*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](#) course resource.*

In [116]:
```python
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](#) to find functions or methods you might not have used yet, or ask questions on [Stack Overflow](#) 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](#) there is housing data for the United States. In particular the datafile for [all homes at a city level](#), `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](#) 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](#) 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 [115]:
```python
# Use this dictionary to map state names to two letter acronyms
states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY': 'Wyoming',
'NA': 'National', 'AL': 'Alabama', 'MD': 'Maryland', 'AK': 'Alaska', 'UT': 'Utah', 'OR': 'Oregon',
'MT': 'Montana', 'IL': 'Illinois', 'TN': 'Tennessee', 'DC': 'District of Columbia', 'VT': 'Vermont',
'ID': 'Idaho', 'AR': 'Arkansas', 'ME': 'Maine', 'WA': 'Washington', 'HI': 'Hawaii', 'WI': 'Wisconsi
n', 'MI': 'Michigan', 'IN': 'Indiana', 'NJ': 'New Jersey', 'AZ': 'Arizona', 'GU': 'Guam', 'MS': 'Mis
sissippi', 'PR': 'Puerto Rico', 'NC': 'North Carolina', 'TX': 'Texas', 'SD': 'South Dakota', 'MP':
'Northern Mariana Islands', 'IA': 'Iowa', 'MO': 'Missouri', 'CT': 'Connecticut', 'WV': 'West Virgini
a', 'SC': 'South Carolina', 'LA': 'Louisiana', 'KS': 'Kansas', 'NY': 'New York', 'NE': 'Nebraska',
'OK': 'Oklahoma', 'FL': 'Florida', 'CA': 'California', 'CO': 'Colorado', 'PA': 'Pennsylvania', 'DE':
'Delaware', 'NM': 'New Mexico', 'RI': 'Rhode Island', 'MN': 'Minnesota', 'VI': 'Virgin Islands', 'N
H': 'New Hampshire', 'MA': 'Massachusetts', 'GA': 'Georgia', 'ND': 'North Dakota', 'VA': 'Virginia'}
```

In [117]:
```python
import pandas as pd
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'. '''
    #uni=pd.read_csv('university_towns.txt', sep='delimiter', header=None)
    #uni=pd.read_csv('university_towns.txt',error_bad_lines=False)
    statentowns=[]
    state=None
    with open('university_towns.txt') as Towndata:
        for line in Towndata:
            presentLine=line[:-1] #slicing the last ch of seq, which is newline
            if presentLine[-6:]=='[edit]': #if last 6 digits  be [edit]
                state=presentLine[:-6]#state is the part before [edit]
                #print(state)
                continue
            if ')' in line :
                townname=presentLine[:presentLine.index('(')-1] #townname is the part before (...)
                #print(townname)
                statentowns.append([state,townname])
            else:
                townname=presentLine.rstrip()
                statentowns.append([state,townname])
    uni=pd.DataFrame(statentowns, columns=["State", "RegionName"])
    return uni
get_list_of_university_towns()
```

Out[117]:

	State	RegionName
0	Alabama	Auburn
1	Alabama	Florence
2	Alabama	Jacksonville
3	Alabama	Livingston
4	Alabama	Montevallo
5	Alabama	Troy
6	Alabama	Tuscaloosa
7	Alabama	Tuskegee
8	Alaska	Fairbanks
9	Arizona	Flagstaff
10	Arizona	Tempe
11	Arizona	Tucson
12	Arkansas	Arkadelphia
13	Arkansas	Conway
14	Arkansas	Fayetteville
15	Arkansas	Jonesboro
16	Arkansas	Magnolia
17	Arkansas	Monticello
18	Arkansas	Russellville
19	Arkansas	Searcy
20	California	Angwin
21	California	Arcata
22	California	Berkeley
23	California	Chico
24	California	Claremont
25	California	Cotati
26	California	Davis
27	California	Irvine
28	California	Isla Vista
29	California	University Park, Los Angeles
...	...	...
487	Virginia	Wise
488	Virginia	Chesapeake
489	Washington	Bellingham
490	Washington	Cheney
491	Washington	Ellensburg
492	Washington	Pullman
493	Washington	University District, Seattle
494	West Virginia	Athens
495	West Virginia	Buckhannon
496	West Virginia	Fairmont
497	West Virginia	Glenville
498	West Virginia	Huntington
499	West Virginia	Montgomery
500	West Virginia	Morgantown
501	West Virginia	Shepherdstown
502	West Virginia	West Liberty
503	Wisconsin	Appleton
504	Wisconsin	Eau Claire
505	Wisconsin	Green Bay
506	Wisconsin	La Crosse
507	Wisconsin	Madison
508	Wisconsin	Menomonie
509	Wisconsin	Milwaukee
510	Wisconsin	Oshkosh
511	Wisconsin	Platteville
512	Wisconsin	River Falls
513	Wisconsin	Stevens Point
514	Wisconsin	Waukesha
515	Wisconsin	Whitewater
516	Wyoming	Laramie

517 rows × 2 columns

In [118]:
```python
import pandas as pd
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.read_excel('gdplev.xls',skiprows=218)
    df=df.iloc[:,4:6]
    df.columns=['Quarter','GDP']
    for i in  range(2, len(df)):
        if (df.iloc[i-2,1]>df.iloc[i-1,1])&(df.iloc[i-1,1]>df.iloc[i,1]):
            print('gotcha')
            return df.iloc[i-2,0]
    #return df2
get_recession_start()
```

gotcha

Out[118]: '2008q3'

In [119]:
```python
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.read_excel('gdplev.xls',skiprows=218)
    df=df.iloc[:,4:6]
    df.columns=['Quarter','GDP']
    for i in  range(2, len(df)):
        if (df.iloc[i+2,1]>df.iloc[i+1,1])&(df.iloc[i+1,1]>df.iloc[i,1])&(df.iloc[i,1]<df.iloc[i-1,1
])&(df.iloc[i-1,1]<df.iloc[i-2,1]):
            print('gotcha')
            return df.iloc[i+2,0]
get_recession_end()
```

gotcha

Out[119]: '2009q4'

In [120]:
```python
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.read_excel('gdplev.xls',skiprows=218)
    df=df.iloc[:,4:6]
    df.columns=['Quarter','GDP']
    for i in  range(2, len(df)):
        if (df.iloc[i+2,1]>df.iloc[i+1,1])&(df.iloc[i+1,1]>df.iloc[i,1])&(df.iloc[i,1]<df.iloc[i-1,1
])&(df.iloc[i-1,1]<df.iloc[i-2,1]):
            print('gotcha')
            return df.iloc[i,0]
get_recession_bottom()
```

gotcha

Out[120]: '2009q2'

In [121]:
```python
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')
    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(a
xis = 1)
        df2[str(year) + 'q2'] = df[[str(year) + '-04', str(year) + '-05', str(year) + '-06']].mean(a
xis = 1)
        df2[str(year) + 'q3'] = df[[str(year) + '-07', str(year) + '-08', str(year) + '-09']].mean(a
xis = 1)
        df2[str(year) + 'q4'] = df[[str(year) + '-10', str(year) + '-11', str(year) + '-12']].mean(a
xis = 1)
    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['State'] = [states[key] for key in df2['State']]
    df2 = df2.set_index(['State', 'RegionName'])
    ans = pd.DataFrame(df2)

    return ans
convert_housing_data_to_quarters()
```

Out[121]:

State	RegionName	2000q1	2000q2	2000q3	2000q4	2001q1	2001q2	
New York	New York	NaN	NaN	NaN	NaN	NaN	NaN	
California	Los Angeles	2.070667e+05	2.144667e+05	2.209667e+05	2.261667e+05	2.330000e+05	2.391000e+05	2.450
Illinois	Chicago	1.384000e+05	1.436333e+05	1.478667e+05	1.521333e+05	1.569333e+05	1.618000e+05	1.664
Pennsylvania	Philadelphia	5.300000e+04	5.363333e+04	5.413333e+04	5.470000e+04	5.533333e+04	5.553333e+04	5.626
Arizona	Phoenix	1.118333e+05	1.143667e+05	1.160000e+05	1.174000e+05	1.196000e+05	1.215667e+05	1.227
Nevada	Las Vegas	1.326000e+05	1.343667e+05	1.354000e+05	1.370000e+05	1.395333e+05	1.417333e+05	1.433
California	San Diego	2.229000e+05	2.343667e+05	2.454333e+05	2.560333e+05	2.672000e+05	2.762667e+05	2.845
Texas	Dallas	8.446667e+04	8.386667e+04	8.486667e+04	8.783333e+04	8.973333e+04	8.930000e+04	8.906
California	San Jose	3.742667e+05	4.065667e+05	4.318667e+05	4.555000e+05	4.706667e+05	4.702000e+05	4.568
Florida	Jacksonville	8.860000e+04	8.970000e+04	9.170000e+04	9.310000e+04	9.440000e+04	9.560000e+04	9.706
California	San Francisco	4.305000e+05	4.644667e+05	4.835333e+05	4.930000e+05	4.940667e+05	4.961333e+05	5.041
Texas	Austin	1.429667e+05	1.452667e+05	1.494667e+05	1.557333e+05	1.612333e+05	1.607333e+05	1.595
Michigan	Detroit	6.616667e+04	6.830000e+04	6.676667e+04	6.703333e+04	6.750000e+04	6.836667e+04	6.926
Ohio	Columbus	9.436667e+04	9.583333e+04	9.713333e+04	9.826667e+04	9.940000e+04	1.002667e+05	1.010
Tennessee	Memphis	7.250000e+04	7.320000e+04	7.386667e+04	7.400000e+04	7.416667e+04	7.493333e+04	7.550
North Carolina	Charlotte	1.269333e+05	1.283667e+05	1.302000e+05	1.315667e+05	1.329333e+05	1.332000e+05	1.328
Texas	El Paso	7.626667e+04	7.686667e+04	7.673333e+04	7.730000e+04	7.823333e+04	7.830000e+04	7.743
Massachusetts	Boston	2.069333e+05	2.191667e+05	2.331000e+05	2.425000e+05	2.496000e+05	2.570667e+05	2.669
Washington	Seattle	2.486000e+05	2.556000e+05	2.625333e+05	2.674000e+05	2.710000e+05	2.724333e+05	2.741
Maryland	Baltimore	5.966667e+04	5.950000e+04	5.883333e+04	5.950000e+04	5.956667e+04	6.013333e+04	6.210
Colorado	Denver	1.622333e+05	1.678333e+05	1.743333e+05	1.803333e+05	1.865000e+05	1.925333e+05	1.964
District of Columbia	Washington	1.377667e+05	1.442000e+05	1.487000e+05	1.477000e+05	1.497667e+05	1.551333e+05	1.646
Tennessee	Nashville	1.138333e+05	1.152667e+05	1.158667e+05	1.169333e+05	1.180333e+05	1.191667e+05	1.201
Wisconsin	Milwaukee	7.803333e+04	7.906667e+04	8.103333e+04	8.233333e+04	8.403333e+04	8.556667e+04	8.706
Arizona	Tucson	1.018333e+05	1.029667e+05	1.044667e+05	1.056667e+05	1.072000e+05	1.087667e+05	1.105
Oregon	Portland	1.528000e+05	1.547667e+05	1.565667e+05	1.574667e+05	1.599000e+05	1.618000e+05	1.642
Oklahoma	Oklahoma City	7.643333e+04	7.750000e+04	7.856667e+04	7.916667e+04	7.983333e+04	8.040000e+04	8.113
Nebraska	Omaha	1.128000e+05	1.141000e+05	1.167333e+05	1.189000e+05	1.208667e+05	1.197667e+05	1.178
New Mexico	Albuquerque	1.258667e+05	1.267000e+05	1.264333e+05	1.267333e+05	1.271000e+05	1.277333e+05	1.285
California	Fresno	9.410000e+04	9.526667e+04	9.646667e+04	9.823333e+04	1.005667e+05	1.035667e+05	1.072
...	...	...	...	...	...	...	...	
Texas	Granite Shoals	NaN	NaN	NaN	NaN	NaN	NaN	
Maryland	Piney Point	1.556667e+05	1.551667e+05	1.584667e+05	1.637000e+05	1.634000e+05	1.648333e+05	1.647
Wisconsin	Maribel	NaN	NaN	NaN	NaN	NaN	NaN	
Idaho	Middleton	1.060667e+05	1.043333e+05	1.019000e+05	1.041667e+05	1.061667e+05	1.083667e+05	1.110
Colorado	Bennett	1.329000e+05	1.358333e+05	1.398000e+05	1.446667e+05	1.483000e+05	1.521000e+05	1.542
New Hampshire	East Hampstead	1.618333e+05	1.691000e+05	1.739667e+05	1.805000e+05	1.909000e+05	1.950667e+05	1.992
Missouri	Garden City	NaN	NaN	NaN	NaN	NaN	NaN	
Arkansas	Mountainburg	5.716667e+04	6.433333e+04	6.783333e+04	6.900000e+04	6.866667e+04	6.386667e+04	6.376
Wisconsin	Oostburg	1.072667e+05	1.081000e+05	1.124333e+05	1.155000e+05	1.191000e+05	1.204333e+05	1.203
California	Twin Peaks	9.736667e+04	1.001667e+05	1.013333e+05	1.017000e+05	1.040000e+05	1.076667e+05	1.098
New York	Upper Brookville	1.230967e+06	1.230967e+06	1.237700e+06	1.261567e+06	1.295167e+06	1.340033e+06	1.403
Hawaii	Volcano	9.870000e+04	1.053667e+05	1.146667e+05	1.247667e+05	1.181333e+05	1.194000e+05	1.232
South Carolina	Wedgefield	NaN	NaN	NaN	NaN	NaN	NaN	
Michigan	Williamston	1.591667e+05	1.613000e+05	1.643000e+05	1.662000e+05	1.664333e+05	1.686333e+05	1.716
Arkansas	Decatur	6.360000e+04	6.440000e+04	6.566667e+04	6.673333e+04	6.720000e+04	6.770000e+04	6.650
Tennessee	Briceville	4.000000e+04	4.173333e+04	4.366667e+04	4.490000e+04	4.480000e+04	4.530000e+04	4.463
Indiana	Edgewood	9.170000e+04	9.186667e+04	9.293333e+04	9.490000e+04	9.893333e+04	1.000667e+05	1.008
Tennessee	Palmyra	NaN	NaN	NaN	NaN	NaN	NaN	
Maryland	Saint Inigoes	1.480667e+05	1.476000e+05	1.572333e+05	1.633667e+05	1.642333e+05	1.682000e+05	1.665
Indiana	Marysville	NaN	NaN	NaN	NaN	NaN	NaN	
California	Forest Falls	1.135333e+05	1.144000e+05	1.141667e+05	1.111333e+05	1.134333e+05	1.130000e+05	1.130
Missouri	Bois D Arc	1.078000e+05	1.069667e+05	1.071000e+05	1.081000e+05	1.107000e+05	1.136667e+05	1.126
Virginia	Henrico	1.285667e+05	1.307667e+05	1.322667e+05	1.332667e+05	1.352333e+05	1.367333e+05	1.386
New Jersey	Diamond Beach	1.739667e+05	1.831000e+05	1.889667e+05	1.931333e+05	1.944000e+05	2.102667e+05	2.302
Tennessee	Gruetli Laager	3.540000e+04	3.546667e+04	3.666667e+04	3.730000e+04	3.773333e+04	3.790000e+04	3.936
Wisconsin	Town of Wrightstown	1.017667e+05	1.054000e+05	1.113667e+05	1.148667e+05	1.259667e+05	1.299000e+05	1.299
New York	Urbana	7.920000e+04	8.166667e+04	9.170000e+04	9.836667e+04	9.486667e+04	9.853333e+04	1.029
Wisconsin	New Denmark	1.145667e+05	1.192667e+05	1.260667e+05	1.319667e+05	1.438000e+05	1.469667e+05	1.483
California	Angels	1.510000e+05	1.559000e+05	1.581000e+05	1.674667e+05	1.768333e+05	1.837667e+05	1.902
Wisconsin	Holland	1.510333e+05	1.505000e+05	1.532333e+05	1.558333e+05	1.618667e+05	1.657333e+05	1.680

10730 rows × 67 columns

In [130]:
```python
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 = get_list_of_university_towns()
    bottom = get_recession_bottom()
    start = get_recession_start()
    housing = convert_housing_data_to_quarters()
    bstart =  housing.columns[housing.columns.get_loc(start) - 1]

    housing['ratio'] = housing[bstart] - housing[bottom]
    housing = housing[[bottom, bstart, 'ratio']]
    housing = housing.reset_index()

    unitowns_housing = pd.merge(housing,unitowns,how='inner',on=['State','RegionName'])
    unitowns_housing['uni'] = True
    housing2 = pd.merge( housing, unitowns_housing, how='outer', on=['State','RegionName',bottom, bs
tart, 'ratio'])
    housing2['uni'] = housing2['uni'].fillna(False)

    university_town =  housing2[housing2['uni'] == True]
    nuniversity_town =  housing2[housing2['uni'] == False]

    t,p = ttest_ind(university_town['ratio'].dropna(), nuniversity_town['ratio'].dropna())

    different = True if p<0.01 else False
    better = "university town" if university_town['ratio'].mean() < nuniversity_town['ratio'].mean()
else "non-university town"

    return(different, p, better)
```