*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 [1]:
```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 [62]:
```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': 'Wisconsin', 'MI': 'Michigan', 'IN': 'Indiana', 'NJ': 'New Jersey', 'AZ': 'Arizona'
, 'GU': 'Guam', 'MS': 'Mississippi', 'PR': 'Puerto Rico', 'NC': 'North Carolina', 'TX': 'Texas', 'SD': 'South Dakota',
'MP': 'Northern Mariana Islands', 'IA': 'Iowa', 'MO': 'Missouri', 'CT': 'Connecticut', 'WV': 'West Virginia', 'SC': 'S
outh Carolina', 'LA': 'Louisiana', 'KS': 'Kansas', 'NY': 'New York', 'NE': 'Nebraska', 'OK': 'Oklahoma', 'FL': 'Florid
a', 'CA': 'California', 'CO': 'Colorado', 'PA': 'Pennsylvania', 'DE': 'Delaware', 'NM': 'New Mexico', 'RI': 'Rhode Isl
and', 'MN': 'Minnesota', 'VI': 'Virgin Islands', 'NH': 'New Hampshire', 'MA': 'Massachusetts', 'GA': 'Georgia', 'ND':
'North Dakota', 'VA': 'Virginia'}
```

In [3]:
```python
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'. '''



    university_town= open('university_towns.txt', 'r')
    lines = university_town.readlines()
    university_town.close()

    lines= [x.rstrip() for x in lines]
    lst=[]
    state=''
    region=''
    for line in lines:
        if '[edit]' in line:
            state = line.replace('[edit]', "")
        elif'(' in line:
            region = line[:line.index('(')-1]
            lst.append([state, region])
        else:
            region = line
            lst.append([state, region])

    df= pd.DataFrame(lst, columns=['State', 'RegionName'])

    return df
get_list_of_university_towns()
```

Out[3]:

	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 [6]:
```python
def get_recession_start():
    '''Returns the year and quarter of the recession start time as a
    string value in a format such as 2005q3'''
    x = pd.ExcelFile('gdplev.xls')
    gdp = x.parse(skiprows=7)
    gdp = gdp[['Unnamed: 4', 'Unnamed: 5']]
    gdp= gdp.loc[212:]
    gdp.columns = ['Quarter','GDP']

    for i in range(0,len(gdp)):
        if (gdp.iloc[i-2][1]>gdp.iloc[i-1][1]) and (gdp.iloc[i-1][1]> gdp.iloc[i][1]):
            startdate = gdp.iloc[i-3][0]
    return startdate
get_recession_start()
```

Out[6]: '2008q3'

In [41]:
```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'''

    gdp = pd.read_excel('gdplev.xls', skiprows=220, parse_cols='E,G', header=None)
    gdp.columns = ['Quarter', 'GDP']

    #start = get_recession_start()
    #start_index = gdp[gdp['Quarter'] == start].index.tolist()[0]
    #gdp=gdp.iloc[start_index:]

    for i in range(4, len(gdp)):
        if (gdp.loc[i-4, 'GDP'] > gdp.loc[i-3, 'GDP']) \
                and (gdp.loc[i-3, 'GDP'] > gdp.loc[i-2, 'GDP']) \
                and (gdp.loc[i-2, 'GDP'] < gdp.loc[i-1, 'GDP']) \
                and (gdp.loc[i-1, 'GDP'] < gdp.loc[i, 'GDP']):
                result = gdp.loc[i, 'Quarter']

    return result

get_recession_end()
```

Out[41]: '2009q4'

In [44]:
```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'''

    gdp = pd.read_excel('gdplev.xls', skiprows=220, parse_cols='E,G', header=None)
    gdp.columns = ['Quarter', 'GDP']

    for i in range(4, len(gdp)):
        if (gdp.loc[i-4, 'GDP'] > gdp.loc[i-3, 'GDP']) \
                and (gdp.loc[i-3, 'GDP'] > gdp.loc[i-2, 'GDP']) \
                and (gdp.loc[i-2, 'GDP'] < gdp.loc[i-1, 'GDP']) \
                and (gdp.loc[i-1, 'GDP'] < gdp.loc[i, 'GDP']):
            result = gdp.loc[i-2, 'Quarter']

    return result

get_recession_bottom()
```

Out[44]: '2009q2'

In [60]:
```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")
    df['State'].replace(states, inplace= True)
    df= df.set_index(["State","RegionName"])
    df = df.iloc[:,49:250]

    def quarters(col):
        if col.endswith(("01", "02", "03")):
            s = col[:4] + "q1"
        elif col.endswith(("04", "05", "06")):
            s = col[:4] + "q2"
        elif col.endswith(("07", "08", "09")):
            s = col[:4] + "q3"
        else:
            s = col[:4] + "q4"
        return s

    housing = df.groupby(quarters, axis = 1).mean()
    housing = housing.sort_index()

    return housing


convert_housing_data_to_quarters()
```

Out[60]:

		2000q1	2000q2	2000q3	2000q4	2001q1	2001q2	2001q3	2001q4
State	RegionName								
Alabama	Adamsville	69033.333333	69166.666667	69800.000000	71966.666667	73466.666667	74000.000000	73333.333333	73100.00
	Alabaster	122133.333333	123066.666667	123166.666667	123700.000000	123233.333333	125133.333333	127766.666667	127200.0
	Albertville	73966.666667	72600.000000	72833.333333	74200.000000	75900.000000	76000.000000	72066.666667	73566.66
	Arab	83766.666667	81566.666667	81333.333333	82966.666667	84200.000000	84533.333333	81666.666667	83900.00
	Ardmore	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Axis	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Baileyton	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Bay Minette	81700.000000	78533.333333	79133.333333	81300.000000	85700.000000	87266.666667	85900.000000	85000.00
	Bayou La Batre	44066.666667	44500.000000	44266.666667	43666.666667	42500.000000	43333.333333	45433.333333	45400.00
	Bessemer	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Birmingham	54033.333333	54400.000000	54966.666667	56066.666667	56833.333333	57600.000000	58433.333333	58700.00
	Boaz	70866.666667	70266.666667	70300.000000	71466.666667	72833.333333	71900.000000	68733.333333	69833.33
	Brent	92933.333333	94333.333333	96166.666667	98333.333333	96533.333333	98500.000000	99366.666667	104100.0
	Brighton	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Brookwood	92566.666667	95100.000000	98866.666667	99966.666667	101666.666667	103666.666667	101833.333333	99900.00
	Buhl	90800.000000	94600.000000	96500.000000	96566.666667	99266.666667	101966.666667	102000.000000	102733.3
	Calera	108933.333333	110366.666667	108000.000000	107933.333333	109333.333333	112200.000000	114333.333333	112133.3
	Center Point	80966.666667	81233.333333	81500.000000	82233.333333	83366.666667	84333.333333	84466.666667	85333.33
	Centreville	95300.000000	96566.666667	98000.000000	99366.666667	98100.000000	99266.666667	100400.000000	104633.3
	Chalkville	94100.000000	94433.333333	94433.333333	94433.333333	96066.666667	97433.333333	98400.000000	99366.66
	Chancellor	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Chelsea	162066.666667	167033.333333	166900.000000	167400.000000	170633.333333	174300.000000	174900.000000	173300.0
	Chickasaw	51200.000000	53666.666667	54933.333333	56066.666667	55133.333333	53566.666667	53533.333333	53400.00
	Chunchula	80266.666667	81766.666667	82200.000000	83400.000000	85400.000000	86100.000000	86733.333333	86566.66
	Citronelle	64833.333333	66633.333333	68066.666667	67766.666667	67866.666667	70000.000000	71466.666667	70933.33
	Clay	120900.000000	122266.666667	123966.666667	126500.000000	128033.333333	128300.000000	129233.333333	130466.6
	Coden	62600.000000	64800.000000	66866.666667	68566.666667	68933.333333	71300.000000	72733.333333	73600.00
	Coker	118100.000000	120766.666667	118166.666667	115333.333333	114066.666667	115700.000000	118833.333333	122566.6
	Concord	78600.000000	78700.000000	80133.333333	82800.000000	85133.333333	85800.000000	84866.666667	84766.66
	Cottondale	100833.333333	102633.333333	104766.666667	105300.000000	106733.333333	109233.333333	109200.000000	109900.0
...	...	...	...	...	...	...	...	...	...
Wisconsin	Vernon	183200.000000	178200.000000	174300.000000	177466.666667	186233.333333	190500.000000	193100.000000	197466.6
	Vienna	178033.333333	181533.333333	182433.333333	186300.000000	190600.000000	191566.666667	193733.333333	194833.3
	Vinland	119800.000000	126766.666667	134933.333333	137833.333333	144300.000000	148200.000000	147733.333333	151200.0
	Wales	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Waterford	121200.000000	119433.333333	120200.000000	121966.666667	127000.000000	133000.000000	139233.333333	143100.0
	Waukesha	141266.666667	138433.333333	136733.333333	139833.333333	147600.000000	151066.666667	153100.000000	155033.3
	Waunakee	187400.000000	191433.333333	192666.666667	193733.333333	197333.333333	198500.000000	201266.666667	202100.0
	Waupun	84000.000000	84000.000000	84400.000000	84533.333333	85100.000000	86600.000000	87666.666667	87100.00
	Wausau	69566.666667	69466.666667	71033.333333	69833.333333	69866.666667	71366.666667	73100.000000	74033.33
	Wayne	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	West Allis	80933.333333	81933.333333	84066.666667	80666.666667	79100.000000	80333.333333	81433.333333	82400.00
	Weston	80166.666667	82700.000000	84166.666667	85800.000000	87266.666667	88500.000000	90066.666667	91766.66
	Wheatland	129800.000000	132133.333333	134933.333333	137833.333333	142566.666667	143000.000000	143966.666667	142633.3
	Whitelaw	96033.333333	101433.333333	108033.333333	114000.000000	116333.333333	114966.666667	115666.666667	115866.6
	Williams Bay	122900.000000	115300.000000	110766.666667	110700.000000	112600.000000	117100.000000	119533.333333	121400.0
	Wilson	129033.333333	129366.666667	132433.333333	137200.000000	143533.333333	143833.333333	143233.333333	141966.6
	Wilson	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Wind Lake	124366.666667	123666.666667	124133.333333	126066.666667	132133.333333	139466.666667	144200.000000	147333.3
	Wind Point	149233.333333	145266.666667	140866.666667	147866.666667	158666.666667	163966.666667	170833.333333	169566.6
	Wisconsin Dells	100000.000000	103400.000000	105000.000000	105433.333333	105533.333333	107100.000000	109866.666667	111633.3
	Wolf River	95166.666667	99333.333333	103800.000000	107633.333333	111400.000000	115033.333333	116900.000000	118233.3
	Woodruff	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Woodville	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Yorkville	139066.666667	134900.000000	135033.333333	140300.000000	148800.000000	155233.333333	161133.333333	162800.0
Wyoming	Bar Nunn	93033.333333	89666.666667	88900.000000	93500.000000	98833.333333	99900.000000	99100.000000	98333.33
	Burns	101533.333333	104566.666667	108366.666667	113000.000000	115833.333333	117200.000000	117800.000000	117633.3
	Casper	89233.333333	89600.000000	89733.333333	93166.666667	95500.000000	97633.333333	99433.333333	100633.3
	Cheyenne	116866.666667	120033.333333	121533.333333	123633.333333	125533.333333	126300.000000	126466.666667	128133.3
	Evansville	128033.333333	128766.666667	130833.333333	132066.666667	130566.666667	131433.333333	132400.000000	133466.6
	Pine Bluffs	93733.333333	95066.666667	94633.333333	98066.666667	103233.333333	104600.000000	106500.000000	104066.6

10730 rows × 67 columns

In [61]:
```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).'''

    towns = get_list_of_university_towns()
    startdate = get_recession_start()
    bottomdate = get_recession_bottom()
    houses = convert_housing_data_to_quarters()

    houses = houses.reset_index()
    houses['recession_diff'] = houses[startdate] - houses[bottomdate]

    towns_houses = pd.merge(houses, towns, how='inner', on=['State', 'RegionName'])
    towns_houses['ctown'] = True
    houses = pd.merge(houses, towns_houses, how='outer', on = ['State', 'RegionName',
                                                              bottomdate, startdate,
                                                              'recession_diff'])

    houses['ctown'] = houses['ctown'].fillna(False)
    unitowns = houses[houses['ctown'] == True]
    not_unitowns = houses[houses['ctown'] == False]

    t, p = ttest_ind(unitowns['recession_diff'].dropna(), not_unitowns['recession_diff'].dropna())
    different = True if p < 0.01 else False
    betters = "university town" if unitowns['recession_diff'].mean() < not_unitowns['recession_diff'].mean() else "non
-university town"

    return different,p,betters

run_ttest()
```

Out[61]: (True, 0.0043252148535112009, 'university town')

In [ ]: