Phase 1 Project

Introduction

Congratulations on completing phase 1! In this notebook, you will complete a final project that applies your python and visualization skills to real word business data.

Objectives

In this project, you will...

- Read in data that has been stored as a `json` file.
- Describe how the data is structured.
- Use Python to filter a nested data structure
- Define python functions
- Calculate descriptive statistics
- Visualize data via matplotlib

Task: Compare New York Pizza Restaurants with Above Average and Below Average Ratings

Business Understanding

A client at your analytics firm is considering opening a pizza restaurant in New York City.

They have asked you to develop a business intelligence report to fact check the following claims:

1. Your client wants to ensure they have an above average Yelp rating. They have previously owned restuarants in other cities, where a `3` was the average. They would like to know if that holds true in New York City.
2. Your client has noticed that restaurants on yelp with a high review count seem to be quite successful. They have decided to focus on maximizing their review count which they believe will allow them to have an above average overall review.
3. After looking at a few restaurants on Yelp, your client believes that most above average restaurants have a price point of `$$`. They are considering increasing their prices from `$` to `$$` to match the majority of above average restaurants, and would like you to find the most common price point for above average restaurants in New York City.
4. In terms of location they have been told that above average restaurants are usually further east and below average are usually further west, but that the biggest difference is whether the restaurant is on the north or south side. They would like you to determine if the data supports this claim.
5. They believe that the `10012` zipcode in New York City is the best place to open a restaurant. They wish to open a restaurant in close proximity to other highly rated restaurants, and they believe `10012` has the most in NYC.

The primary purpose of this analysis is *descriptive*, meaning your analysis should report calculated statistics such as `counts` and `mean`, and should be focused on providing a simple, factual, understanding of the data.

Data Understanding

You have been provided a Yelp dataset containing information about restaurants in New York City. The data is named `pizza_businesses.json` and is stored in the current working directory. You will need to load in this dataset, inspect how the data is structured, and use the provided information to fact check your client's claims.

Load the data

A dataset containing information about New York pizza restaurants is stored in this notebook's repository with the name `pizza_businesses.json`.

In the cell below, load the json data into a python dictionary.

In [1]:
```
# Import the json python package
# YOUR CODE HERE
import json
# Load in the data
# YOUR CODE HERE
with open("pizza_businesses.json") as f:
    data = json.load(f)
```

NOTE: The data is represented in the form of a list of dictionnaries where each dictionnary presents information on a restaurant (name, review_count, rating, price, location(contains another dictionnary) , transactions, phone, latitude, longitude)

Describe the data

Now that you've loaded in the dataset, the structure of the data should be inspected.

In the cell below, evaluate

- The datatype of the overall dataset
- The datatype of a single observation
- The number of observations, and then
- Isolate the first observation in the dataset

In [2]:
```
# Replace None with your code!

# Find the datatype for the overall dataset
dataset_type = type(data)
# Isolate the first observation
first_observation = data[0]
# Find the datatype for the first observation
observation_type = type(first_observation)
# How many observations are there
num_observations = len(data)

#Some Conclusions using the normal print function
print("The type of the overall data set is",dataset_type,"\n")
print("The first restauarant in the data set has the following information",first_observation,"\n")
print("The type of each observation in the data set is",observation_type,"\n")
print("There are",num_observations,"restaurants in the data set")
```

```
The type of the overall data set is <class 'list'>

The first restauarant in the data set has the following information {'name': 'Prince Street Pizza', 'review_count': 3976, 'rating': 4.5, 'price': '\\$', 'location': {'a
ddress1': '27 Prince St', 'address2': None, 'address3': '', 'city': 'New York', 'zip_code': '10012', 'country': 'US', 'state': 'NY', 'display_address': ['27 Prince St',
'New York, NY 10012']}, 'transactions': ['delivery', 'pickup'], 'phone': '+12129664100', 'latitude': 40.72308755605564, 'longitude': -73.99453001177575}

The type of each observation in the data set is <class 'dict'>

There are 1000 restaurants in the data set
```

Run this following cell unchanged to print out descriptive information for the dataset!

In [3]:
```
from pprint import pprint

print(f'The dataset is a \033[1m{dataset_type}\033[0m')
print(f'The observations are a \033[1m{observation_type}\033[0m',)
print(f'There are \033[1m{num_observations} observations.\033[0m')
print('\033[1m\nThe first observation:\033[0m')
print('=========================================')
pprint(first_observation)
print('=========================================')
```

```
The dataset is a <class 'list'>
The observations are a <class 'dict'>
There are 1000 observations.

The first observation:
=========================================
{'latitude': 40.72308755605564,
 'location': {'address1': '27 Prince St',
              'address2': None,
              'address3': '',
              'city': 'New York',
              'country': 'US',
              'display_address': ['27 Prince St', 'New York, NY 10012'],
              'state': 'NY',
              'zip_code': '10012'},
 'longitude': -73.99453001177575,
 'name': 'Prince Street Pizza',
 'phone': '+12129664100',
 'price': '\\$',
 'rating': 4.5,
 'review_count': 3976,
 'transactions': ['delivery', 'pickup']}
=========================================
```

Find the possible rating options.

In the cell below, create a variable called `rating_options` that has a set datatype (https://realpython.com/python-sets/), and is a unique collection of the possible ratings a restaurant can recieve.

In [4]:
```
# Create the `rating_options` variable
rating_options = set([])

# Loop over all of the observations in the dataset
for i in range(len(data)):
    # Isolate the rating for the restaurant
    # YOUR CODE HERE
    rating = data[i]['rating']
    # Add the rating to
    # the `rating_options` variable
    # YOUR CODE HERE
    rating_options.add(rating)
rating_options
```

Out[4]:
```
{1.0, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0}
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

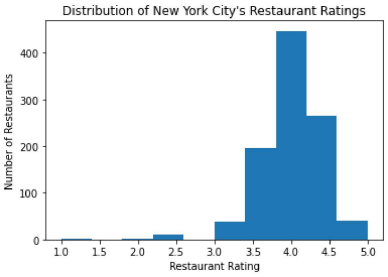
In [5]:
```
assert type(rating_options) == set #True
assert len(rating_options) == 8 #True
assert list(rating_options)[0] != list(rating_options)[1] #True
```

Plot the distribution for ratings

Now that you know what rating options are available, in the cell below plot a histogram showing the distribution of ratings.

In [6]:
```
# Import matplotlib's pyplot module
import matplotlib.pyplot as plt
# Create an empty list.
# We will store all ratings in this list
ratings = []

# Loop over every restaurant in the dataset
# YOUR CODE HERE
for restaurant in data:

    # Isolate the rating
    # YOUR CODE HERE
    rating=restaurant['rating']
    # Append the rating to the `ratings` list
    # YOUR CODE HERE
    ratings.append(rating)

# Create a matplotlib subplot
# YOUR CODE HERE
fig, ax = plt.subplots()
ax.set_facecolor("white")

# Plot a histogram of the ratings list
# YOUR CODE HERE
ax.hist(ratings,linewidth=1);

#Calculating the average rating of restaurants in NYC
sum_of_ratings=0
average_rating_NYC=sum(ratings)/len(ratings)
print("The average rating of a restaurant in NYC is",average_rating_NYC)
ax.set_xlabel("Restaurant Rating")
ax.set_ylabel("Number of Restaurants")
ax.set_title("Distribution of New York City's Restaurant Ratings");
```

```
The average rating of a restaurant in NYC is 4.016
```



Interpret the ratings histogram. How does the visualization relate to your client's claims?

YOUR ANSWER HERE: As shown clearly in the graph (and supported by the calculation above), the average rating of restaurant in NYC is 4.016. That rating is higher than the average rating of the restaurants in other areas where my client operated. In other words, the standards for restaurants in NYC are higher compared to those in other areas. Accordingly, the client has to work on improving the overall quality of service at his restaurants in order to achieve a higher rating and be able to compete in NYC.

Isolate the restaurants with an above average rating

Now that you have an understanding for what is an average rating, next you will isolate restaurants with above average and below average ratings so you can compare them.

In the cell below, filter out all restaurants that do not have a rating of at least `4.5`.

In [7]:
```
# Create an empty list
# You will store restaurants in this list
above_average = []

# Loop over the dataset
# YOUR CODE HERE
for restaurant in data:

    # Isolate the rating
    # YOUR CODE HERE
    rating = restaurant['rating']

    # Check if the rating is at least 4.5
    # YOUR CODE HERE
    if rating >= 4.5:

        # If the rating is at least 4.5
        # Add the restaurant to the list
        # YOUR CODE HERE
        above_average.append(restaurant)
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [8]:
```
assert type(above_average) == list
assert type(above_average[0]) == dict
assert len(above_average) == 306
```

Isolate restaurants with a below average rating

Now repeat the process for below average ratings.

In the cell below, isolate restaurants that have a rating of no more than `3.5`.

In [9]:
```
# Create an empty list
# You will store restaurants in this list
below_average = []

# Loop over the restaurants in the dataset
# YOUR CODE HERE
for restaurant in data:

    # Isolate the rating
    # YOUR CODE HERE
    rating=restaurant['rating']

    # Check if the rating is no more than 3.5
    # YOUR CODE HERE
    if rating <=3.5:
        below_average.append(restaurant)

        # If the rating no more than 3.5
        # Add the restaurant to the list
        # YOUR CODE HERE
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [10]:
```
assert type(below_average) == list
assert type(below_average[0]) == dict
assert len(below_average) == 247
```

Calculate average review counts for both groups

Now that you've isolated above average and below average restaurants, you can calculate the average number of reviews received by both groups.

To do this, you will need to isolate the review counts for both groups, and calculate their average.

The code for isolating the review counts will look very similar to code you have previously written, which is a good sign that a function should be defined!

In the cell below, define a function called `isolate_values` that receives two arguments:

1. A list of dictionaries
2. A string indicating the key that should be isolated for each dictionary

This function should:

- Loop over every dictionary in the inputted list
- Pull out the value assigned to the inputted key
- Append the value to a new list
- Return the new list of values

In [11]:
```
def isolate_values(dictionaries, key):
    # Create an empty list
    # for storing data
    # YOUR CODE HERE
    data_list=[]

    # Loop over every dicionary
    # YOUR CODE HERE
    for data in dictionaries:

        # Isolate the value of the dictionary with the `key`
        # YOUR CODE HERE
        value = data[key]

        # Append the value to the list
        # YOUR CODE HERE
        data_list.append(value)

    # Return the list of values
    # YOUR CODE HERE
    return data_list
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [12]:
```
from types import FunctionType

assert type(isolate_values) == FunctionType
assert type(isolate_values([{'test': 1}], 'test')) == list
assert len(isolate_values([{'test': 1}], 'test')) == 1
assert len(isolate_values(above_average, 'name')) == len(above_average)
assert isolate_values(above_average, 'name')[-1] == above_average[-1]['name']
```

Now use the `isolate_values` function to create a list called `abv_avg_rev_cnts` that contains the review counts for every above average restaurant.

In [13]:
```
# Replace None with your code
abv_avg_rev_cnts = isolate_values(above_average,"review_count")
```

Now use the `isolate_values` function to create a list called `blw_avg_rev_cnts` that contains the review counts for every below average restaurant.

In [14]:
```
# Replace None with your code
blw_avg_rev_cnts = isolate_values(below_average,"review_count")
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

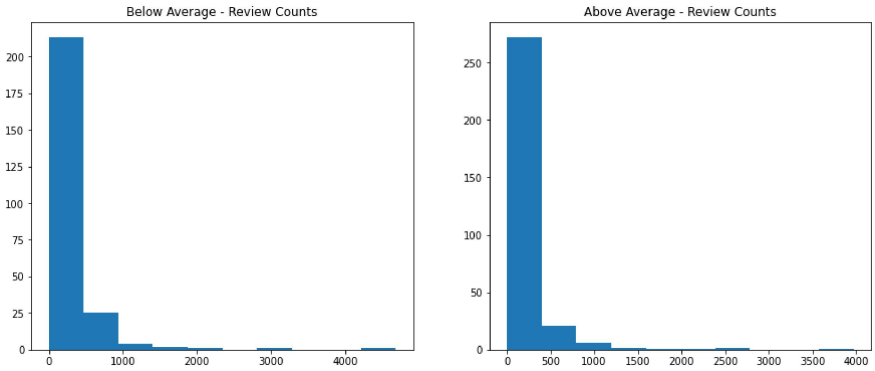
In [15]:
```
assert type(abv_avg_rev_cnts) == list
assert type(abv_avg_rev_cnts[0]) == int
assert type(blw_avg_rev_cnts) == list
assert type(blw_avg_rev_cnts[0]) == int
assert len(abv_avg_rev_cnts) == len(above_average)
assert len(blw_avg_rev_cnts) == len(below_average)
assert abv_avg_rev_cnts[101] == above_average[101]['review_count']
assert blw_avg_rev_cnts[101] == below_average[101]['review_count']
```

Now that you've isolated the review counts for both groups, you can calculate the average review count!

But before you do that, you should inspect the distribution of review counts to make sure `mean` is an appropriate measure of centrality.

In the cell below, we plot a histogram for above average and below average restaurant review counts.

In [16]:
```
# Initialize a matplotlib subplot
# with 1 row and 2 columns
fig, ax = plt.subplots(1,2, figsize=(15,6))

# Plot a histogram of below average review counts
# on the first axis
ax[0].hist(blw_avg_rev_cnts)

# Set the title for the first axis
# to "Below Average - Review Counts"
ax[0].set_title("Below Average - Review Counts")

# Plot a histogram of above average review counts
# on the first axis
ax[1].hist(abv_avg_rev_cnts)

# Set the title for the first axis
# to "Above Average - Review Counts"
ax[1].set_title("Above Average - Review Counts");

for i in range(len(ax)):
    ax[i].set_facecolor("white")
```



Interpret the above visualizations. What statistic is best suited for these data?

YOUR ANSWER HERE: As shown in the visualizations above, the vast majority of the Below Average restaurants have a review count of 500 while the majority of Above Average restaurants have a review count of around 400. Very few restaurants have different numbers of review counts. Since the distribution can be very-well represented by the most frequent number of review counts, the mode is the best suited statistic for these data.

In the cell below, calculate the average review count for above average and below average restaurants.

In [17]:
```
# Import numpy
# YOUR CODE HERE
import numpy as np

# Replace None with your code
abv_avg_rev_cnt_center = sum(abv_avg_rev_cnts)/len(abv_avg_rev_cnts)
blw_avg_rev_cnt_center = sum(blw_avg_rev_cnts)/len(blw_avg_rev_cnts)

print('Above average review count:', abv_avg_rev_cnt_center)
print('Below average review count:', blw_avg_rev_cnt_center)
```

```
Above average review count: 189.3856209150327
Below average review count: 285.99595141700405
```

We can see that there is a big difference between the mode and median values as the distribution is skewed. This proves that the mean is an inappropriate measure of central tendency for representing this distribution.

Interpret the average review count for both groups. How does this relate to your client's claims?

YOUR ANSWER HERE: The calculation above shows that the restaurants with a rating below average had a higher average review count than the restaurants with a rating above average. This contradicts the claim of the client who believes that the restaurants with a higher review count tend to be successful. As shown above, the less successful restaurants which are not rated as highly by customers were reviewed more often. Maybe this means that more customers are encouraged to post reviews for restaurants when they have a negative experience rather than a positive one. Which gives a higher review count for the lower-rated restaurants.

Count the price option frequency

The `price` variable in the dataset is a string of dollar signs indicating how expensive a restaurant's price point is.

In the cell below, write a for loop that counts how frequently a given price point appears for the `above_average` dataset

In [18]:
```
# Create an empty dictionary to store the
# counts for each price point
abv_avg_prices = {}

# Loop over the above average restaurants
# YOUR CODE HERE
for restaurant in above_average:

    # Isolate the price point for the restuarant
    # YOUR CODE HERE
    price_point = restaurant['price']

    # Check if the price has been added to the dictionary
    # YOUR CODE HERE
    if price_point in abv_avg_prices.keys():

            # If the price is already a key in the dictionary
            # Add one to the count for that price point
            # YOUR CODE HERE
            abv_avg_prices[price_point]+=1

    # If the price has not been added to the dictionary
    # Else set the price as the key and the value as the integer `1`
    # YOUR CODE HERE
    else:
        abv_avg_prices[price_point] = 1
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [19]:
```
assert type(abv_avg_prices) == dict
assert len(abv_avg_prices) == 5
assert '\\$\\$\\$\\$' in abv_avg_prices
```

Now reapply the same process, but instead calculate the price point frequencies for the `below_average` dataset.

In [20]:
```
# Create an empty dictionary to store the
# counts for each price point
# YOUR CODE HERE
blw_avg_prices = {}

# Loop over the below average restaurants
# YOUR CODE HERE
for restaurant in below_average:

    # Isolate the price point for the restuarant
    # YOUR CODE HERE
    price_point = restaurant['price']

    # Check if the price has been added to the dictionary
    # YOUR CODE HERE
    if price_point in blw_avg_prices:

            # If the price is already a key in the dictionary
            # Add one to the count for that price point
            # YOUR CODE HERE
            blw_avg_prices[price_point]+=1

    # If the price has not been added to the dictionary
    # Set the price as the key and the value as the integer `1`
    # YOUR CODE HERE
    else:
        blw_avg_prices[price_point] = 1
```

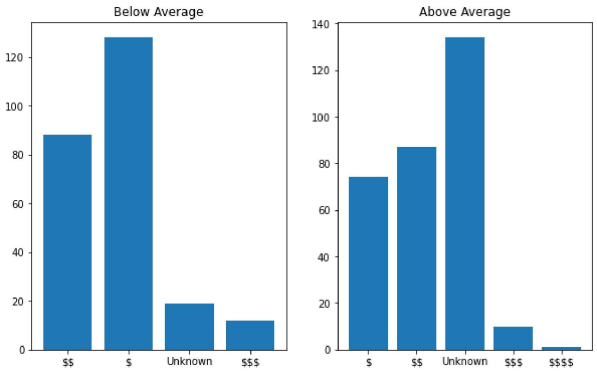
Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [21]:
```
assert type(blw_avg_prices) == dict
assert len(blw_avg_prices) == 4
assert '\\$' in blw_avg_prices
```

Create a bar plot that sets the frequency of each price point as the y axis

In [22]:
```
# Create a matplotlib subplot with 1 row and 2 columns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
# YOUR CODE HERE
fig,axes = plt.subplots(figsize=(10,6),nrows=1,ncols=2)


# Isolate keys of the below average price count dictionary
# This will be the x-axis
# YOUR CODE HERE
x_axis = blw_avg_prices.keys()

# Isolate the values of the below average price count dictionary
# This will be the y-axis
# YOUR CODE HERE
y_axis = blw_avg_prices.values()

# Plot the below average price point counts as a bar plot
# on the first axis
# YOUR CODE HERE
axes[0].bar(x_axis,y_axis)

# Set the title for the first axis
# to the string "Below Average"
# YOUR CODE HERE
axes[0].set_title("Below Average")

# Isolate keys of the above average price count dictionary
# This will be the x-axis
# YOUR CODE HERE
x_axis = abv_avg_prices.keys()

# Isolate the values of the above average price count dictionary
# This will be the y-axis
# YOUR CODE HERE
y_axis = abv_avg_prices.values()

# Plot the above average price counts as a bar plot
# on the second axis
# YOUR CODE HERE
axes[1].bar(x_axis,y_axis)

# Set the title for the second axis to
# the string 'Above Average'
# YOUR CODE HERE
axes[1].set_title("Above Average");

for i in range(len(axes)):
    axes[i].set_facecolor("white")
```



Interpret the above visualization. How does it relate to your client's claims?

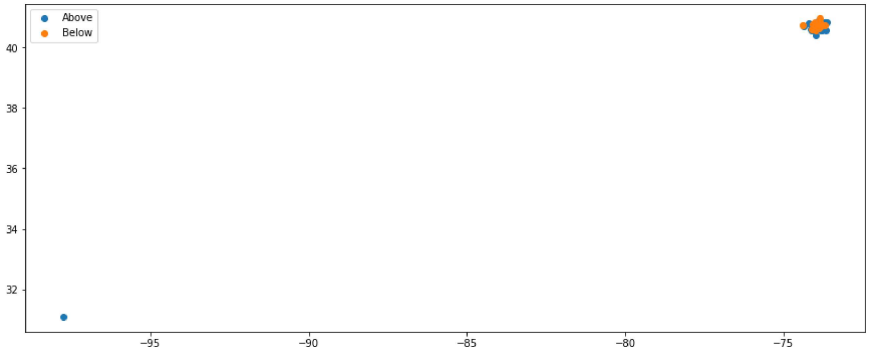
YOUR ANSWER HERE:

The client claims that most above average restaurants in NYC have a $\$ price point which is encouraging him to increase his price range from $ to \$$ to match the competition (with the above average restaurants). However, the visualization "Above Average" shows that the mode of the distribution is the "Unknown". In other words, the majority of restaurants in NYC do not have a defined price point. This does not totally refute the client's claims however it still does not prove him right. These "Unknown" price points might as well be \$$ in reality which would mean the client is right. On the other hand, these price points can also be any of the other possibilities: \$, $\$$ or \$$\$$

Analyze restaurant location

In the cell below, use the `isolate_values` function to isolate `longitude` and `latitude` for above and below average restaurants.

In [23]:
```
# Replace None with your code

# Isolate longitude for above average restaurants
abv_avg_lon = isolate_values(above_average,'longitude')

# Isolate latitude for above average restaurants
abv_avg_lat = isolate_values(above_average,'latitude')

# Isolate longitude for below average restaurants
blw_avg_lon = isolate_values(below_average,'longitude')

# Isolate latitude for below average restaurants
blw_avg_lat = isolate_values(below_average,'latitude')

plt.figure(figsize=(15,6))
plt.scatter(abv_avg_lon, abv_avg_lat, label='Above')
plt.scatter(blw_avg_lon, blw_avg_lat, label='Below')
plt.legend();
```



Remove the outlier

There is one restaurant in the above average dataset with a location dramatically west and south of all other observations. Let's remove that restaurant from the above average dataset and regenerate the scatter plot.

In [24]:
```
# Create an empty list
# that will contain data with
# the outlier removed
no_outliers = [] # YOUR CODE HERE

# Create an empty list to append the outlier
outlier = [] # YOUR CODE HERE

# Loop over every restaurant in the above average dataset
# YOUR CODE HERE
for restaurant in above_average:

    # Isolate the restaurant's longitude
    # YOUR CODE HERE
    rest_lon = restaurant['longitude']

    # Check if the longitude value is greater than the integer -90
    # YOUR CODE HERE
    if rest_lon > -90:
        # Append the restaurant to the no_outliers list
        # YOUR CODE HERE
        no_outliers.append(rest_lon)

    # If longitude is less than -90 it is an outlier
    # and should be appended to the outlier list
    # YOUR CODE HERE
    else:
        outlier.append(rest_lon)
```

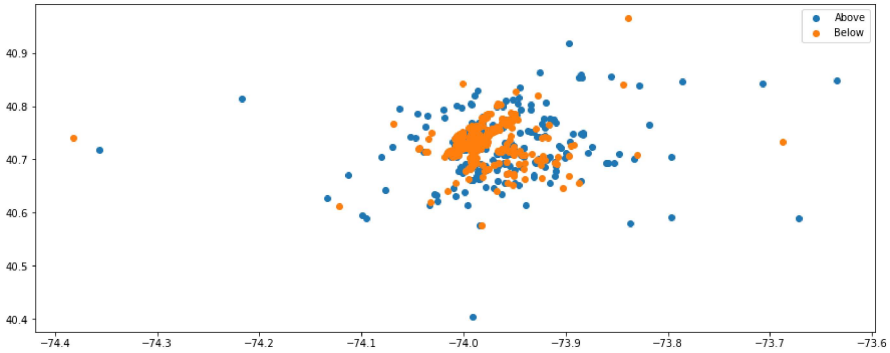
Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [25]:
```
assert type(no_outliers) == list
assert type(outlier) == list
assert len(no_outliers) == len(above_average) - 1
assert len(outlier) == 1
```

Now regenerate the longitude and latitude for above average restaurants using the `no_outliers` dataset, and regenerate the scatter plot!

In [26]:
```
abv_avg_lon = []
abv_avg_lat = []
for restaurant in above_average:
    if restaurant['longitude'] in no_outliers:
        abv_avg_lon.append(restaurant['longitude'])
        abv_avg_lat.append(restaurant['latitude'])
plt.figure(figsize=(15,6))
plt.scatter(abv_avg_lon, abv_avg_lat, label='Above')
plt.scatter(blw_avg_lon, blw_avg_lat, label='Below')
plt.legend();
```

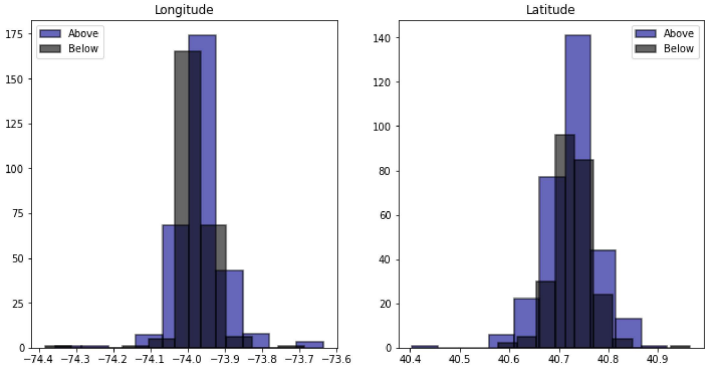


Nice. This is much more interesting.

Plot the distribution of latitude and longitude

To get a better sense about how latitude and longitude are working, in the cell below plot histograms for latitude and longitude.

In [27]:
```
# Create a matplotlib subplot with 1 row and 2 columns
# YOUR CODE HERE
fig,axes = plt.subplots(figsize=(12,6),nrows=1,ncols=2)
# Plot a histogram of above average longitude
# on the first subplot axis. Set alpha to .6
# Set label to the string "Above"
# YOUR CODE HERE
axes[0].hist(abv_avg_lon,alpha=0.6,color="darkblue",edgecolor="black",lw=1.5)

# Plot a histogram of below average longitude
# on the first subplot axis. Set alpha to .6
# Set label to the string "Below"
# YOUR CODE HERE
axes[0].hist(blw_avg_lon,alpha=0.6,color="black",edgecolor="black",lw=1.5)
axes[0].legend(["Above","Below"],loc=2)

# Set the title for the first subplot axis
# to the string "Longitude"
# YOUR CODE HERE
axes[0].set_title("Longitude")

# Plot a histogram of above average latitude
# on the second subplot axis. Set alpha to .6
# Set label to the string "Above"
# YOUR CODE HERE
axes[1].hist(abv_avg_lat,alpha=0.6,color="darkblue",edgecolor="black",lw=1.5)

# Plot a histogram of below average latitude
# on the second subplot axis. Set alpha to .6
# Set label to the string "Below"
# YOUR CODE HERE
axes[1].hist(blw_avg_lat,alpha=0.6,color="black",edgecolor="black",lw=1.5)
axes[1].legend(["Above","Below"])

# Set the title for the second subplot axis
# to the string "Latitude"
# YOUR CODE HERE
axes[1].set_title("Latitude");
```



In [28]:
```
#The mean of the below and above average longitudes distributions:
mean_abv_avg_lon = sum(abv_avg_lon)/len(abv_avg_lon)
mean_blw_avg_lon = sum(blw_avg_lon)/len(blw_avg_lon)

#The mean of the below and above average latitudes distributions:
mean_abv_avg_lat = sum(abv_avg_lat)/len(abv_avg_lat)
mean_blw_avg_lat = sum(blw_avg_lat)/len(blw_avg_lat)
print("The average longitude for the above average restaurants is",mean_abv_avg_lon)
print("The average longitude for the below average restaurants is",mean_blw_avg_lon)
print("We can see that the below average restuarants have a more negative average longitude")
print("That means the below average restaurants are located further to the west\n\n")

print("The average latitude for the above average restaurants is",mean_abv_avg_lat)
print("The average latitude for the below average restaurants is",mean_blw_avg_lat)
print("We can see that the 2 mean values almost match.")
print("Accordingly, there is no big difference in restaurant ratings when comparing those located in the North \nversus those located in the south")
```

```
The average longitude for the above average restaurants is -73.96787728751276
The average longitude for the below average restaurants is -73.97721755216998
We can see that the below average restuarants have a more negative average longitude
That means the below average restaurants are located further to the west


The average latitude for the above average restaurants is 40.72544361701399
The average latitude for the below average restaurants is 40.72914576917316
We can see that the 2 mean values almost match.
Accordingly, there is no big difference in restaurant ratings when comparing those located in the North
versus those located in the south
```

Interpret the above visualization. How does it relate to your client's claims?

YOUR ANSWER HERE: The client claims that the above average restaurants are usually further east while the below average restaurants are usually further west. The client also claims that the biggest difference is between restaurants located on the north side and those located on the south side.

General information: Positive latitude means the area is further north of the equator. Negative latitude means the area is further south of the equator. Positive longitude means the area is further to the west while negative longitude means the area is further to the east.

The first visualization shows that most of the below average restaurants tend to have a more negative Longitude which means that these restaurants are located further to the west. Accordingly, this also means that the above average restaurants tend to have a more positive longitude value which means that they're located further to the east. In statistical terms, the mode of the distribution for the Above Average Longitude is around -73.95 while that of the Below Average Longitude is around -74 ; which is lower meaning the Below Average restaurants are usually further west. Furthermore, the calculation of the mean for the longitude distribution (shown above) proves that the below average restaurants tend to be located further west of the above average restaurants since the mean (of the below average longitude distribution) has a more negative value.

The second visualization shows that the latitude values for the below and above average restaurants almost match. The only difference is that the number of above average restaurants is higher. The calculation of the mean for the above and below average restaurant latitude distribution produces two very similar values. This proves that there is no big difference in ratings between restaurants located in the North and restaurants that are in the South.

In summary, the client is right about the fact that below average restaurants tend to be further west while above average restaurants tend to be further east. However, his claim that the biggest difference lies between restaurants located in the North versus those located in the South is inaccurate and incorrect.

Find the most common zipcode for above average restaurants

In the cell below, loop over the restaurants in the above average dataset and count the frequency of the restaurants zipcode.

In [29]:
```
# Create an empty dictionary
# This dictionary will hold the counts
# for each zipcode
abv_avg_zip_cnts = {} # YOUR CODE HERE

# Loop over the above average dataset
# YOUR CODE HERE
for restaurant in above_average:

    # Isolate the restaurant's zipcode
    # YOUR CODE HERE
    restaurant_zip_code = restaurant['location']['zip_code']

    # Check if the zipcode is a key in the dictionary
    # YOUR CODE HERE
    if restaurant_zip_code in abv_avg_zip_cnts.keys():

        # Add one to the zipcode's value
        # YOUR CODE HERE
        abv_avg_zip_cnts[restaurant_zip_code]+=1

    # If the zipcode is not a key
    # add it to the dictionary with a value of 1
    # YOUR CODE HERE
    else:
         abv_avg_zip_cnts[restaurant_zip_code]=1
```

Run the next cell unchanged to test your work!

> If the below cell runs without throwing an error, your code is likely correct!

In [30]:
```
assert type(abv_avg_zip_cnts) == dict
assert len(abv_avg_zip_cnts) == 104 or len(abv_avg_zip_cnts) == 103
assert '10012' in abv_avg_zip_cnts
```

Now loop over the `abv_avg_zip_cnts` dictionary and find the zipcode with the largest count.

For this question, there are multiple ways to find the solution. Comments have not been provided.

In [31]:
```
# Your code here
zipcodes_list = abv_avg_zip_cnts.keys()
print("The list of zip codes is",list(zipcodes_list),"\n\n")
zipcodes_count = abv_avg_zip_cnts.values()
print("The list of zip codes counts is",list(zipcodes_count),"\n\n")

print("The zipcode with the largest count showed up",max(zipcodes_count),"times.")

for zipcode in abv_avg_zip_cnts:
    if abv_avg_zip_cnts[zipcode] == max(zipcodes_count):
        print("The zipcode with the largest count is",zipcode)
```

```
The list of zip codes is ['10012', '11201', '11211', '11209', '10028', '11222', '10014', '10075', '11205', '10013', '11217', '10023', '10009', '10003', '10016', '1110
2', '10022', '11249', '10010', '10011', '11215', '10006', '11237', '11238', '11218', '10017', '11101', '11226', '10019', '10002', '11231', '10128', '10065', '11104', '1
0001', '10029', '11221', '76542', '11373', '10018', '10027', '11233', '10032', '07302', '10038', '11103', '11385', '11105', '10004', '10021', '10036', '10030', '11418',
'10007', '11207', '10035', '10461', '11220', '07094', '11421', '07020', '07047', '07305', '11050', '11212', '11378', '10454', '11216', '07087', '11219', '07901', '0730
7', '10270', '11375', '10302', '07732', '11213', '10034', '11379', '10301', '11214', '10458', '11432', '07010', '11694', '07042', '10464', '10112', '11206', '11203', '0
7002', '11692', '10025', '11224', '11372', '11234', '11561', '10005', '10705', '10305', '10304', '11225', '07304', '11542']


The list of zip codes counts is [10, 8, 11, 4, 1, 8, 13, 2, 5, 9, 4, 3, 9, 10, 5, 1, 6, 1, 1, 6, 9, 2, 6, 3, 2, 5, 5, 4, 6, 9, 5, 1, 3, 3, 5, 2, 3, 1, 1, 6, 3, 2, 1, 2,
1, 3, 3, 4, 2, 1, 3, 1, 1, 3, 2, 1, 2, 1, 2, 3, 1, 5, 1, 1, 2, 3, 1, 3, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 1, 3, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 3, 1, 4, 1, 1, 1, 1, 1,
1, 1, 1, 1]


The zipcode with the largest count showed up 13 times.
The zipcode with the largest count is 10014
```

Interpret the results. How does the most frequent zipcode relate to your client's claims?

YOUR ANSWER HERE: The client claims that the zipcode 10012 has the highest number of highly rated restaurants which encourages him to open his restaurant in the same area. However, our data proves the client wrong. The data shows that the highest number of above average (highly rated restaurants) are located in the area with zipcode 10014 (13 restaurants). The area with zipcode 10012 has 10 restaurants which is still a large number but it's not the best area as the client claims. There are more highly rated restaurants in the area with zipcode 10014 than in the area with zipcode 10012.

Compile your findings into a report

You have address all of your client's claims! In the cell below, describe the findings of your analysis.

YOUR ANSWER HERE: 1) The average Yelp ratings of the client's restaurants in other areas (rating of 3) is below the average restaurant rating in NYC. Therefore, the client has to consider improving the overall quality of service and customer experience at his restaurants in order to compete with others in NYC.

2) A high review count is not an indication of a high overall restaurant rating in NYC. The average review count of highly rated restaurants is lower than that of the below average restaurants. In other words, in NYC, more people tend to review the below average restaurants than the highly rated ones. We can assume that people are more inclined to share their negative experience at a restaurant than their positive one. Accordingly, maximizing review counts will not, in any way, guarantee a higher rating.

3) Most of the highly rated restaurants in NYC do not have a defined price point (Unknown). Hence, a restaurant's higher price point does not necessarily indicate that the restaurant is highly rated (as the client claims). So, increasing the price point from $ to \$$ will not help the client in competing with the above average restaurants.

4) Most of the highly rated restaurants in NYC are located further to the east compared to the below average restaurants which are further to the west. Apart from that, there is no huge difference (as the client claims) in terms of below and above average restaurants located in the North or South. If the client chooses to open his new restaurant in an area that is further to the west of NYC, then he will be closer to the highly rated restaurants. However, the choice of opening in the North West area or South West area will not have a major effect on the restaurant's chances of competing with the highly rated restaurants.

5) The area with zipcode 10014 has the highest number of highly rated restaurants in NYC and is the best area to open a new restaurant in close proximity to the competition. The area with zipcode 10012 (mentioned by the client) is only the second best area with a total of 10 highly rated restaurants. If the client really wants to have his restaurant in the area with the absolutely highest number of highly rated restaurants, the area with zipcode 10014 is the best choice.

Summary

Well done! As you can see, manipulating nested data stuctures can be a challenging task. Pulling information from json objects requires a thoughtful inspection of how the data is organized, and code that allows you to avoid repetition.