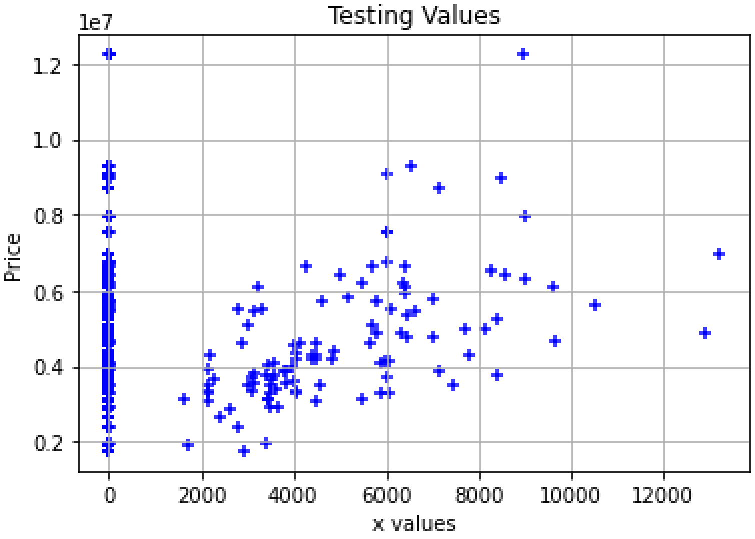
```
theta = np.zeros((6,1))
thetaTest, costTest = gradientDescent(x, y, theta, alpha, iterations)
```

```
print("Theta for testing data:")
print(thetaTest)
print()
print("The final cost for testing data:")
print(costTest)
```

```
Theta for testing data:
[[0.2205630906383037]
 [833.3641986539633]
 [0.7011076886513877]
 [0.3531127590305734]
 [0.5074684039855979]
 [0.13857178422005106]]

The final cost for testing data:
[1.25245899e+13 1.24550016e+13 1.23858541e+13 ... 1.53978999e+12
 1.53978978e+12 1.53978956e+12]
```

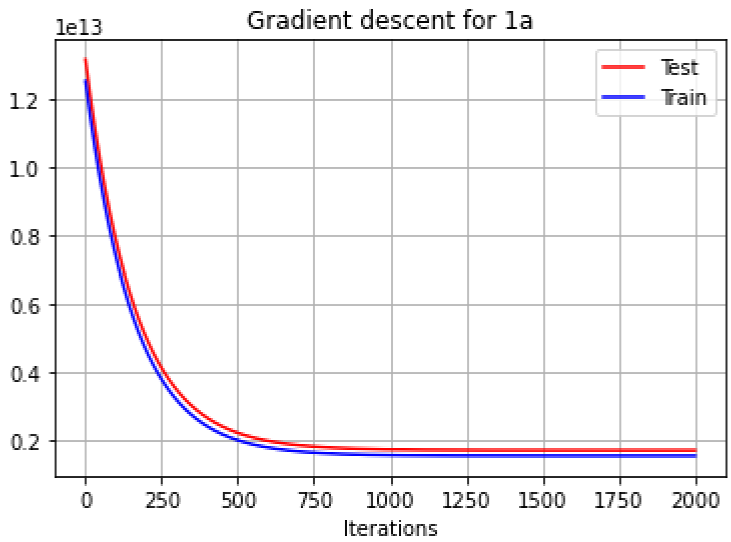
In [73]:

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')
plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Testing Values')
plt.show()
```



In [74]:

```
plt.plot(range (1, iterations +1), costTrain, color= 'red', label = 'Test')
plt.plot(range (1, iterations +1), costTest, color= 'blue', label = 'Train')

#plt.plot(iterations, prevCostTrain, color= 'blue')
plt.xlabel("Iterations")
plt.ylabel("")
plt.title("Gradient descent for 1a")
plt.legend()
plt.grid()
```



In [75]:

```
###########
#Problem 1b
###########
```

In [76]:

```
#Testing set for 1b:
price = housingTest.values[:,0]  # This will be the y value
area = housingTest.values[:,1]
bedrooms = housingTest.values[:,2]
bathrooms = housingTest.values[:,3]
stories = housingTest.values[:, 4]
mainroad = housingTest.values[:, 5]
guestroom = housingTest.values[:, 6]
basement = housingTest.values[:, 7]
hotwaterheating = housingTest.values[:, 8]
airconditioning = housingTest.values[:, 9]
parking = housingTest.values[:, 10]
prefarea = housingTest.values[:, 11]
m = len(housingTest)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x5 = mainroad.reshape(m,1)
x6 = guestroom.reshape(m,1)
x7 = basement.reshape(m,1)
x8 = hotwaterheating.reshape(m,1)
x9 = airconditioning.reshape(m,1)
x10 = parking.reshape(m,1)
x11 = prefarea.reshape(m,1)

# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11))
print("x values for testing :")
print(x[:5])
print ("")
print("y values for testing :")
print(y[:5])
```

```
x values for testing :
[[1.0 4000 3 1 2 1 0 0 0 0 1 0]
 [1.0 9620 3 1 1 1 0 1 0 0 2 1]
 [1.0 3460 4 1 2 1 0 0 0 1 0 0]
 [1.0 13200 2 1 1 1 0 1 1 0 1 0]
 [1.0 3660 4 1 2 0 0 0 0 0 0 0]]

y values for testing :
[[4585000]
 [6083000]
 [4007500]
 [6930000]
 [2940000]]
```

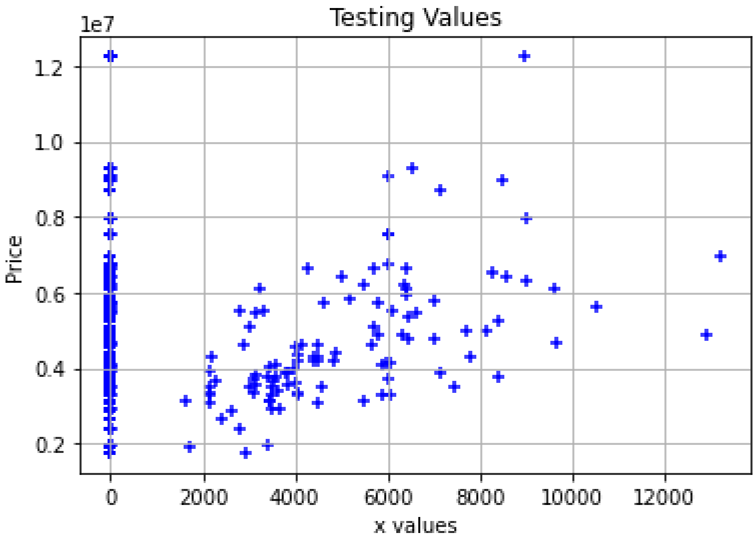
In [77]:

```
theta = np.zeros((12,1))
iterations = 2000
alpha = 0.0000000001
thetaTest, costTest = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for training data:")
print(thetaTest)
print()
print("The final cost for training data:")
print(costTest)
```

```
Theta for training data:
[[0.22056308003657035]
 [833.3641580386068]
 [0.7011076583814723]
 [0.3531127455608999]
 [0.5074683848505653]
 [0.1870646693413445]
 [0.04592777143957116]
 [0.1026908152991846]
 [0.012004189180178627]
 [0.11213058089359688]
 [0.13857178045251137]
 [0.05564844024957321]]

The final cost for training data:
[1.25245899e+13 1.24550016e+13 1.23858540e+13 ... 1.53978986e+12
 1.53978964e+12 1.53978943e+12]
```

In [78]:

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(mainroad, price, color = 'blue', marker='+')
plt.scatter(guestroom, price, color = 'blue', marker='+')
plt.scatter(basement, price, color = 'blue', marker='+')
plt.scatter(hotwaterheating, price, color = 'blue', marker='+')
plt.scatter(airconditioning, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Testing Values')
plt.show()
```



In [79]:

```
#Training set for 1b:
price = housingTrain.values[:,0]  # This will be the y value
area = housingTrain.values[:,1]
bedrooms = housingTrain.values[:,2]
bathrooms = housingTrain.values[:,3]
stories = housingTrain.values[:, 4]
mainroad = housingTrain.values[:, 5]
guestroom = housingTrain.values[:, 6]
basement = housingTrain.values[:, 7]
hotwaterheating = housingTrain.values[:, 8]
airconditioning = housingTrain.values[:, 9]
parking = housingTrain.values[:, 10]
prefarea = housingTrain.values[:, 11]
m = len(housingTrain)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x5 = mainroad.reshape(m,1)
x6 = guestroom.reshape(m,1)
x7 = basement.reshape(m,1)
x8 = hotwaterheating.reshape(m,1)
x9 = airconditioning.reshape(m,1)
x10 = parking.reshape(m,1)
x11 = prefarea.reshape(m,1)


# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11))
print("x values for training :")
print(x[:5])
print ("")
print("y values for training :")
print(y[:5])
```

```
x values for training :
[[1.0 3620 2 1 1 1 0 0 0 0 0 0]
 [1.0 4000 2 1 1 1 0 0 0 0 0 0]
 [1.0 3040 2 1 1 0 0 0 0 0 0 0]
 [1.0 3600 2 1 1 1 0 0 0 0 0 0]
 [1.0 9860 3 1 1 1 0 0 0 0 0 0]]

y values for training :
[[1750000]
 [2695000]
 [2870000]
 [2590000]
 [4515000]]
```

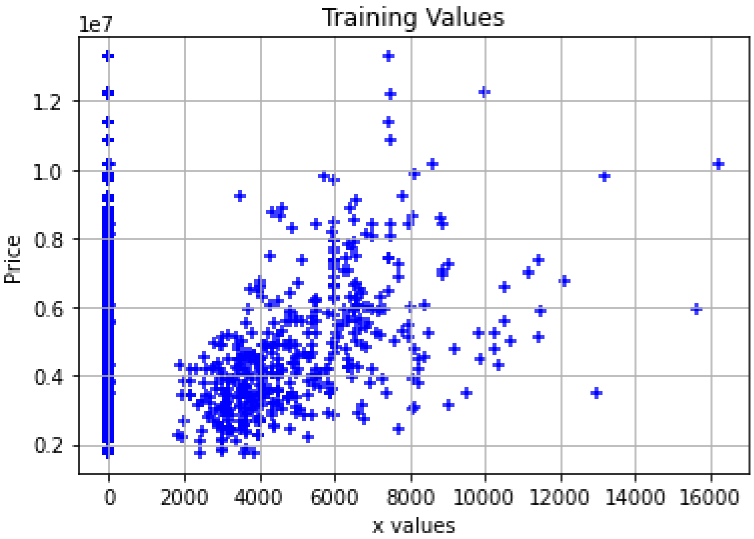
In [80]:

```
theta = np.zeros((12,1))
thetaTrain, costTrain = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for training data:")
print(thetaTrain)
print()
print("The final cost for training data:")
print(costTrain)
```

```
Theta for training data:
[[0.21070686461957314]
 [859.3424343699008]
 [0.6956135322645889]
 [0.3373637419827863]
 [0.49487479653835237]
 [0.19057891973835442]
 [0.060576888136419926]
 [0.10076616809686127]
 [0.019217972547656875]
 [0.11375606938337021]
 [0.18559851020567358]
 [0.0746453668785651]]

The final cost for training data:
[1.31633712e+13 1.30921973e+13 1.30214655e+13 ... 1.70465846e+12
 1.70465818e+12 1.70465790e+12]
```

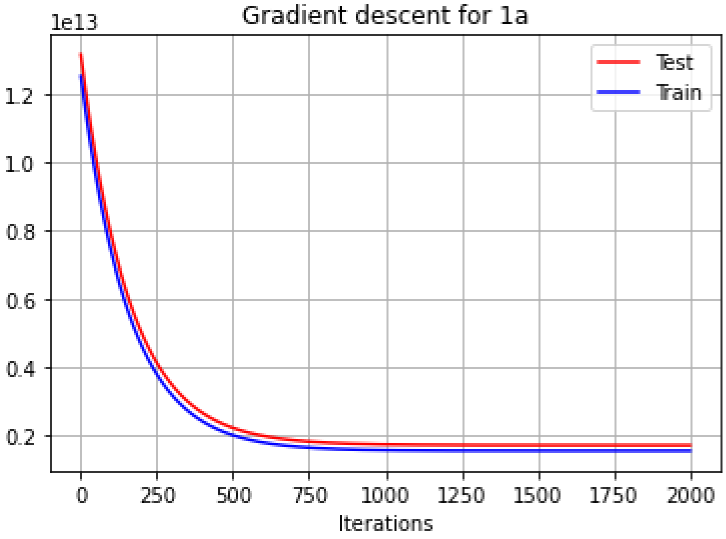
In [81]:

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(mainroad, price, color = 'blue', marker='+')
plt.scatter(guestroom, price, color = 'blue', marker='+')
plt.scatter(basement, price, color = 'blue', marker='+')
plt.scatter(hotwaterheating, price, color = 'blue', marker='+')
plt.scatter(airconditioning, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [82]:

```
plt.plot(range (1, iterations +1), costTrain, color= 'red', label = 'Test')
plt.plot(range (1, iterations +1), costTest, color= 'blue', label = 'Train')

plt.xlabel("Iterations")
plt.ylabel("")
plt.title("Gradient descent for 1a")
plt.legend()
plt.grid()
```



In [83]:

```
###########
#Problem 2a:
#Repeat problem 1a, this time with input normalization and
#input standardization as part of your pre-processing logic.
###########
```

In [84]:

```
# Scailing all values for test and training
varList = ["price", "area", "bedrooms", "bathrooms", "stories", "mainroad", "guestroom
scaler = MinMaxScaler()
housingTest[varList] = scaler.fit_transform(housingTest[varList])
housingTrain[varList] = scaler.fit_transform(housingTrain[varList])
print("Train values: ", len(housingTrain), ":")
print(housingTrain[:5])
print()
print("Test values: ", len(housingTest), ":")
housingTest[:5]
```

```
Train values:  436 :
        price      area  bedrooms  bathrooms  stories  mainroad  guestroom  \
542  0.000000  0.124199       0.2        0.0      0.0       1.0        0.0   
496  0.081818  0.150654       0.2        0.0      0.0       1.0        0.0   
484  0.096970  0.083821       0.2        0.0      0.0       0.0        0.0   
507  0.072727  0.122807       0.2        0.0      0.0       1.0        0.0   
252  0.239394  0.558619       0.4        0.0      0.0       1.0        0.0   

     basement  hotwaterheating  airconditioning  parking  prefarea  \
542       0.0              0.0              0.0      0.0       0.0   
496       0.0              0.0              0.0      0.0       0.0   
484       0.0              0.0              0.0      0.0       0.0   
507       0.0              0.0              0.0      0.0       0.0   
252       0.0              0.0              0.0      0.0       0.0   

    furnishingstatus  
542      unfurnished  
496      unfurnished  
484      unfurnished  
507      unfurnished  
252   semi-furnished  

Test values:  109 :
```

Out[84]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwate
239	0.270000	0.203463	0.50	0.0	0.333333	1.0	0.0	0.0	
113	0.412667	0.690043	0.50	0.0	0.000000	1.0	0.0	1.0	
325	0.215000	0.156710	0.75	0.0	0.333333	1.0	0.0	0.0	
66	0.493333	1.000000	0.25	0.0	0.000000	1.0	0.0	1.0	
479	0.113333	0.174026	0.75	0.0	0.333333	0.0	0.0	0.0	

In [85]:

```
price = housingTest.values[:,0]  # This will be the y value
area = housingTest.values[:,1]
bedrooms = housingTest.values[:,2]
bathrooms = housingTest.values[:,3]
stories = housingTest.values[:, 4]
parking = housingTest.values[:, 10]
m = len(housingTest)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x10 = parking.reshape(m,1)

# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x10))
print("x values for testing :")
print(x[:5])
```

```
print ("")
print("y values for testing:")
print(y[:5])
```

```
x values for testing :
[[1.0 0.20346320346320346 0.5 0.0 0.3333333333333333 0.3333333333333333]
 [1.0 0.69004329004329 0.5 0.0 0.0 0.6666666666666666]
 [1.0 0.15670995670995674 0.75 0.0 0.3333333333333333 0.0]
 [1.0 1.0 0.25 0.0 0.0 0.3333333333333333]
 [1.0 0.17402597402597403 0.75 0.0 0.3333333333333333 0.0]]

y values for testing:
[[0.27]
 [0.41266666666666674]
 [0.215]
 [0.4933333333333334]
 [0.11333333333333337]]
```

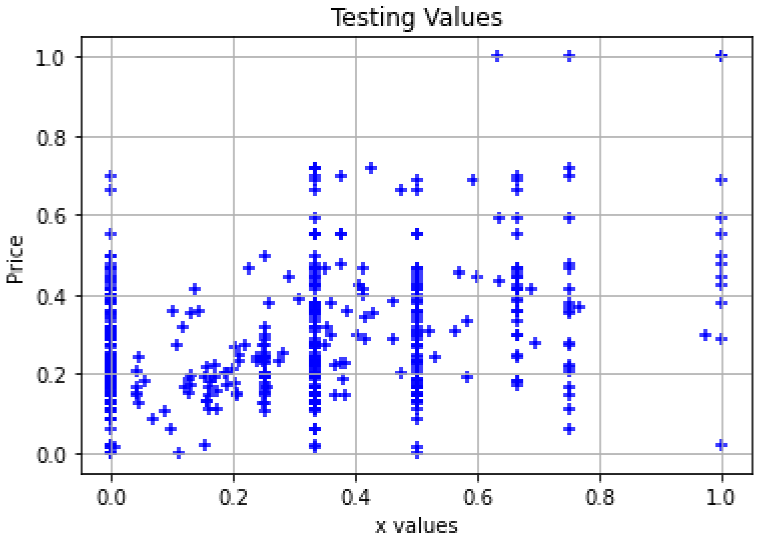
In [86]:

```
theta = np.zeros((6,1))
iterations = 1500
alpha = 0.01
thetaTest, costTest = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for testing data:")
print(thetaTest)
print()
print("The final cost for testing data:")
print(costTest)
```

```
Theta for testing data:
[[0.11485880555642139]
 [0.17487398621902986]
 [0.07207542236071565]
 [0.14157047522898814]
 [0.1533454916129055]
 [0.1401905385106042]]

The final cost for testing data:
[0.05186972 0.05062113 0.04940901 ... 0.00567301 0.00567221 0.00567141]
```

In [87]:

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Testing Values')
plt.show()
```



In [88]:

```
# Training set
price = housingTrain.values[:,0]  # This will be the y value
area = housingTrain.values[:,1]
bedrooms = housingTrain.values[:,2]
bathrooms = housingTrain.values[:,3]
stories = housingTrain.values[:, 4]
parking = housingTrain.values[:, 10]
m = len(housingTrain)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x10 = parking.reshape(m,1)

# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x10))
print("x values for training :")
print(x[:5])
print ("")
print("y values for training :")
print(y[:5])
```

```
x values for training :
[[1.0 0.12419938735728209 0.2 0.0 0.0 0.0]
 [1.0 0.15065441381230854 0.2 0.0 0.0 0.0]
 [1.0 0.08382066276803118 0.2 0.0 0.0 0.0]
 [1.0 0.12280701754385964 0.2 0.0 0.0 0.0]
 [1.0 0.5586187691450848 0.4000000000000001 0.0 0.0 0.0]]

y values for training :
[[0.0]
 [0.08181818181818179]
 [0.09696969696969696]
 [0.07272727272727272]
 [0.23939393939393935]]
```

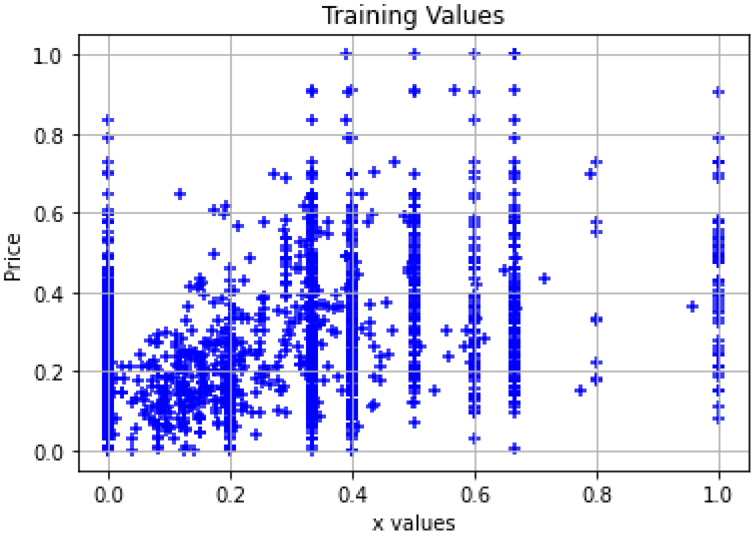
In [89]:

```
thetaTrain, costTrain = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for training data:")
print(thetaTrain)
print()
print("The final cost for training data:")
print(thetaTrain)
```

```
Theta for training data:
[[0.10838120074159972]
 [0.14785905431069957]
 [0.09500434634530074]
 [0.1530269553242469]
 [0.13035076815124277]
 [0.12049269833412143]]

The final cost for training data:
[[0.10838120074159972]
 [0.14785905431069957]
 [0.09500434634530074]
 [0.1530269553242469]
 [0.13035076815124277]
 [0.12049269833412143]]
```

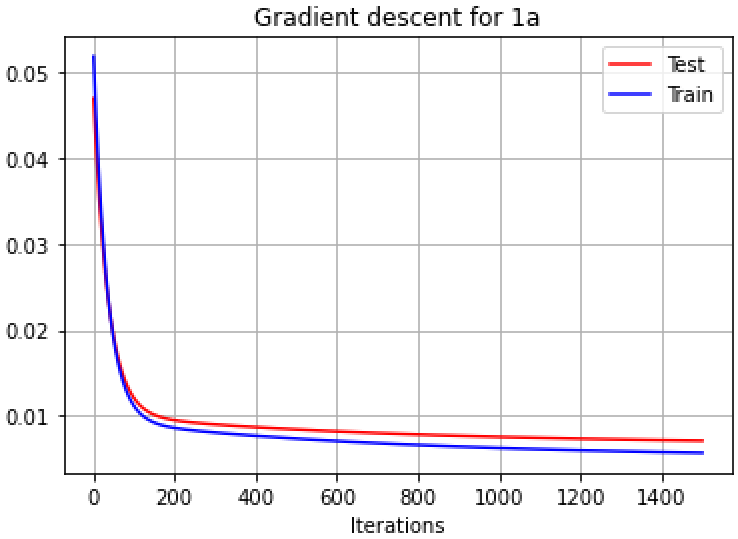
In [90]:

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [91]:

```
plt.plot(range (1, iterations +1), costTrain, color= 'red', label = 'Test')
plt.plot(range (1, iterations +1), costTest, color= 'blue', label = 'Train')

#plt.plot(iterations, prevCostTrain, color= 'blue')
plt.xlabel("Iterations")
plt.ylabel("")
plt.title("Gradient descent for 1a")
plt.legend()
plt.grid()
```



In [92]:

```
###########
#Problem 2b:
#Repeat problem 1b, this time with input normalization and
#input standardization as part of your pre-processing logic.
###########
```

In [93]:

```
price = housingTest.values[:,0]  # This will be the y value
area = housingTest.values[:,1]
bedrooms = housingTest.values[:,2]
bathrooms = housingTest.values[:,3]
stories = housingTest.values[:, 4]
mainroad = housingTest.values[:, 5]
guestroom = housingTest.values[:, 6]
basement = housingTest.values[:, 7]
hotwaterheating = housingTest.values[:, 8]
airconditioning = housingTest.values[:, 9]
parking = housingTest.values[:, 10]
prefarea = housingTest.values[:, 11]
m = len(housingTest)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x5 = mainroad.reshape(m,1)
x6 = guestroom.reshape(m,1)
x7 = basement.reshape(m,1)
x8 = hotwaterheating.reshape(m,1)
x9 = airconditioning.reshape(m,1)
x10 = parking.reshape(m,1)
x11 = prefarea.reshape(m,1)


# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11))
print("x values for testing :")
print(x[:5])
print ("")
```

```
print("y values for testing :")
print(y[:5])
```

```
x values for testing :
[[1.0 0.2034632034632034 6 0.5 0.0 0.3333333333333333 1.0 0.0 0.0 0.0 0.0
  0.3333333333333333 0.0]
 [1.0 0.69004329004329 0.5 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.6666666666666666
  1.0]
 [1.0 0.15670995670995674 0.75 0.0 0.3333333333333333 1.0 0.0 0.0 0.0 1.0
  0.0 0.0]
 [1.0 1.0 0.25 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.3333333333333333 0.0]
 [1.0 0.17402597402597403 0.75 0.0 0.3333333333333333 0.0 0.0 0.0 0.0 0.0
  0.0 0.0]]

y values for testing :
[[0.27]
 [0.41266666666666674]
 [0.215]
 [0.49333333333333334]
 [0.11333333333333337]]
```

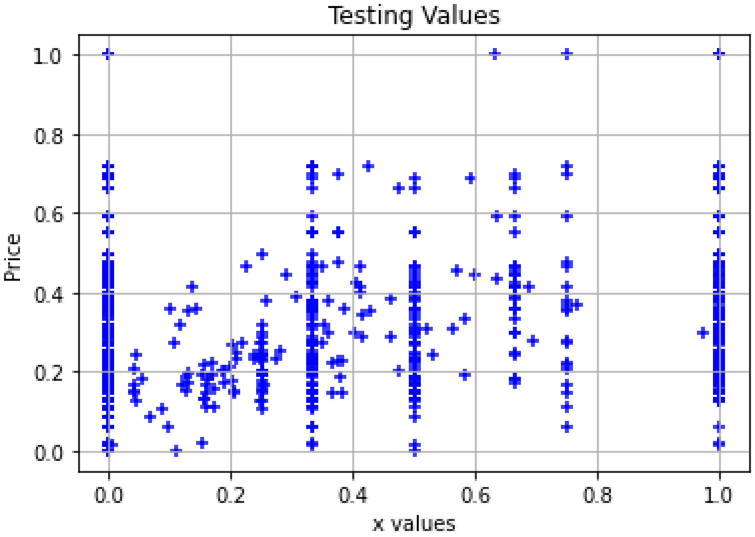
In [94]:

```
iterations = 1500
alpha = 0.1

theta = np.zeros((12,1))
thetaTest, costTest = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for testing data:")
print(thetaTest)
print()
print("The final cost for testing data:")
print(costTest[:4])
```

```
Theta for testing data:
[[0.013779490423040972]
 [0.21651445506268402]
 [0.015224850625834757]
 [0.2657005120867469]
 [0.1359174982935238]
 [0.05576791183291983]
 [-0.027423466369392895]
 [0.06441043582884905]
 [0.029372408580139318]
 [0.1003066032805787]
 [0.1486494851422313]
 [0.016039816971516508]]

The final cost for testing data:
[0.03281147 0.02166061 0.01552498 0.01212623]
```

In [95]:

```
plt.scatter(area, price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(mainroad, price, color = 'blue', marker='+')
plt.scatter(guestroom, price, color = 'blue', marker='+')
plt.scatter(basement, price, color = 'blue', marker='+')
plt.scatter(hotwaterheating, price, color = 'blue', marker='+')
plt.scatter(airconditioning, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')
plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Testing Values')
plt.show()
```



In [96]:

```
price = housingTrain.values[:,0]  # This will be the y value
area = housingTrain.values[:,1]
bedrooms = housingTrain.values[:,2]
bathrooms = housingTrain.values[:,3]
stories = housingTrain.values[:, 4]
mainroad = housingTrain.values[:, 5]
guestroom = housingTrain.values[:, 6]
basement = housingTrain.values[:, 7]
hotwaterheating = housingTrain.values[:, 8]
airconditioning = housingTrain.values[:, 9]
parking = housingTrain.values[:, 10]
prefarea = housingTrain.values[:, 11]
m = len(housingTrain)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x5 = mainroad.reshape(m,1)
x6 = guestroom.reshape(m,1)
x7 = basement.reshape(m,1)
x8 = hotwaterheating.reshape(m,1)
x9 = airconditioning.reshape(m,1)
x10 = parking.reshape(m,1)
x11 = prefarea.reshape(m,1)


# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11))
print("x values for training :")
print(x[:5])
print ("")
```

```
print("y values for training :")
print(y[:5])
```

```
x values for training :
[[1.0 0.12419938735728209 0.2 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.15065441381230854 0.2 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.08382066276803118 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.12280701754385964 0.2 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.5586187691450848 0.4000000000000001 0.0 0.0 1.0 0.0 0.0 0.0 0.0
  0.0 0.0]]

y values for training :
[[0.0]
 [0.08181818181818179]
 [0.09696969696969696]
 [0.07272727272727272]
 [0.23939393939393935]]
```

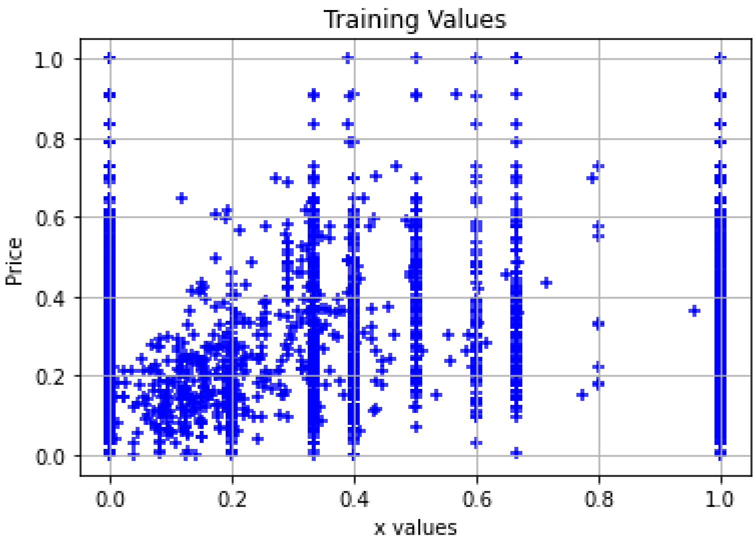
In [97]:

```
thetaTrain, costTrain = gradientDescent(x, y, theta, alpha, iterations)
print("Theta for training data:")
print(thetaTrain)
print()
print("The final cost for training data:")
print(costTrain[:4])
```

```
Theta for training data:
[[-0.002091560853722144]
 [0.29272369579438445]
 [0.0665762401745976]
 [0.1778248352255282]
 [0.11216513523269021]
 [0.043279100485721124]
 [0.03911178584312893]
 [0.029527549184113176]
 [0.10597747254457737]
 [0.0786535143705551]
 [0.06742557232164764]
 [0.06355495281933218]]

The final cost for training data:
[0.03044483 0.02062033 0.01511105 0.01200624]
```

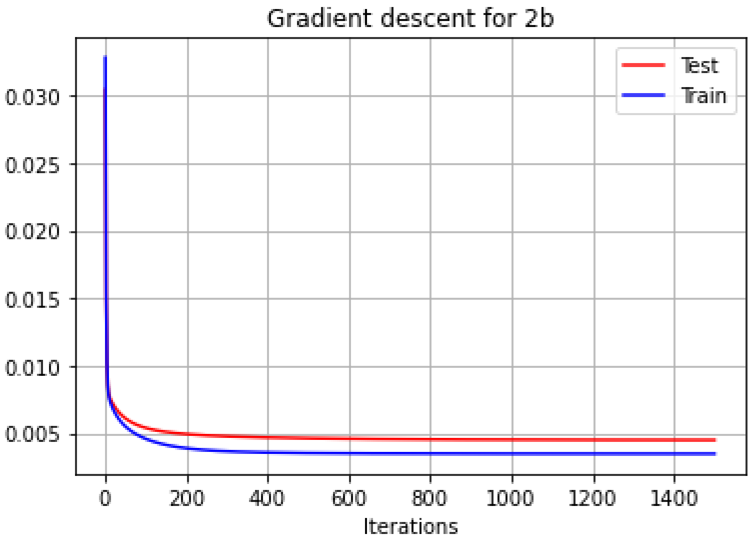
In [98]:

```
plt.scatter(area, price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(mainroad, price, color = 'blue', marker='+')
plt.scatter(guestroom, price, color = 'blue', marker='+')
plt.scatter(basement, price, color = 'blue', marker='+')
plt.scatter(hotwaterheating, price, color = 'blue', marker='+')
plt.scatter(airconditioning, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [99]:

```
plt.plot(range (1, iterations +1), costTrain, color= 'red', label = 'Test')
plt.plot(range (1, iterations +1), costTest, color= 'blue', label = 'Train')

#plt.plot(iterations, prevCostTrain, color= 'blue')
plt.xlabel("Iterations")
plt.ylabel("")
plt.title("Gradient descent for 2b")
plt.legend()
plt.grid()
```



In [100…

```
###########
#Problem 3a
###########
```

In [101…

```
#ensuring the data is still normalized
housingTest[:5]
```

Out[101]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwate
239	0.270000	0.203463	0.50	0.0	0.333333	1.0	0.0	0.0	
113	0.412667	0.690043	0.50	0.0	0.000000	1.0	0.0	1.0	
325	0.215000	0.156710	0.75	0.0	0.333333	1.0	0.0	0.0	
66	0.493333	1.000000	0.25	0.0	0.000000	1.0	0.0	1.0	
479	0.113333	0.174026	0.75	0.0	0.333333	0.0	0.0	0.0	

In [102…

```
housingTrain[:5]
```

Out[102]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterh
542	0.000000	0.124199	0.2	0.0	0.0	1.0	0.0	0.0	
496	0.081818	0.150654	0.2	0.0	0.0	1.0	0.0	0.0	
484	0.096970	0.083821	0.2	0.0	0.0	0.0	0.0	0.0	
507	0.072727	0.122807	0.2	0.0	0.0	1.0	0.0	0.0	
252	0.239394	0.558619	0.4	0.0	0.0	1.0	0.0	0.0	

In [103…

```python
price = housingTest.values[:,0]  # This will be the y value
area = housingTest.values[:,1]
bedrooms = housingTest.values[:,2]
bathrooms = housingTest.values[:,3]
stories = housingTest.values[:, 4]
parking = housingTest.values[:, 10]
m = len(housingTest)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x10 = parking.reshape(m,1)

# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x10))
print("x values for testing :")
print(x[:5])
print ("")
print("y values for testing:")
print(y[:5])
theta = np.zeros((6,1))
```

```
x values for testing :
[[1.0 0.20346320346320346 0.5 0.0 0.3333333333333333 0.3333333333333333]
 [1.0 0.69004329004329 0.5 0.0 0.0 0.6666666666666666]
 [1.0 0.15670995670995674 0.75 0.0 0.3333333333333333 0.0]
 [1.0 1.0 0.25 0.0 0.0 0.3333333333333333]
 [1.0 0.17402597402597403 0.75 0.0 0.3333333333333333 0.0]]

y values for testing:
[[0.27]
 [0.41266666666666674]
 [0.215]
 [0.4933333333333334]
 [0.11333333333333337]]
```

In [104…

```
# adjustable variables:
iterations = 1500
alpha = 0.01
regRate = 10
```

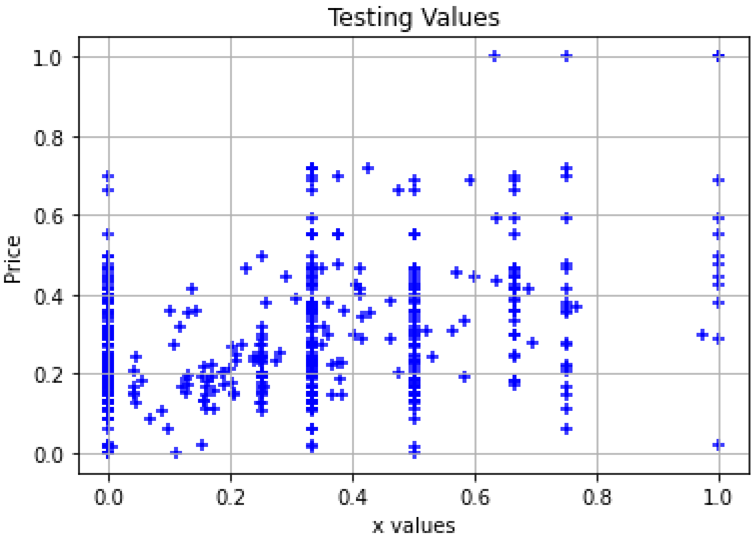
In [105…

```
thetaTest, costTest = gradientDescentProb3(x, y, theta, alpha, iterations, regRate)
print("Theta for testing data:")
print(thetaTest)
print()
print("The final cost for testing data:")
print(costTest[:4])
```

```
Theta for testing data:
[[0.13502524533883137]
 [0.1253313253724721]
 [0.07952944735821464]
 [0.09261297689410655]
 [0.1129562809173663]
 [0.10132031727875634]]

The final cost for testing data:
[0.05186972 0.05062228 0.04941239 0.04823889]
```

In [106…

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Testing Values')
plt.show()
```



In [107…

```
price = housingTrain.values[:,0]  # This will be the y value
area = housingTrain.values[:,1]
bedrooms = housingTrain.values[:,2]
bathrooms = housingTrain.values[:,3]
stories = housingTrain.values[:, 4]
parking = housingTrain.values[:, 10]
m = len(housingTrain)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x10 = parking.reshape(m,1)

# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x10))
print("x values for training :")
print(x[:5])
print ("")
print("y values for training:")
print(y[:5])
```

```
x values for training :
[[1.0 0.12419938735728209 0.2 0.0 0.0 0.0]
 [1.0 0.15065441381230854 0.2 0.0 0.0 0.0]
 [1.0 0.08382066276803118 0.2 0.0 0.0 0.0]
 [1.0 0.12280701754385964 0.2 0.0 0.0 0.0]
 [1.0 0.5586187691450848 0.4000000000000001 0.0 0.0 0.0]]

y values for training:
[[0.0]
 [0.08181818181818179]
 [0.09696969696969696]
 [0.07272727272727272]
 [0.23939393939393935]]
```

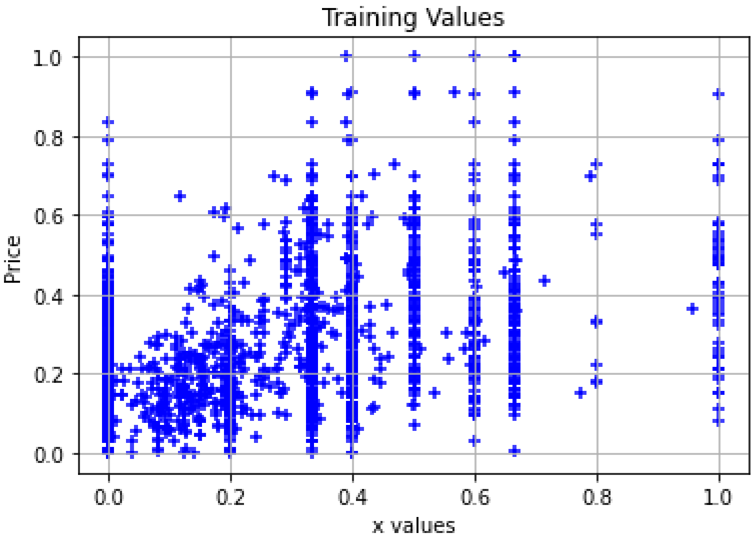
In [108…

```
thetaTrain, costTrain = gradientDescentProb3(x, y, theta, alpha, iterations, regRate)
print("Theta for training data:")
print(thetaTrain)
print()
print("The final cost for training data:")
print(costTrain[:4])
```

```
Theta for training data:
[[0.11634400857604375]
 [0.13260421650312162]
 [0.09185345650047384]
 [0.13717602726160016]
 [0.12112900476608221]
 [0.11178689613746272]]

The final cost for training data:
[0.04697501 0.04596321 0.04497923 0.04402231]
```

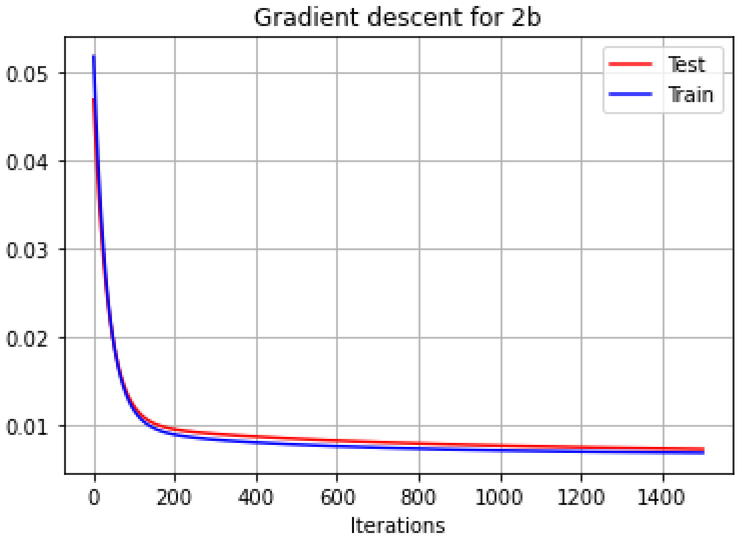
In [109…

```
plt.scatter(area,price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [110…

```
plt.plot(range (1, iterations +1), costTrain, color= 'red', label = 'Test')
plt.plot(range (1, iterations +1), costTest, color= 'blue', label = 'Train')

#plt.plot(iterations, prevCostTrain, color= 'blue')
plt.xlabel("Iterations")
plt.ylabel("")
plt.title("Gradient descent for 2b")
plt.legend()
plt.grid()
```



In [111…

```
###########
#Problem 3b
###########
```

In [112…

```
price = housingTest.values[:,0]  # This will be the y value
area = housingTest.values[:,1]
bedrooms = housingTest.values[:,2]
bathrooms = housingTest.values[:,3]
stories = housingTest.values[:, 4]
mainroad = housingTest.values[:, 5]
guestroom = housingTest.values[:, 6]
basement = housingTest.values[:, 7]
hotwaterheating = housingTest.values[:, 8]
airconditioning = housingTest.values[:, 9]
parking = housingTest.values[:, 10]
prefarea = housingTest.values[:, 11]
m = len(housingTest)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x5 = mainroad.reshape(m,1)
x6 = guestroom.reshape(m,1)
x7 = basement.reshape(m,1)
x8 = hotwaterheating.reshape(m,1)
x9 = airconditioning.reshape(m,1)
x10 = parking.reshape(m,1)
x11 = prefarea.reshape(m,1)


# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11))
print("x values for testing :")
print(x[:5])
print ("")
print("y values for testing :")
print(y[:5])
```

```
x values for testing :
[[1.0 0.20346320346320346 0.5 0.0 0.3333333333333333 1.0 0.0 0.0 0.0 0.0
  0.3333333333333333 0.0]
 [1.0 0.69004329004329 0.5 0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.6666666666666666
  1.0]
 [1.0 0.15670995670995674 0.75 0.0 0.3333333333333333 1.0 0.0 0.0 0.0 1.0
  0.0 0.0]
 [1.0 1.0 0.25 0.0 0.0 1.0 0.0 1.0 1.0 0.0 0.3333333333333333 0.0]
 [1.0 0.17402597402597403 0.75 0.0 0.3333333333333333 0.0 0.0 0.0 0.0 0.0
  0.0 0.0]]

y values for testing :
[[0.27]
 [0.41266666666666674]
 [0.215]
 [0.4933333333333334]
 [0.11333333333333337]]
```

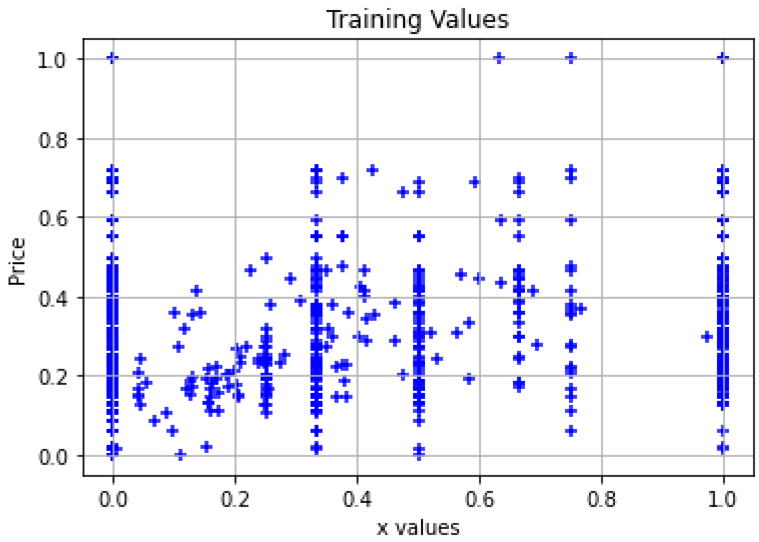
In [113…

```
iterations = 1500
alpha = 0.01
regRate = 5

theta = np.zeros((12,1))
thetaTest, costTest = gradientDescentProb3(x, y, theta, alpha, iterations, regRate)
print("Theta for testing data:")
print(thetaTest)
print()
print("The final cost for testing data:")
print(costTest[:4])
```

```
Theta for testing data:
[[0.05366192062167915]
 [0.10371469806718317]
 [0.0603239105389004]
 [0.10092848278902036]
 [0.0977813420737624]
 [0.07450973620197783]
 [0.00270318807560269]
 [0.03638025199260598]
 [0.013828961564551039]
 [0.10114176740401092]
 [0.10140570991061192]
 [0.013143342033126127]]

The final cost for testing data:
[0.0508477  0.04865903 0.04658363 0.0446156 ]
```

In [114…

```
plt.scatter(area, price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(mainroad, price, color = 'blue', marker='+')
plt.scatter(guestroom, price, color = 'blue', marker='+')
plt.scatter(basement, price, color = 'blue', marker='+')
plt.scatter(hotwaterheating, price, color = 'blue', marker='+')
plt.scatter(airconditioning, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [115…

```
price = housingTrain.values[:,0]  # This will be the y value
area = housingTrain.values[:,1]
bedrooms = housingTrain.values[:,2]
bathrooms = housingTrain.values[:,3]
stories = housingTrain.values[:, 4]
mainroad = housingTrain.values[:, 5]
guestroom = housingTrain.values[:, 6]
basement = housingTrain.values[:, 7]
hotwaterheating = housingTrain.values[:, 8]
airconditioning = housingTrain.values[:, 9]
parking = housingTrain.values[:, 10]
prefarea = housingTrain.values[:, 11]
m = len(housingTrain)

#reshaping all arrays
y = price.reshape(m,1)
x0 = np.ones((m,1))
x1 = area.reshape(m,1)
x2 = bedrooms.reshape(m,1)
x3 = bathrooms.reshape(m,1)
x4 = stories.reshape(m,1)
x5 = mainroad.reshape(m,1)
x6 = guestroom.reshape(m,1)
x7 = basement.reshape(m,1)
x8 = hotwaterheating.reshape(m,1)
x9 = airconditioning.reshape(m,1)
x10 = parking.reshape(m,1)
x11 = prefarea.reshape(m,1)


# Combining all x values
x = np.hstack((x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11))
print("x values for testing :")
print(x[:5])
print ("")
print("y values for testing :")
print(y[:5])
```

```
x values for testing :
[[1.0 0.12419938735728209 0.2 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.15065441381230854 0.2 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.08382066276803118 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.12280701754385964 0.2 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0]
 [1.0 0.5586187691450848 0.4000000000000001 0.0 0.0 1.0 0.0 0.0 0.0 0.0
  0.0 0.0]]

y values for testing :
[[0.0]
 [0.08181818181818179]
 [0.09696969696969696]
 [0.07272727272727272]
 [0.23939393939393935]]
```

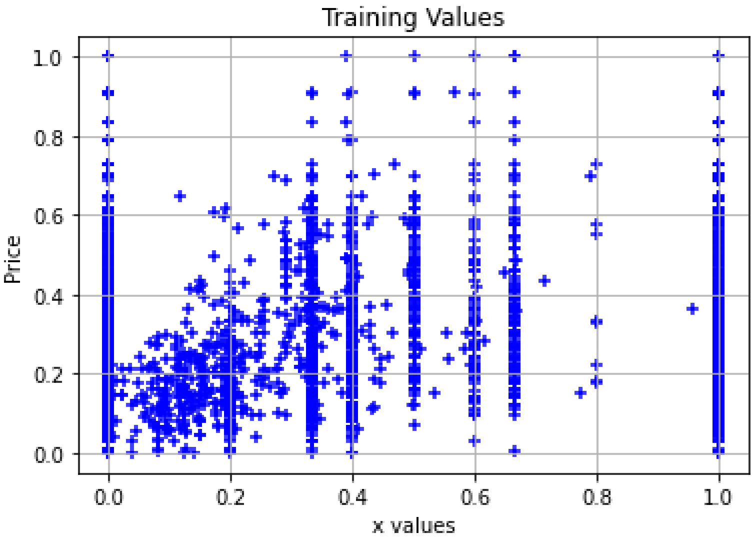
In [116…

```
thetaTrain, costTrain = gradientDescentProb3(x, y, theta, alpha, iterations, regRate)
print("Theta for training data:")
print(thetaTrain)
print()
print("The final cost for training data:")
print(costTrain[:4])
```

```
Theta for training data:
[[0.041843436484087 77]
 [0.09181972578591903]
 [0.06066985330064156]
 [0.12124714911743986]
 [0.09665993764762791]
 [0.05498223174214131]
 [0.04564730043740319]
 [0.027819229141241117]
 [0.04781343960130446]
 [0.08965462174708529]
 [0.08142090773921845]
 [0.07395201682278775]]

The final cost for training data:
[0.04602978 0.04414338 0.04235125 0.04064865]
```

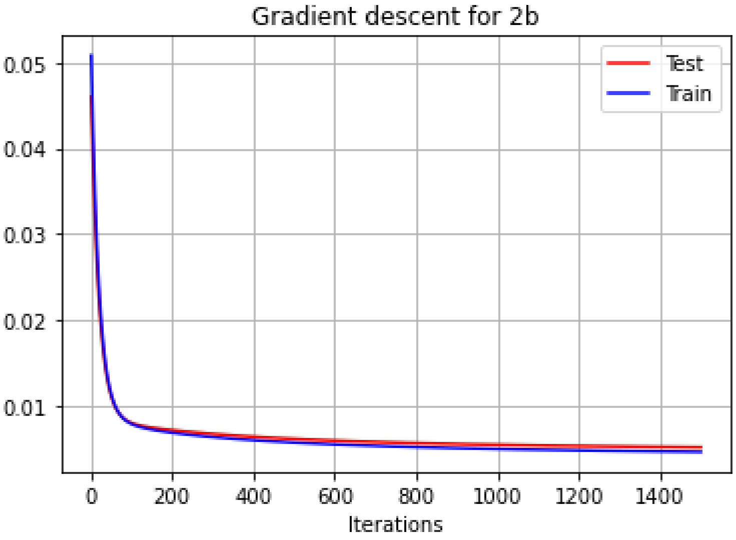
In [117…

```
plt.scatter(area, price, color = 'blue', marker='+')
plt.scatter(bedrooms, price, color = 'blue', marker='+')
plt.scatter(bathrooms, price, color = 'blue', marker='+')
plt.scatter(stories, price, color = 'blue', marker='+')
plt.scatter(mainroad, price, color = 'blue', marker='+')
plt.scatter(guestroom, price, color = 'blue', marker='+')
plt.scatter(basement, price, color = 'blue', marker='+')
plt.scatter(hotwaterheating, price, color = 'blue', marker='+')
plt.scatter(airconditioning, price, color = 'blue', marker='+')
plt.scatter(parking, price, color = 'blue', marker='+')

plt.grid()
plt.ylabel('Price')
plt.xlabel('x values')
plt.title('Training Values')
plt.show()
```



In [118…

```
plt.plot(range (1, iterations +1), costTrain, color= 'red', label = 'Test')
plt.plot(range (1, iterations +1), costTest, color= 'blue', label = 'Train')

#plt.plot(iterations, prevCostTrain, color= 'blue')
plt.xlabel("Iterations")
plt.ylabel("")
plt.title("Gradient descent for 2b")
plt.legend()
plt.grid()
```


Gradient descent for 2b
Test
Train
Iterations