Linear Regression with one variable

f(x) = wx + b

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
#x_train is size in 1000 sq feet
#y_train is target price in 1000 USD
x_train = np.array([1.0, 2.0])
y_train = np.array([300.0, 500.0])
#printing input variable and target price as f string
print(f"x_train = {x_train}")
print(f"y_train = {y_train}")
```

```
x_train = [1. 2.]
y_train = [300. 500.]
```

In [4]:
```
# m = number of training examples
# Numpy .shape returns a python tuple with an entry for each dimension
print(f"x_train.shape: {x_train.shape}")
```

```
x_train.shape: (2,)
```

In [5]:
```
# x_train.shape[0] is the length of the array and number of examples
m = x_train.shape[0]

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

```
Number of training example is: 2
```

In [6]:
```
#Also len() function can be used

m = len(x_train)
print(f"Number of training example is: {m}")
```

```
Number of training example is: 2
```

In [7]:
```
# i is ith training example
#Python is zeroth index
# So (x^0, y^0) is 1.0, 300.0
# (x^1, y^1) is 2.0, 500.0

#To access a value in a Numpy array, one indexes the array with the desired offset.
#For example the syntax toaccess location zero ofx_train is x_train[0]
i=0
x_i = x_train[i]
y_i = y_train[i]
print(f"(x^{i}), y^({i})) = ({x_i}, {y_i})")
```

```
(x^0), y^(0)) = (1.0, 300.0)
```

In [8]:
```
i=1
x_i = x_train[i]
y_i = y_train[i]
print(f"(x^{i}), y^({i})) = ({x_i}, {y_i})")
```

```
(x^1), y^(1)) = (2.0, 500.0)
```

In [9]:
```
#Plotting the data frame
#The function arguments marker and c show the points as red crosses
# c is color r is 'red'

plt.scatter(x_train,y_train,marker='x',c='r')
# Set the title
plt.title("Housing Prices")
# Set the y-axis label
plt.ylabel('Price (in 1000s of dollars)')
# Set the x-axis label
plt.xlabel('Size (1000 sqft)')
plt.show()
```

In [11]:
```
# f(x) = wx + b represents straight lines
# We start with w= 100 and b= 100

w= 100
b= 100
print(f"w: {w}")
print(f"b: {b}")
```

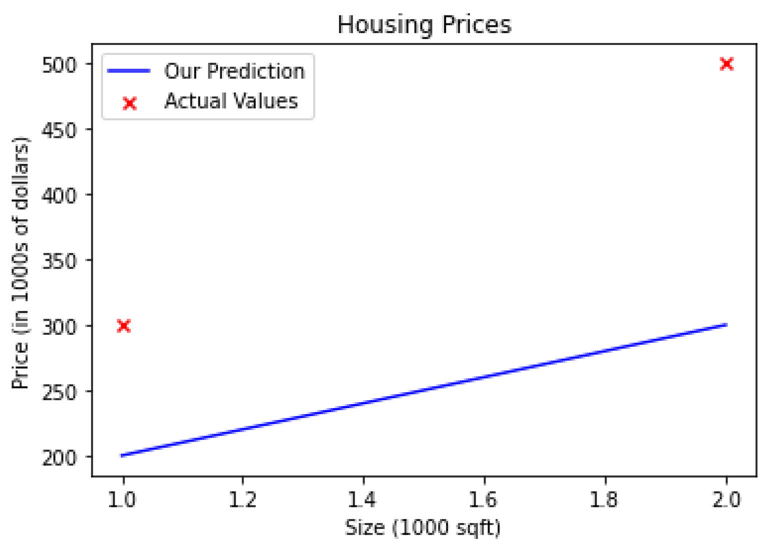
```
w: 100
b: 100
```

In [ ]:
```
# Computing f(x) for two data points
# For x(0), f_wb = w * x[0] + b
# For x(1), f_wb = w * x[1] + b

# For a large number of data points, this can get unwieldy and repetitive.
# So instead, you can calculate thefunction output in aforloop as shown in thecompute_model_output function below

# Note: The argument description (ndarray (m,)) describes a Numpy n-dimensional arrayof shape (m,).
#(scalar)describes an argument without dimensions, just a magnitude.
# Note : np.zero(n) will return a one-dimensional numpy array with entries
```

In [18]:
```
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
```

In [ ]:
```
# We will call the compute_model_output function and plot the output
```

In [19]:
```
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("Housing Prices")

# Set the y-axis label
plt.ylabel('Price (in 1000s of dollars)')

# Set the x-axis label
plt.xlabel('Size (1000 sqft)')
plt.legend()
plt.show()
```



In [ ]:
```
# We can see setting w = 100, b = 100 does not result in a line that fits our data
```

In [20]:
```
w = 200
b = 100
x_i = 1.2
cost_1200sqft = w * x_i + b
print(f"${cost_1200sqft:.0f}thousand dollars")
```

```
$340thousand dollars
```

In [ ]:
```
#Linear regression builds a model which establishes a relationship between features and targets

#In the example above, the feature was house size and the target was house price

#for simple linear regression, the model has two parameters and whose values are 'fit' usingtraining data

#once a model's parameters have been determined, the model can be used to make predictions onnovel data.
```