List of Files for this assignment:

ex2.py - Python script that steps you through the exercise
ex2_reg.py - Python script for the later parts of the exercise
ex2data1.txt - Training set for the first half of the exercise
ex2data2.txt - Training set for the second half of the exercise
[o] ex2modules.py - module file that includes functions to compute the sigmoid function, compute the cost of logistic regression, run gradient descent, make predictions, and map features
[o] indicates files you will need to complete

Section 1: Logistic regression without Regularization

In this part, you will implement some code for performing training with logistic regression without using regularization.

Part 1: Plotting the Data

The loading of training data (assigned to variables X and y in the code) has already been done for you as follows:

```
x, y = np.loadtxt('ex1data1.txt', delimiter=',', unpack=True)
x = np.matrix(x).T
y = np.matrix(y).T
```

Training data can be visualized within the rectangular 2D plot space. The code to do this has also been written down for you.

Part 2: Computing Initial Cost and Gradient

The backbone code for computing initial cost and gradient is already written down for you. For this part, all you need to do is to implement logRegCost() and logRegGrad()

Part 3: Optimizing using Scipy.optimize.minimize

In this part, you will use a built-in function (scipy.optimize.minimize()) to find the optimal parameters theta. minimize() speeds up optimizing theta parameters of your function to compute cost. It can be thought of as an alternative to gradientDescent() function. After optimization, theta parameters will be used for visualizing the decision boundary on the training data.

Part 4: Prediction and Evaluation

Now that you have finished training, you would check the prediction on a given test data. Then, you would compute the accuracy of training on our training set. The backbone code for doing this has been already written for you in ex2.py. Your job is to implement sigmoid() function and predict() function in ex2modules.py.

Section 2: Logistic regression with Regularization

In this section of the assignment, you will implement some code for performing training with logistic regression WITH using regularization. The base code for this is specified in ex2_reg.py.

Part 5: Adding Polynomial Features and Computing Initial Cost and Gradient

In this part, you are given a dataset with data points that are not linearly separable. However, logistic regression can still be used for classifying the data points. This is possible by adding polynomial features. The code to do this is already done for you. After that, you compute the initial cost and gradient for regularized logistic regression. The backbone code to do this is already written down for you. Note that this is a logistic regression problem with regularization parameter. Looking back at your implemenations of logRegGrad() and logRegCost() for Section 1 of this assignment, if the functions already support regularization parameters, no further modification is needed. However, if the functions only support cases with no regularization parameters, you should better change that to a more general implementation for logistic regression with regularization parameters.

Part 6: Optimizing using Scipy.optimize.minimize + Prediction and Evaluation

In this part, you will use a built-in function function (scipy.optimize.minimize()) to find the optimal parameters theta. Then, you would compute the accuracy of training on our training set. The backbone code for doing this has been already written for you in ex2_reg.py. Your job is to implement predict() function.