In [20]:
```
# Initialize Otter
import otter
grader = otter.Notebook("lab5.ipynb")
```

Lab 5: Modeling and Estimation

In this lab you will work with the tips dataset in order to:

1. Implement a basic model, define loss functions
2. Minimize loss functions using numeric libraries

Setup

In [21]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
np.random.seed(42)
plt.style.use('fivethirtyeight')
sns.set()
sns.set_context("talk")
```

Loading the Tips Dataset

To begin with, we load the tips dataset from the `seaborn` library. The tips data contains records of tips, total bill, and information about the person who paid the bill.

In [22]:
```
data = sns.load_dataset("tips")

print("Number of Records:", len(data))
data.head()
```

```
Number of Records: 244
```

Out[22]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Question 1: Defining the Model

In this lab we will attempt to model the tip value (in dollars) as a function of the total bill. As a consequence we define the following mathematical model:

$$\texttt{Tip} = \theta^* \times \texttt{TotalBill}$$

This follows the similar intuition that tips are some **unknown** percentage of the total bill. We will then try to estimate the slope of this relationship which corresponds to the percent tip.

Here the parameter θ^* represents the true percent tip that we would like to estimate.

Implement the python function for this model (yes this is very easy):

In [23]:
```
def model(theta, total_bill):
    """
    Takes the parameter theta and the total bill, and returns the computed t

    Parameters
    ----------
    theta: tip percentage
    total_bill: total bill value in dollars
    """
    return theta * total_bill
```

In [24]:
```
grader.check("q1")
```

Out[24]:

q1 passed! ✨

Loss Functions

A loss function is what we use to compare different outcomes $f(\theta)$ given some value of θ. (In lecture, some examples of variable θ were portfolio allocation, amount of goods in a transportation problem, etc.)

Recall that, in the movie recommender system, we minimized *squared error of estimated movie ratings*: i.e.,

$$\min_{U,V} \left\{ \sum_{i=1}^{I} \sum_{m=1}^{M} (r_{im} - \hat{r}_{im})^2 \right\} = \min_{U,V} \left\{ \sum_{i=1}^{I} \sum_{m=1}^{M} (r_{im} - u_i^T v_m)^2 \right\},$$

where U and V jointly are the variables. Note that we compute the discrepancy between estimated ratings ($\hat{r}_{im} = u_i^T v_m$) and observed rating r_{ij} by the sum of squared errors. This is also called the squared-loss.

In this lab we will study the *choice of the squared loss vs. the absolute loss functions* when finding the θ that explains data the *best*. In this tips data, x and y are given, and we want to find the best θ. **Hence, θ is the variable.**

Suppose for a given total bill x, we observe a tip value of y and our model predicts a tip value $\hat{y}$ by:

$$\hat{y} = \theta x$$

then any of the following might be appropriate **loss functions**

1. **Squared Loss** (also known as the L^2 loss pronounced "ell-two"):

$$L\left(y, \hat{y}\right) = \left(y - \hat{y}\right)^2$$

1. **Absolute Loss** (also known as the L^1 loss pronounced "ell-one"):

$$L\left(y, \hat{y}\right) = \left|y - \hat{y}\right|$$

In this lab we will compute two *best* θ's. They are,

1. The *best* θ in **squared loss-sense**
2. The *best* θ in **absolute loss-sense**

Question 2a: Implement the squared loss function

In this question, you are going to define functions for **squared loss** and **absolute loss**.

$$L\left(y, \hat{y}\right) = \left(y - \hat{y}\right)^2$$

Using the comments below, implement the squared loss function. Your answer should not use any loops.

In [25]:
```
def squared_loss(y_obs, y_hat):
    """
    Calculate the squared loss of the observed data and predicted data.

    Parameters
    ------------
    y_obs: an array of observed values
    y_hat: an array of predicted values

    Returns
    ------------
    An array of loss values corresponding to the squared loss for each predi
    """
    return (y_obs - y_hat) ** 2
```

In [26]:
```
grader.check("q2a")
```

Out[26]:

q2a passed! 🎉

Question 2b: Plotting Squared Loss

Suppose you observe a bill of $28 with a tip $3. (Does this tip look reasonable?)

Transform this information in our model, we have a $y = 3.00$ and $x = 28.00$. Now suppose we pick an initial range of θ's (tip percent in this case) for you. Use the `model` and `squared_loss` function defined above to plot the loss for a range of θ values:

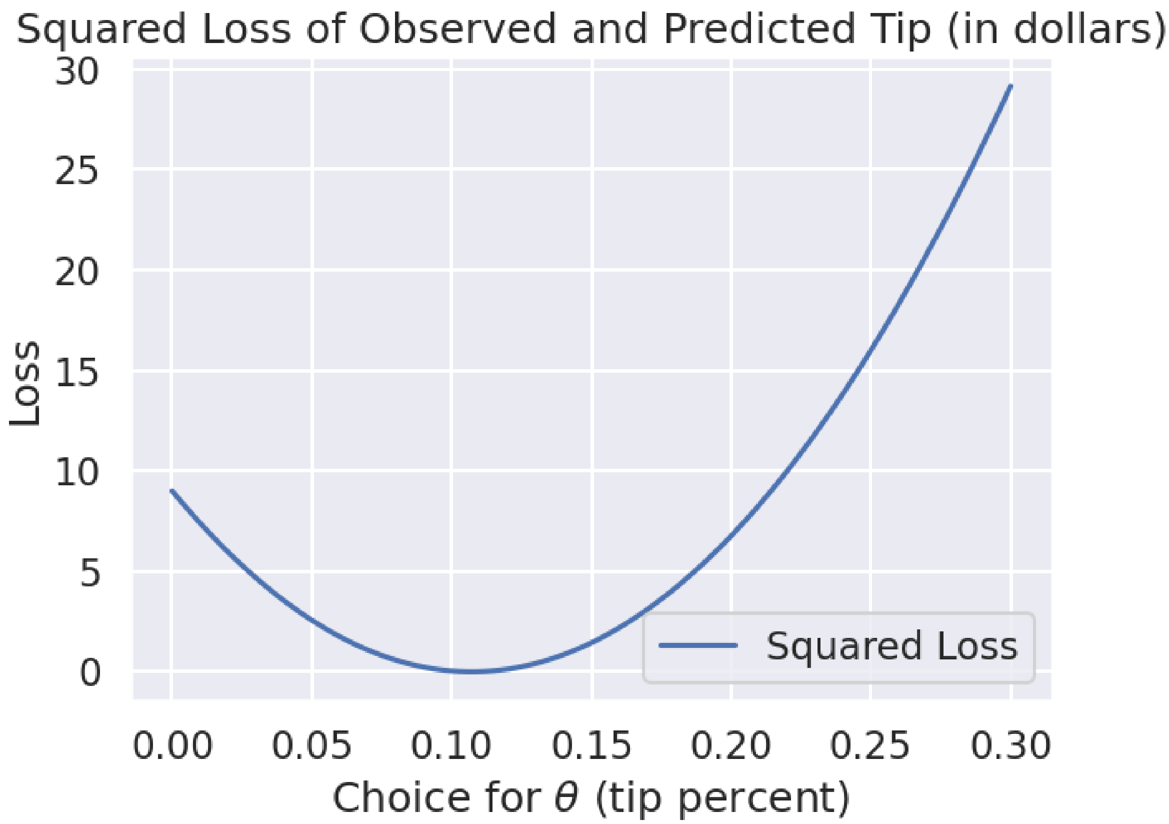
In [27]:
```
y = 3.00
x = 28.00
thetas = np.linspace(0, 0.3, 200) # A range of theta values

## Finish this by replacing 0.0 with the correct calculation
## Hint: You will use squared_loss y, model, theta and x
#loss should be a numpy array where the ith entry corresponds to the loss fo
loss = np.array([ 0.0 for theta in thetas])

loss = np.array([squared_loss(y, model(theta, x)) for theta in thetas])
```

In [28]:
```
grader.check("q2b")
```

Out[28]:

q2b passed! 💯

In [29]:
```
plt.plot(thetas, loss, label="Squared Loss")
plt.title("Squared Loss of Observed and Predicted Tip (in dollars)")
plt.xlabel(r"Choice for $\theta$ (tip percent)")
plt.ylabel(r"Loss")
plt.legend(loc=4)
plt.savefig("squared_loss_my_plot.png",  bbox_inches = 'tight')
```



Question 2c: Implement the absolute loss

$$L(y, \hat{y}) = |y - \hat{y}|$$

In [30]:
```
def abs_loss(y_obs, y_hat):
    """
    Calculate the absolute loss of the observed data and predicted data.

    Parameters
    ------------
    y_obs: an array of observed values
    y_hat: an array of predicted values

    Returns
    ------------
    An array of loss values corresponding to the absolute loss for each prec
    """
    return np.abs(y_obs - y_hat)
```

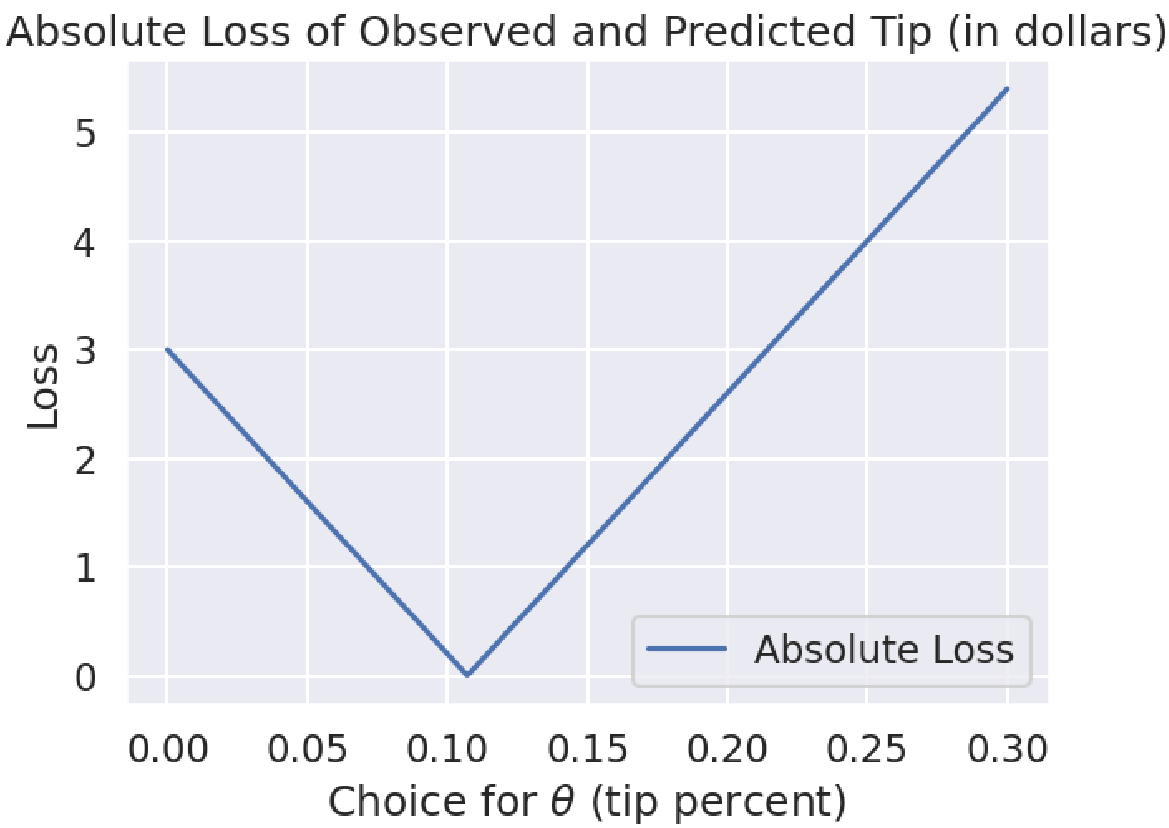
In [31]:
```
grader.check("q2c")
```

Out[31]:

q2c passed! 🌈

Below is the plot of the absolute loss.

In [32]:
```
y = 3.00
x = 28.00
thetas = np.linspace(0, 0.3, 200)

# Code provided for you this time. (you're welcome)
loss = np.array([abs_loss(y, model(theta,x)) for theta in thetas])

plt.plot(thetas, loss, label="Absolute Loss")
plt.title("Absolute Loss of Observed and Predicted Tip (in dollars)")
plt.xlabel(r"Choice for $\theta$ (tip percent)")
plt.ylabel(r"Loss")
plt.legend(loc=4)
plt.savefig("absolute_loss_my_plot.png",  bbox_inches = 'tight')
```



Question 2d: Plotting **Average Loss** for our Data

Remember we define our model to be:

$$\hat{y} = \theta x$$

Now, we can extend the above loss functions to an entire dataset by taking the average. Let the dataset $\mathcal{D}$ be the set of observations:

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

where x_i is the total bill and y_i is the tip dollar amount.

We can define the average loss over the dataset as:

$$L\left(\theta, \mathcal{D}\right) = \frac{1}{n}\sum_{i=1}^{n} L(m_{\theta}(x_i), y_i) = \frac{1}{n}\sum_{i=1}^{n} L(\theta x_i, y_i) = \frac{1}{n}\sum_{i=}^{n}$$

where $m_\theta(x_i) = \theta x_i = \hat{y}_i$ is the model evaluated using the parameters θ on the bill amount x_i.

Complete the following code block to render a plot of the average absolute and squared loss for different values of θ

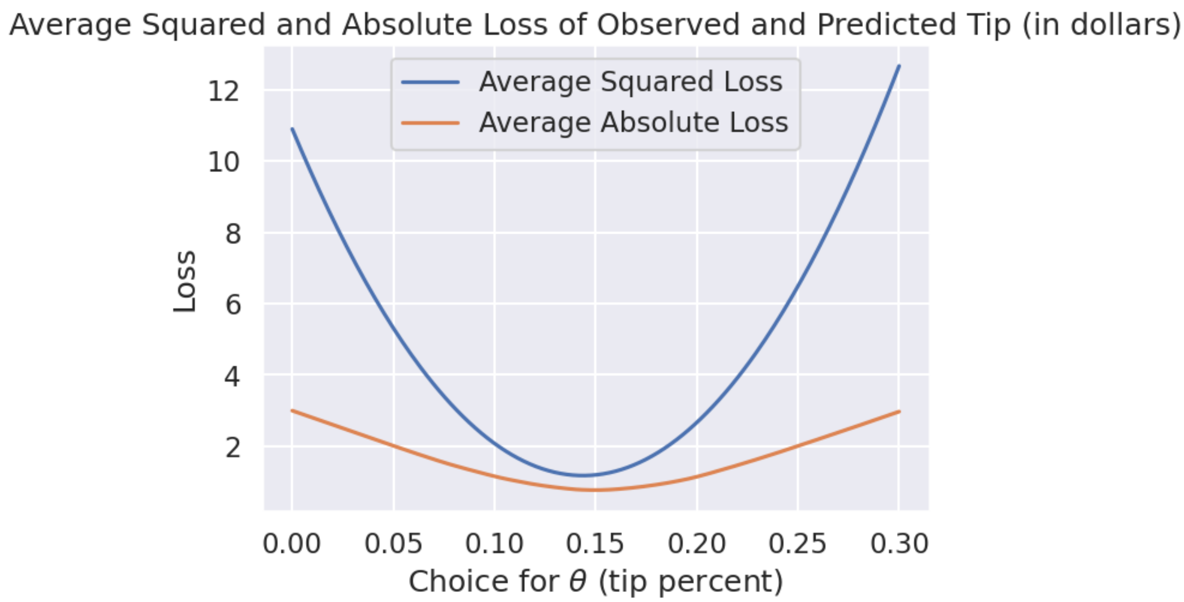
In [33]:
```
thetas = np.linspace(0, 0.3, 200) # A range of theta values
y = data['tip']
x = data['total_bill']

# Replace 0.0 with the correct value computed
# Use the model and loss functions from above

# This time, each loss array should be a numpy array where the ith entry cor
# average loss across all data points for the ith theta

avg_squared_loss = np.array([np.mean(squared_loss(y, model(theta, x))) for t
avg_absolute_loss = np.array([np.mean(abs_loss(y, model(theta, x))) for thet
```

To test your loss calculations, run the cell below. If your code was correct, the following plot should look like:

Average Loss

Note: Your colors might be different.

In [34]:
```
plt.plot(thetas, avg_squared_loss, label = "Average Squared Loss")
plt.plot(thetas, avg_absolute_loss, label = "Average Absolute Loss")
plt.title("Average Squared and Absolute Loss of Observed and Predicted Tip (
plt.xlabel(r"Choice for $\theta$ (tip percent)")
plt.ylabel(r"Loss")
plt.legend()
plt.savefig("average_loss_my_plot.png",  bbox_inches = 'tight')
```



Based on the plot above, approximately what is the optimal value of theta you would choose for this model?

In [35]:
```
q2d2 = "0.15"
```

Question 3: Minimizing The Loss

In class, we used calculus to make improvements to our solution until convergence; however, there are specialized functions that are specifically designed to compute θ that minimize the loss function. In this lab we will use computational techniques to minimize the loss. Here we will use the `scipy.optimize.minimize` routine to minimize the average loss.

Complete the following python function:

In [36]:
```
from scipy.optimize import minimize

def minimize_average_loss(loss_function, model, x, y):
    """
    Minimize the average loss calculated from using different thetas, and
    find the estimation of theta for the model.

    Parameters
    ------------
    loss_function: A loss function, can be the squared or absolute loss func
    model: A defined model function, here we use the model defined above
    x: the x values (total bills)
    y: the y values (tip amounts)

    Returns
    -----------
    The estimation for theta (tip percent) as a scalar
```

```
    Note we will ignore failed convergence for this lab ...
    """

    ## Notes on the following function call which you need to finish:
    #
    # 0. the ... should be replaced with the average loss evaluated on
    #       the data x, y using the model and appropriate loss function
    # 1. x0 is the initial value for THETA.  Yes, this is confusing
    #       but people who write optimization libraries like to use x
    #       as the variable name to optimize, not theta.

    avg_loss = np.mean(loss_function(y, model(theta=0.0, total_bill=x)))


    return minimize(lambda theta: np.mean(loss_function(y, model(theta, x)))
```

In [37]:
```
grader.check("q3")
```

Out[37]:

q3 passed! ✨

To double-check your work, the cell below will rerun all of the autograder tests.

In [38]:
```
grader.check_all()
```

Out[38]:
```
q1 results: All test cases passed!

q2a results: All test cases passed!

q2b results: All test cases passed!

q2c results: All test cases passed!

q3 results: All test cases passed!
```

Submission

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Select *File > Save and export Notebook as > HTML*.
5. Open in Google Chrome and print to PDF.
6. Submit to Gradescope