coursera

Tools

You will utilize the function developed in previous labs as well as matplotlib and NumPy.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from lab_utils_multi import zscore_normalize_features, run_gradient_descent_feng
np.set_printoptions(precision=2)  # reduced display precision on numpy arrays
```

Feature Engineering and Polynomial Regression Overview

Out of the box, linear regression provides a means of building models of the form:

$$f_{\mathbf{w},b} = w_0x_0 + w_1x_1 + ... + w_{n-1}x_{n-1} + b \tag{1}$$

What if your features/data are non-linear or are combinations of features? For example, Housing prices do not tend to be linear with living area but penalize very small or very large houses resulting in the curves shown in the graphic above. How can we use the machinery of linear regression to fit this curve? Recall, the 'machinery' we have is the ability to modify the parameters $\mathbf{w}$, $\mathbf{b}$ in (1) to 'fit' the equation to the training data. However, no amount of adjusting of $\mathbf{w}$,$\mathbf{b}$ in (1) will achieve a fit to a non-linear curve.

Polynomial Features

Above we were considering a scenario where the data was non-linear. Let's try using what we know so far to fit a non-linear curve. We'll start with a simple quadratic: $y = 1 + x^2$

You're familiar with all the routines we're using. They are available in the lab*utils.py file for review. We'll use [`np.c..]`* which is a NumPy routine to concatenate along the column boundary.

In [ ]:
```
# create target data
x = np.arange(0, 20, 1)
y = 1 + x**2
X = x.reshape(-1, 1)
```