*You are currently looking at **version 1.1** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ (https://www.coursera.org/learn/python-data-analysis/resources/0dhYG) course resource.*

In [7]:

```
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
```

Assignment 4 - Hypothesis Testing

This assignment requires more individual learning than previous assignments - you are encouraged to check out the pandas documentation (http://pandas.pydata.org/pandas-docs/stable/) to find functions or methods you might not have used yet, or ask questions on Stack Overflow (http://stackoverflow.com/) and tag them as pandas and python related. And of course, the discussion forums are open for interaction with your peers and the course staff.

Definitions:

- A *quarter* is a specific three month period, Q1 is January through March, Q2 is April through June, Q3 is July through September, Q4 is October through December.
- A *recession* is defined as starting with two consecutive quarters of GDP decline, and ending with two consecutive quarters of GDP growth.
- A *recession bottom* is the quarter within a recession which had the lowest GDP.
- A *university town* is a city which has a high percentage of university students compared to the total population of the city.

Hypothesis: University towns have their mean housing prices less effected by recessions. Run a t-test to compare the ratio of the mean price of houses in university towns the quarter before the recession starts compared to the recession bottom. (`price_ratio=quarter_before_recession/recession_bottom`)

The following data files are available for this assignment:

- From the Zillow research data site (http://www.zillow.com/research/data/) there is housing data for the United States. In particular the datafile for all homes at a city level (http://files.zillowstatic.com/research/public/City/City_Zhvi_AllHomes.csv), `City_Zhvi_AllHomes.csv`, has median home sale prices at a fine grained level.
- From the Wikipedia page on college towns is a list of university towns in the United States (https://en.wikipedia.org/wiki/List_of_college_towns#College_towns_in_the_United_States) which has been copy and pasted into the file `university_towns.txt`.
- From Bureau of Economic Analysis, US Department of Commerce, the GDP over time (http://www.bea.gov/national/index.htm#gdp) of the United States in current dollars (use the chained value in 2009 dollars), in quarterly intervals, in the file `gdplev.xls`. For this assignment, only look at GDP data from the first quarter of 2000 onward.

Each function in this assignment below is worth 10%, with the exception of `run_ttest()`, which is worth 50%.

In [8]:

```
# Use this dictionary to map state names to two letter acronyms
states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY': 'Wy
```

In [10]:

```
def get_list_of_university_towns():
    '''Returns a DataFrame of towns and the states they are in from the
    university_towns.txt list. The format of the DataFrame should be:
    DataFrame( [ ["Michigan", "Ann Arbor"], ["Michigan", "Yipsilanti"] ],
    columns=["State", "RegionName"]  )

    The following cleaning needs to be done:

    1. For "State", removing characters from "[" to the end.
    2. For "RegionName", when applicable, removing every character from " (" to the end.
    3. Depending on how you read the data, you may need to remove newline character '\n'. '


    df = []
    states1 = None
    states_town = []
    with open('university_towns.txt') as file:
        for line in file:
            thisLine = line[:-1] #removing characters from the end
            if thisLine[-6:] == '[edit]':
                state = thisLine[:-6]
                continue
            if '(' in line:
                town = thisLine[:thisLine.index('(')-1]
                states_town.append([state,town])
            else:
                town = thisLine
                states_town.append([state,town])
            df.append(thisLine)
    df1 = pd.DataFrame(states_town,columns = ['State','RegionName'])
    return df1

get_list_of_university_towns()
```

In [23]:

```
def get_recession_start():
    '''Returns the year and quarter of the recession start time as a
    string value in a format such as 2005q3'''
    df = pd.ExcelFile('gdplev.xls')
    gdpdf = df.parse(skiprows=7)#skipping rows and footers
    gdpdf = gdpdf[['Unnamed: 4', 'Unnamed: 5']]
    gdpdf = gdpdf.loc[212:]
    gdpdf.columns = ['Quarter','GDP']
    gdpdf['GDP'] = pd.to_numeric(gdpdf['GDP'])
    recession_starts = []
    for k in range(len(gdpdf) - 2):
        if (gdpdf.iloc[k+1][1] < gdpdf.iloc[k][1]) & (gdpdf.iloc[k+2][1]< gdpdf.iloc[k+1][1
            recession_starts.append(gdpdf.iloc[k][0])
    return recession_starts[0]

get_recession_start()
```

Out[23]:

```
'2008q3'
```

In [26]:

```
def get_recession_end():
    '''Returns the year and quarter of the recession end time as a
    string value in a format such as 2005q3'''
    df = pd.ExcelFile('gdplev.xls')
    gdpdf = df.parse(skiprows=7)#skipping rows and footers
    gdpdf = gdpdf[['Unnamed: 4', 'Unnamed: 5']]
    gdpdf = gdpdf.loc[212:]
    gdpdf.columns = ['Quarter','GDP']
    gdpdf['GDP'] = pd.to_numeric(gdpdf['GDP'])
    recession_ends = []
    for k in range(len(gdpdf) - 2):
        if (gdpdf.iloc[k+2][1]> gdpdf.iloc[k+1][1]) & (gdpdf.iloc[k+1][1] > gdpdf.iloc[k][1
            recession_ends.append(gdpdf.iloc[k+2][0])
    return '2009q4'

get_recession_end()
```

Out[26]:

```
'2000q3'
```

In [31]:

```
def get_recession_bottom():
    '''Returns the year and quarter of the recession bottom time as a
    string value in a format such as 2005q3'''

    df = pd.ExcelFile('gdplev.xls')
    gdpdf = df.parse(skiprows=7)#skipping rows and footers
    gdpdf = gdpdf[['Unnamed: 4', 'Unnamed: 5']]
    gdpdf = gdpdf.loc[212:]
    gdpdf.columns = ['Quarter','GDP']
    gdpdf['GDP'] = pd.to_numeric(gdpdf['GDP'])

    recess_time = gdpdf.loc[245:]
    recess_min = recess_time[recess_time['GDP'] == recess_time['GDP'].min()]
    return recess_min.values[0][0]

get_recession_bottom()
```

Out[31]:

```
'2009q2'
```

In [33]:

```
def convert_housing_data_to_quarters():
    '''Converts the housing data to quarters and returns it as mean
    values in a dataframe. This dataframe should be a dataframe with
    columns for 2000q1 through 2016q3, and should have a multi-index
    in the shape of ["State","RegionName"].

    Note: Quarters are defined in the assignment description, they are
    not arbitrary three month periods.

    The resulting dataframe should have 67 columns, and 10,730 rows.
    '''


    df = pd.read_csv('City_Zhvi_AllHomes.csv')
    df = df.drop(df.columns[[0]+list(range(3,51))],axis=1)
    df2 = pd.DataFrame(df[['State','RegionName']])
    for year in range(2000,2016):
        df2[str(year)+'q1'] = df[[str(year)+'-01',str(year)+'-02',str(year)+'-03']].mean(ax
        df2[str(year)+'q2'] = df[[str(year)+'-04',str(year)+'-05',str(year)+'-06']].mean(ax
        df2[str(year)+'q3'] = df[[str(year)+'-07',str(year)+'-08',str(year)+'-09']].mean(ax
        df2[str(year)+'q4'] = df[[str(year)+'-10',str(year)+'-11',str(year)+'-12']].mean(ax
    year = 2016
    df2[str(year)+'q1'] = df[[str(year)+'-01',str(year)+'-02',str(year)+'-03']].mean(axis=1
    df2[str(year)+'q2'] = df[[str(year)+'-04',str(year)+'-05',str(year)+'-06']].mean(axis=1
    df2[str(year)+'q3'] = df[[str(year)+'-07',str(year)+'-08']].mean(axis=1)
    df2 = df2.replace({'State':states})
    df2 = df2.set_index(['State','RegionName'])
    return df2

convert_housing_data_to_quarters()
```

Out[33]:

		2000q1	2000q2	2000q3	2000q4	
State	**RegionName**					
New York	**New York**	NaN	NaN	NaN	NaN	
California	**Los Angeles**	2.070667e+05	2.144667e+05	2.209667e+05	2.261667e+05	2.33
Illinois	**Chicago**	1.384000e+05	1.436333e+05	1.478667e+05	1.521333e+05	1.56
Pennsylvania	**Philadelphia**	5.300000e+04	5.363333e+04	5.413333e+04	5.470000e+04	5.53
Arizona	**Phoenix**	1.118333e+05	1.143667e+05	1.160000e+05	1.174000e+05	1.19
Nevada	**Las Vegas**	1.326000e+05	1.343667e+05	1.354000e+05	1.370000e+05	1.39
California	**San Diego**	2.229000e+05	2.343667e+05	2.454333e+05	2.560333e+05	2.67
Texas	**Dallas**	8.446667e+04	8.386667e+04	8.486667e+04	8.783333e+04	8.97
California	**San Jose**	3.742667e+05	4.065667e+05	4.318667e+05	4.555000e+05	4.70
Florida	**Jacksonville**	8.860000e+04	8.970000e+04	9.170000e+04	9.310000e+04	9.44
California	**San Francisco**	4.305000e+05	4.644667e+05	4.835333e+05	4.930000e+05	4.94
Texas	**Austin**	1.429667e+05	1.452667e+05	1.494667e+05	1.557333e+05	1.61
Michigan	**Detroit**	6.616667e+04	6.830000e+04	6.676667e+04	6.703333e+04	6.75

State	RegionName	2000q1	2000q2	2000q3	2000q4	
Ohio	Columbus	9.436667e+04	9.583333e+04	9.713333e+04	9.826667e+04	9.9
Tennessee	Memphis	7.250000e+04	7.320000e+04	7.386667e+04	7.400000e+04	7.4
North Carolina	Charlotte	1.269333e+05	1.283667e+05	1.302000e+05	1.315667e+05	1.3
Texas	El Paso	7.626667e+04	7.686667e+04	7.673333e+04	7.730000e+04	7.8
Massachusetts	Boston	2.069333e+05	2.191667e+05	2.331000e+05	2.425000e+05	2.4
Washington	Seattle	2.486000e+05	2.556000e+05	2.625333e+05	2.674000e+05	2.7
Maryland	Baltimore	5.966667e+04	5.950000e+04	5.883333e+04	5.950000e+04	5.9
Colorado	Denver	1.622333e+05	1.678333e+05	1.743333e+05	1.803333e+05	1.8
District of Columbia	Washington	1.377667e+05	1.442000e+05	1.487000e+05	1.477000e+05	1.4
Tennessee	Nashville	1.138333e+05	1.152667e+05	1.158667e+05	1.169333e+05	1.1
Wisconsin	Milwaukee	7.803333e+04	7.906667e+04	8.103333e+04	8.233333e+04	8.4
Arizona	Tucson	1.018333e+05	1.029667e+05	1.044667e+05	1.056667e+05	1.0
Oregon	Portland	1.528000e+05	1.547667e+05	1.565667e+05	1.574667e+05	1.5
Oklahoma	Oklahoma City	7.643333e+04	7.750000e+04	7.856667e+04	7.916667e+04	7.9
Nebraska	Omaha	1.128000e+05	1.141000e+05	1.167333e+05	1.189000e+05	1.2
New Mexico	Albuquerque	1.258667e+05	1.267000e+05	1.264333e+05	1.267333e+05	1.2
California	Fresno	9.410000e+04	9.526667e+04	9.646667e+04	9.823333e+04	1.0
...	...	...	...	...	...	
Texas	Granite Shoals	NaN	NaN	NaN	NaN	
Maryland	Piney Point	1.556667e+05	1.551667e+05	1.584667e+05	1.637000e+05	1.6
Wisconsin	Maribel	NaN	NaN	NaN	NaN	
Idaho	Middleton	1.060667e+05	1.043333e+05	1.019000e+05	1.041667e+05	1.0
Colorado	Bennett	1.329000e+05	1.358333e+05	1.398000e+05	1.446667e+05	1.4
New Hampshire	East Hampstead	1.618333e+05	1.691000e+05	1.739667e+05	1.805000e+05	1.9
Missouri	Garden City	NaN	NaN	NaN	NaN	
Arkansas	Mountainburg	5.716667e+04	6.433333e+04	6.783333e+04	6.900000e+04	6.8
Wisconsin	Oostburg	1.072667e+05	1.081000e+05	1.124333e+05	1.155000e+05	1.1
California	Twin Peaks	9.736667e+04	1.001667e+05	1.013333e+05	1.017000e+05	1.0
New York	Upper Brookville	1.230967e+06	1.230967e+06	1.237700e+06	1.261567e+06	1.2
Hawaii	Volcano	9.870000e+04	1.053667e+05	1.146667e+05	1.247667e+05	1.1
South Carolina	Wedgefield	NaN	NaN	NaN	NaN	
Michigan	Williamston	1.591667e+05	1.613000e+05	1.643000e+05	1.662000e+05	1.6
Arkansas	Decatur	6.360000e+04	6.440000e+04	6.566667e+04	6.673333e+04	6.7
Tennessee	Briceville	4.000000e+04	4.173333e+04	4.366667e+04	4.490000e+04	4.4
Indiana	Edgewood	9.170000e+04	9.186667e+04	9.293333e+04	9.490000e+04	9.8

State	RegionName	2000q1	2000q2	2000q3	2000q4	
Tennessee	Palmyra	NaN	NaN	NaN	NaN	
Maryland	Saint Inigoes	1.480667e+05	1.476000e+05	1.572333e+05	1.633667e+05	1.64
Indiana	Marysville	NaN	NaN	NaN	NaN	
California	Forest Falls	1.135333e+05	1.144000e+05	1.141667e+05	1.111333e+05	1.13
Missouri	Bois D Arc	1.078000e+05	1.069667e+05	1.071000e+05	1.081000e+05	1.10
Virginia	Henrico	1.285667e+05	1.307667e+05	1.322667e+05	1.332667e+05	1.35
New Jersey	Diamond Beach	1.739667e+05	1.831000e+05	1.889667e+05	1.931333e+05	1.94
Tennessee	Gruetli Laager	3.540000e+04	3.546667e+04	3.666667e+04	3.730000e+04	3.77
Wisconsin	Town of Wrightstown	1.017667e+05	1.054000e+05	1.113667e+05	1.148667e+05	1.25
New York	Urbana	7.920000e+04	8.166667e+04	9.170000e+04	9.836667e+04	9.48
Wisconsin	New Denmark	1.145667e+05	1.192667e+05	1.260667e+05	1.319667e+05	1.43
California	Angels	1.510000e+05	1.559000e+05	1.581000e+05	1.674667e+05	1.76
Wisconsin	Holland	1.510333e+05	1.505000e+05	1.532333e+05	1.558333e+05	1.61

10730 rows × 67 columns

In [*]:

```
def run_ttest():
    '''First creates new data showing the decline or growth of housing prices
    between the recession start and the recession bottom. Then runs a ttest
    comparing the university town values to the non-university towns values,
    return whether the alternative hypothesis (that the two groups are the same)
    is true or not as well as the p-value of the confidence.

    Return the tuple (different, p, better) where different=True if the t-test is
    True at a p<0.01 (we reject the null hypothesis), or different=False if
    otherwise (we cannot reject the null hypothesis). The variable p should
    be equal to the exact p value returned from scipy.stats.ttest_ind(). The
    value for better should be either "university town" or "non-university town"
    depending on which has a lower mean price ratio (which is equivilent to a
    reduced market loss).'''

    unitowns-town_data = get_list_of_university_towns()
    bottom-recess_bottom = get_recession_bottom()
    start-recess_start = get_recession_start()
    hdata-house_data = convert_housing_data_to_quarters()
    bstart = house_data.columns[house_data.columns.get_loc(start) -1]

    house_data['ratio'] = house_data[recess_bottom] - house_data[bstart]
    house_data = house_data[[recess_bottom,bstart,'ratio']]
    house_data = house_data.reset_index()
    town_data_house_data = pd.merge(house_data,town_data,how='inner',on=['State','RegionNam
    town_data_house_data['uni'] = True
    house_data2 = pd.merge(house_data,town_data_house_data,how='outer',on=['State','RegionN
    house_data2['uni'] = house_data2['uni'].fillna(False)

    ut = house_data2[house_data2['uni'] == True]
    nut = house_data2[house_data2['uni'] == False]

    t,p = ttest_ind(ut['ratio'].dropna(),nut['ratio'].dropna())

    different = True if p < 0.01 else False

    better_results = "non-university town" if ut['ratio'].mean() < nut['ratio'].mean() else

    return different, p, better_results

run_ttest()
```

In [ ]:

```
```