In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
housing = pd.DataFrame(pd.read_csv("Housing.csv"))
housing.head()
```

Out[3]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished

In [4]:
```
m=len(housing)
m
```

Out[4]:
```
545
```

In [5]:
```
housing.shape
```

Out[5]:
```
(545, 13)
```

In [6]:
```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

In [7]:
```
housing.describe()
```

Out[7]:

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

In [8]:
```
varlist = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']

def binary_map(x):
    return x.map({'yes' : 1, "no" : 0})

housing[varlist] = housing[varlist].apply(binary_map)
housing.head()
```

Out[8]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	furnished
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	furnished
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	semi-furnished
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	furnished
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	furnished

In [9]:
```
from sklearn.model_selection import train_test_split

np.random.seed(0)
df_train, df_test = train_test_split(housing, train_size = 0.7, test_size = 0.3, random_state = np.random)
df_train.shape
```

Out[9]:
```
(381, 13)
```

In [10]:
```
num_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking', 'price']
df_Newtrain = df_train[num_vars]
df_Newtest = df_test[num_vars]
df_Newtrain.head()
```

Out[10]:

	area	bedrooms	bathrooms	stories	parking	price
454	4500	3	1	2	0	3143000
382	3990	3	1	2	0	3500000
231	4320	3	1	1	0	4690000
271	1905	5	1	2	0	4340000
250	3510	3	1	3	0	4515000

In [11]:
```
df_Newtest.head()
```

Out[11]:

	area	bedrooms	bathrooms	stories	parking	price
239	4000	3	1	2	1	4585000
113	9620	3	1	1	2	6083000
325	3460	4	1	2	0	4007500
66	13200	2	1	1	1	6930000
479	3660	4	1	2	0	2940000

In [12]:
```
def hypothesis(theta, X, n):
    h = np.ones((X.shape[0],1))
    theta = theta.reshape(1,n+1)
    for i in range(0,X.shape[0]):
        h[i] = float(np.matmul(theta, X[i]))
    h = h.reshape(X.shape[0])
    return h
```

In [24]:
```
def gradientDescent(theta, alpha, num_iter, h, X, y, n):
    cost = np.ones(num_iter)
    for i in range(0,num_iter):
        theta[0] = theta[0] - (alpha/X.shape[0]) * sum(h - y)
        for j in range(1,n+1):
            theta[j] = theta[j] - (alpha/X.shape[0]) * sum((h-y) * X.transpose()[j])
        h = hypothesis(theta, X, n)
        cost[i] = (1/X.shape[0]) * 0.5 * sum(np.square(h - y))
    theta = theta.reshape(1,n+1)
    return theta, cost
```

In [25]:
```
def linearRegression(X, y, alpha, num_iter):
    n = X.shape[1]
    column = np.ones((X.shape[0],1))
    X = np.concatenate((column, X), axis = 1)
    theta = np.zeros(n+1)
    h = hypothesis(theta, X, n)
    theta, cost = GD(theta,alpha,num_iter,h,X,y,n)
    return theta, cost
```

In [26]:
```
X_t = df_Newtrain.values[:,[0,1,2,3,4]]
Y_t = df_Newtrain.values[:,5]

X_v = df_Newtest.values[:,[0,1,2,3,4]]
Y_v = df_Newtest.values[:,5]
```

In [27]:
```
mean = np.ones(X_t.shape[1])
std = np.ones(X_t.shape[1])
for i in range(0, X_t.shape[1]):
    mean[i] = np.mean(X_t.transpose()[i])
    std[i] = np.std(X_t.transpose()[i])
    for j in range(0, X_t.shape[0]):
        X_t[j][i] = (X_t[j][i] - mean[i])/std[i]
mean = np.ones(X_v.shape[1])
std = np.ones(X_v.shape[1])
for i in range(0, X_v.shape[1]):
    mean[i] = np.mean(X_v.transpose()[i])
    std[i] = np.std(X_v.transpose()[i])
    for j in range(0, X_v.shape[0]):
        X_v[j][i] = (X_v[j][i] - mean[i])/std[i]
```

In [28]:
```
df_Newtrain.shape
```

Out[28]:
```
(381, 11)
```

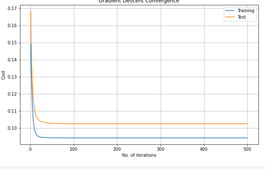
In [29]:
```
theta, cost = linear_regression(X_t, Y_t, 0.1, 500)
cost = list(cost)

theta2, cost2 = linear_regression(X_v, Y_v, 0.1, 500)
cost2 = list(cost2)

n_iterations = [x for x in range(1,501)]
n_iterations2 = [x for x in range(1,501)]

plt.plot(n_iterations, cost, label = 'Training')
plt.plot(n_iterations2, cost2, label = 'Test')
plt.legend()
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Gradient Descent Convergence')
```

Out[29]:
```
Text(0.5, 1.0, 'Gradient Descent Convergence')
```



In [30]:
```
num_vars = ['area', 'bedrooms', 'bathrooms', 'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'parking', 'prefarea', 'price']
df_Newtrain = df_train[num_vars]
df_Newtest = df_test[num_vars]
df_Newtrain.head()
```

Out[30]:

	area	bedrooms	bathrooms	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	price
454	4500	3	1	1	0	0	0	1	0	0	3143000
382	3990	3	1	1	0	0	0	0	0	0	3500000
231	4320	3	1	1	0	0	0	0	0	1	4690000
271	1905	5	1	0	0	1	0	0	0	0	4340000
250	3510	3	1	1	0	0	0	0	0	0	4515000

In [31]:
```
df_Newtest.head()
```

Out[31]:

	area	bedrooms	bathrooms	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	price
239	4000	3	1	1	0	0	0	0	1	0	4585000
113	9620	3	1	1	0	1	0	0	2	1	6083000
325	3460	4	1	1	0	0	0	1	0	0	4007500
66	13200	2	1	1	0	1	1	0	1	0	6930000
479	3660	4	1	0	0	0	0	0	0	0	2940000

In [32]:
```
X_v = df_Newtest.values[:,0:10]
Y_v = df_Newtest.values[:,10]
```

In [33]:
```
mean = np.ones(X_t.shape[1])
std = np.ones(X_t.shape[1])
for i in range(0, X_t.shape[1]):
    mean[i] = np.mean(X_t.transpose()[i])
    std[i] = np.std(X_t.transpose()[i])
    for j in range(0, X_t.shape[0]):
        X_t[j][i] = (X_t[j][i] - mean[i])/std[i]
mean = np.ones(X_v.shape[1])
std = np.ones(X_v.shape[1])
for i in range(0, X_v.shape[1]):
    mean[i] = np.mean(X_v.transpose()[i])
    std[i] = np.std(X_v.transpose()[i])
    for j in range(0, X_v.shape[0]):
        X_v[j][i] = (X_v[j][i] - mean[i])/std[i]
```

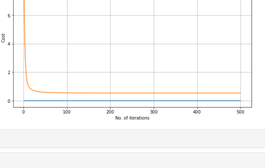
In [34]:
```
theta, cost = linear_regression(X_t, Y_t, 0.1, 500)
cost = list(cost)

theta2, cost2 = linear_regression(X_v, Y_v, 0.1, 500)
cost2 = list(cost2)

n_iterations = [x for x in range(1,501)]
n_iterations2 = [x for x in range(1,501)]

plt.plot(n_iterations, cost, label = 'Training')
plt.plot(n_iterations2, cost2, label = 'Test')
plt.legend()
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Gradient Descent Convergence')
```

Out[34]:
```
Text(0.5, 1.0, 'Gradient Descent Convergence')
```



In [ ]:
```
```

In [ ]:
```
```