*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 [4]:

```
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 [5]:

```
# 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 Dak
ota', 'MP': 'Northern Mariana Islands', 'IA': 'Iowa', 'MO': 'Missouri', 'CT': 'Connecticut', 'WV':
'West Virginia', 'SC': 'South Carolina', 'LA': 'Louisiana', 'KS': 'Kansas', 'NY': 'New York', 'NE':
'Nebraska', 'OK': 'Oklahoma', 'FL': 'Florida', 'CA': 'California', 'CO': 'Colorado', 'PA': 'Pennsyl
vania', 'DE': 'Delaware', 'NM': 'New Mexico', 'RI': 'Rhode Island', 'MN': 'Minnesota', 'VI': 'Virgi
n Islands', 'NH': 'New Hampshire', 'MA': 'Massachusetts', 'GA': 'Georgia', 'ND': 'North Dakota', 'V
A': 'Virginia'}
```

In [6]:

```
#load all relevant files
homes=pd.read_csv('City_Zhvi_AllHomes.csv')
gdp=pd.read_excel('gdplev.xls')
```

In [7]:

```
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"]  )'''
    #load data
    university_towns=pd.read_csv('university_towns.txt', header=None, sep='\n')
    #create new column
    university_towns['RegionName']=university_towns[0]
    #rename what is now the first column
    university_towns=university_towns.rename(columns={0:'State'})
    #every row of RegionName that ends in a ']', replace brackets and everything in between with n
othing
    university_towns.loc[university_towns['RegionName'].str.endswith(']'), ['RegionName']]=universi
ty_towns['RegionName'].str.replace("\[.*\]","")
    #every row that ends in a '\n', replace space, parentheses and everything in between with noth
ing
    university_towns['RegionName']=university_towns['RegionName'].str.replace("\n","")
    #every row of State that ends in a ']', replace brackets and everything in between with
nothing
    university_towns.loc[university_towns['State'].str.endswith(']'), ['State']]=university_towns['
State'].str.replace("\[.*\]","")
    #every row of RegionName that has a " (", split on that and take first part
    university_towns['RegionName'] = university_towns['RegionName'].str.split(" \(").str[0]
    #every row of RegionName that has a "," split on that and take first part
    university_towns['RegionName'] = university_towns['RegionName'].str.split("\,").str[0]
    #every row of State that has a value that's not a state name (as seen in dictionary) make it b
lank
    university_towns.loc[~university_towns['State'].isin(states.values()), ['State']]=university_to
wns['State'].str.replace(".*", "")
    #make all those blanks NaN
    university_towns.replace(r'^\s*$', np.nan, regex=True, inplace = True)
    #forward fill to get state names in all rows of State column
    university_towns=university_towns.fillna(method='ffill')
    #take out all rows that don't have a city in the RegionName column
    university_towns=university_towns[university_towns['State']!=university_towns['RegionName']]
    return university_towns
get_list_of_university_towns()
```

Out[7]:

	State	RegionName
1	Alabama	Auburn
2	Alabama	Florence
3	Alabama	Jacksonville
4	Alabama	Livingston
5	Alabama	Montevallo
6	Alabama	Troy
7	Alabama	Tuscaloosa
8	Alabama	Tuskegee
10	Alaska	Fairbanks
12	Arizona	Flagstaff
13	Arizona	Tempe
14	Arizona	Tucson
16	Arkansas	Arkadelphia
17	Arkansas	Conway
18	Arkansas	Fayetteville
19	Arkansas	Jonesboro
20	Arkansas	Magnolia
21	Arkansas	Monticello

	State	RegionName
22	Arkansas	Russellville
23	Arkansas	Searcy
25	California	Angwin
26	California	Arcata
27	California	Berkeley
28	California	Chico
29	California	Claremont
30	California	Cotati
31	California	Davis
32	California	Irvine
33	California	Isla Vista
34	California	University Park
...	...	...
533	Virginia	Wise
534	Virginia	Chesapeake
536	Washington	Bellingham
537	Washington	Cheney
538	Washington	Ellensburg
539	Washington	Pullman
540	Washington	University District
542	West Virginia	Athens
543	West Virginia	Buckhannon
544	West Virginia	Fairmont
545	West Virginia	Glenville
546	West Virginia	Huntington
547	West Virginia	Montgomery
548	West Virginia	Morgantown
549	West Virginia	Shepherdstown
550	West Virginia	West Liberty
552	Wisconsin	Appleton
553	Wisconsin	Eau Claire
554	Wisconsin	Green Bay
555	Wisconsin	La Crosse
556	Wisconsin	Madison
557	Wisconsin	Menomonie
558	Wisconsin	Milwaukee
559	Wisconsin	Oshkosh
560	Wisconsin	Platteville
561	Wisconsin	River Falls
562	Wisconsin	Stevens Point
563	Wisconsin	Waukesha
564	Wisconsin	Whitewater
566	Wyoming	Laramie

517 rows × 2 columns

In [8]:

```
def get_recession_start():
    '''Returns the year and quarter of the recession start time as a
    string value in a format such as 2005q3'''
    gdp=pd.read_excel('gdplev.xls', header=None, skiprows=8)
    #get relevant columns, rows
    gdp=gdp[[4,6]].iloc[212:]
    #rename columns
    gdp=gdp.rename(columns={4:'Quarter', 6:'GDP'})
    #subset DF to rows where less than preceding but larger than following
    recession=gdp[(gdp['GDP']<gdp['GDP'].shift(1)) & (gdp['GDP']>gdp['GDP'].shift(-1))]
    #reset the index of the subset and get value in first row, column
    return recession.reset_index(drop=True).iloc[0, 0]
get_recession_start()
```

Out[8]:

```
'2008q3'
```

In [29]:

```
def get_recession_end():
    gdp=pd.read_excel('gdplev.xls', header=None, skiprows=8)
    #get relevant columns, rows
    gdp=gdp[[4,6]].iloc[212:]
    #rename columns
    gdp=gdp.rename(columns={4:'Quarter', 6:'GDP'})
    #subset gdp to where recession starts
    recession_start=gdp[(gdp['GDP']<gdp['GDP'].shift(1)) & (gdp['GDP']>gdp['GDP'].shift(-1))]
    #get index of quarter in which recession starts
    number=recession_start.index[0]
    #subset gdp again from where recession starts and include rest of data
    recession=gdp.loc[number:]
    #subset on that to where GDP less than previous quarter but it goes up in next two quarters
    recession_end=recession[(recession['GDP'].shift(2)<recession['GDP'].shift(1))].iloc[0,0]
    return recession_end
    #NOTE THE ANSWER THEY WANT IS NOT THE LAST QUARTER OF DOWNWARD BUT THE SECOND QUARTER OF UPWAR
D. YOU WERE TOO LAZY TO CHANGE SINCE YOU PASSED.
get_recession_end()
```

Out[29]:

```
'2009q4'
```

In [30]:

```
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', header=None, skiprows=8)
    #get relevant columns, rows
    gdp=gdp[[4,6]].iloc[212:]
    #rename columns
    gdp=gdp.rename(columns={4:'Quarter', 6:'GDP'})
    recession_start=gdp[(gdp['GDP']<gdp['GDP'].shift(1)) & (gdp['GDP']>gdp['GDP'].shift(-1))]
    #get index of quarter in which recession starts
    start=recession_start.index[0]
    #new dataframe starting from where recession started
    recession_end=gdp.loc[start:]
    #subset the new dataframe so it starts on the last quarter of the recession
    recession_end=recession_end[(recession_end['GDP']<recession_end['GDP'].shift(1)) & (recession_e
nd['GDP']<recession_end['GDP'].shift(-1)) & (recession_end['GDP'].shift(-1)<recession_end['GDP'].sh
ift(-2))]
    #get index of quarter where recession starts
    end=recession_end.index[0]
    #create new dataframe that goes from quarter to where recession starts to where it ends
    recession=gdp.loc[start:end]
    #find row with minimum value in GDP
    bottom=recession[recession['GDP']==recession['GDP'].min()]
    #get the value in Quarter column
    return bottom.iloc[0]['Quarter']
get_recession_bottom()
```

Out[30]:

```
'2009q2'
```

In [31]:

```
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.
    '''
    homes=pd.read_csv('City_Zhvi_AllHomes.csv')
    #the dictionary to use to you can switch two letter state names to full state names
    states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY': 'Wyomin
g', 'NA': 'National', 'AL': 'Alabama', 'MD': 'Maryland', 'AK': 'Alaska', 'UT': 'Utah', 'OR': 'Orego
n', 'MT': 'Montana', 'IL': 'Illinois', 'TN': 'Tennessee', 'DC': 'District of Columbia', 'VT': 'Verm
ont', 'ID': 'Idaho', 'AR': 'Arkansas', 'ME': 'Maine', 'WA': 'Washington', 'HI': 'Hawaii', 'WI': 'Wi
sconsin', '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': 'South Carolina', 'LA': 'Louisiana', 'KS': 'Kansas', 'NY': 'New York', 'NE': 'Nebr
aska', 'OK': 'Oklahoma', 'FL': 'Florida', 'CA': 'California', 'CO': 'Colorado', 'PA': 'Pennsylvania
', 'DE': 'Delaware', 'NM': 'New Mexico', 'RI': 'Rhode Island', 'MN': 'Minnesota', 'VI': 'Virgin Isl
ands', 'NH': 'New Hampshire', 'MA': 'Massachusetts', 'GA': 'Georgia', 'ND': 'North Dakota', 'VA': '
Virginia'}
    #makes list of columns you will want to drop
    columns=['RegionID', 'Metro', 'CountyName', 'SizeRank']
    #drop those columns
    homes.drop(columns, inplace=True, axis=1)
    #drop other columns
    homes.drop(homes.columns[2:47], inplace=True, axis=1)
    #map those two letter state names to full names
    homes['State']=homes['State'].map(states)
    #set the index to RegionName, State
    homes=homes.set_index(['State', 'RegionName'])
    #make columns into datetime
    homes.columns=pd.to_datetime(homes.columns)
    #resample to find the average of all columns within the same quarter (as determined by datetim
e)
    homes=homes.resample('Q',axis=1).mean()
    #rewrite column names that came about in previous step (datetime of endpoint of quarter) so th
at they say the name of the quarter as a whole
    homes = homes.rename(columns=lambda x: str(x.to_period('Q')).lower())
    return homes
convert_housing_data_to_quarters()
```

Out[31]:

		2000q1	2000q2	2000q3	2000q4	2001q1	2001q2	200
State	**RegionName**							
New York	**New York**	NaN	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	2.45
Illinois	**Chicago**	1.384000e+05	1.436333e+05	1.478667e+05	1.521333e+05	1.569333e+05	1.618000e+05	1.66
Pennsylvania	**Philadelphia**	5.300000e+04	5.363333e+04	5.413333e+04	5.470000e+04	5.533333e+04	5.553333e+04	5.62
Arizona	**Phoenix**	1.118333e+05	1.143667e+05	1.160000e+05	1.174000e+05	1.196000e+05	1.215667e+05	1.22
Nevada	**Las Vegas**	1.326000e+05	1.343667e+05	1.354000e+05	1.370000e+05	1.395333e+05	1.417333e+05	1.43
California	**San Diego**	2.229000e+05	2.343667e+05	2.454333e+05	2.560333e+05	2.672000e+05	2.762667e+05	2.84
Texas	**Dallas**	8.446667e+04	8.386667e+04	8.486667e+04	8.783333e+04	8.973333e+04	8.930000e+04	8.90
California	**San Jose**	3.742667e+05	4.065667e+05	4.318667e+05	4.555000e+05	4.706667e+05	4.702000e+05	4.56
Florida	**Jacksonville**	8.860000e+04	8.970000e+04	9.170000e+04	9.310000e+04	9.440000e+04	9.560000e+04	9.70

State	RegionName	2000q1	2000q2	2000q3	2000q4	2001q1	2001q2	200
California	San Francisco	4.305000e+05	4.644667e+05	4.835333e+05	4.930000e+05	4.940667e+05	4.961333e+05	5.04
Texas	Austin	1.429667e+05	1.452667e+05	1.494667e+05	1.557333e+05	1.612333e+05	1.607333e+05	1.59
Michigan	Detroit	6.433333e+04	6.826667e+04	6.726667e+04	6.686667e+04	6.710000e+04	6.820000e+04	6.90
Ohio	Columbus	9.436667e+04	9.583333e+04	9.713333e+04	9.826667e+04	9.940000e+04	1.002667e+05	1.01
Tennessee	Memphis	7.250000e+04	7.320000e+04	7.386667e+04	7.400000e+04	7.416667e+04	7.493333e+04	7.55
North Carolina	Charlotte	1.269333e+05	1.283667e+05	1.302000e+05	1.315667e+05	1.329333e+05	1.332000e+05	1.32
Texas	El Paso	7.626667e+04	7.686667e+04	7.673333e+04	7.730000e+04	7.823333e+04	7.830000e+04	7.74
Massachusetts	Boston	2.069333e+05	2.191667e+05	2.331000e+05	2.425000e+05	2.496000e+05	2.570667e+05	2.66
Washington	Seattle	2.486000e+05	2.556000e+05	2.625333e+05	2.674000e+05	2.710000e+05	2.724333e+05	2.74
Maryland	Baltimore	5.966667e+04	5.950000e+04	5.883333e+04	5.950000e+04	5.956667e+04	6.013333e+04	6.21
Colorado	Denver	1.622333e+05	1.678333e+05	1.743333e+05	1.803333e+05	1.865000e+05	1.925333e+05	1.96
District of Columbia	Washington	1.377667e+05	1.442000e+05	1.487000e+05	1.477000e+05	1.497667e+05	1.551333e+05	1.64
Tennessee	Nashville	1.138333e+05	1.152667e+05	1.158667e+05	1.169333e+05	1.180333e+05	1.191667e+05	1.20
Wisconsin	Milwaukee	7.803333e+04	7.906667e+04	8.103333e+04	8.233333e+04	8.403333e+04	8.556667e+04	8.70
Arizona	Tucson	1.018333e+05	1.029667e+05	1.044667e+05	1.056667e+05	1.072000e+05	1.087667e+05	1.10
Oregon	Portland	1.528000e+05	1.547667e+05	1.565667e+05	1.574667e+05	1.599000e+05	1.618000e+05	1.64
Oklahoma	Oklahoma City	7.643333e+04	7.750000e+04	7.856667e+04	7.916667e+04	7.986667e+04	8.040000e+04	8.11
Nebraska	Omaha	1.128000e+05	1.141000e+05	1.167333e+05	1.189000e+05	1.208667e+05	1.197667e+05	1.17
New Mexico	Albuquerque	1.258667e+05	1.267000e+05	1.264333e+05	1.267333e+05	1.271000e+05	1.277333e+05	1.28
California	Fresno	9.410000e+04	9.526667e+04	9.646667e+04	9.823333e+04	1.005667e+05	1.035667e+05	1.07
...	...	...	...	...	...	...	...	...
Missouri	Garden City	1.224667e+05	1.225000e+05	1.224333e+05	1.224000e+05	1.225000e+05	1.225000e+05	1.22
Wisconsin	Hager City	1.339333e+05	1.332000e+05	1.351000e+05	1.397000e+05	1.422667e+05	1.428333e+05	1.41
Michigan	Lake Isabella	NaN	NaN	8.180000e+04	8.426667e+04	8.730000e+04	8.880000e+04	8.83
Arkansas	Mountainburg	5.716667e+04	6.433333e+04	6.783333e+04	6.900000e+04	6.866667e+04	6.386667e+04	6.37
Wisconsin	Oostburg	1.072667e+05	1.081000e+05	1.124333e+05	1.155000e+05	1.191000e+05	1.204333e+05	1.20
New York	Upper Brookville	1.230967e+06	1.230967e+06	1.237700e+06	1.261567e+06	1.295167e+06	1.340033e+06	1.40
Nebraska	Utica	8.890000e+04	9.193333e+04	9.056667e+04	8.620000e+04	8.126667e+04	7.610000e+04	7.24
Hawaii	Volcano	9.870000e+04	1.053667e+05	1.146667e+05	1.247667e+05	1.181333e+05	1.194000e+05	1.23
South Carolina	Wedgefield	NaN	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	1.71
Arkansas	Decatur	6.360000e+04	6.440000e+04	6.566667e+04	6.673333e+04	6.720000e+04	6.770000e+04	6.65
Indiana	Edgewood	9.170000e+04	9.186667e+04	9.293333e+04	9.490000e+04	9.893333e+04	1.000667e+05	1.00
Tennessee	Palmyra	NaN	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	1.66
Indiana	Marysville	NaN	NaN	NaN	NaN	NaN	NaN	NaN
California	Forest Falls	1.135333e+05	1.144000e+05	1.141667e+05	1.111333e+05	1.134333e+05	1.130333e+05	1.13
Michigan	Greilickville	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Minnesota	McGrath	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Missouri	Bois D Arc	1.078000e+05	1.070000e+05	1.071000e+05	1.081000e+05	1.107000e+05	1.136667e+05	1.12
Virginia	Henrico	1.285667e+05	1.307667e+05	1.322667e+05	1.332667e+05	1.352333e+05	1.367333e+05	1.38
New Jersey	Diamond Beach	1.739667e+05	1.831000e+05	1.889667e+05	1.931333e+05	1.944000e+05	2.102667e+05	2.30

		2000q1	2000q2	2000q3	2000q4	2001q1	2001q2	200
Wisconsin	Lake Ripley	1.457667e+05	1.448333e+05	1.403333e+05	1.369667e+05	1.369667e+05	1.376000e+05	1.4
State	RegionName							
Pennsylvania	Highland Township	1.325000e+05	1.321333e+05	1.340333e+05	1.349333e+05	1.382667e+05	1.432000e+05	1.4
	Palmyra Township	9.046667e+04	9.026667e+04	9.260000e+04	9.453333e+04	9.393333e+04	9.330000e+04	9.17
Wisconsin	Town of Wrightstown	1.017667e+05	1.054000e+05	1.113667e+05	1.148667e+05	1.259667e+05	1.299000e+05	1.29
New York	Urbana	7.920000e+04	8.166667e+04	9.170000e+04	9.836667e+04	9.486667e+04	9.853333e+04	1.02
Wisconsin	New Denmark	1.145667e+05	1.192667e+05	1.260667e+05	1.319667e+05	1.438000e+05	1.469667e+05	1.48
California	Angels	1.510000e+05	1.559000e+05	1.581000e+05	1.674667e+05	1.768333e+05	1.837667e+05	1.90
Wisconsin	Holland	1.510333e+05	1.505000e+05	1.532333e+05	1.558333e+05	1.618667e+05	1.657667e+05	1.68
New Jersey	Lebanon Borough	1.658000e+05	1.698333e+05	1.732667e+05	1.772333e+05	1.803333e+05	1.838000e+05	1.88

13099 rows × 72 columns

In [34]:

```
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).'''
    #load relevant data
    hdf = convert_housing_data_to_quarters()
    rec_start=get_recession_start()
    rec_bottom = get_recession_bottom()
    ul = get_list_of_university_towns()
    #to get quarter before recession started
    gdp=pd.read_excel('gdplev.xls', header=None, skiprows=8)
    #narrow data to relevant stuff and rename columns
    gdp=gdp[[4,6]].iloc[212:]
    gdp=gdp.rename(columns={4:'Quarter', 6:'GDP'})
    #pull out the index from the row before the quarter the recession started
    qrt_bfr_rec_start=gdp.index[gdp['Quarter']==rec_start][0]-1
    #pull out quarter from the row with that index (loc instead of iloc because the index you pull
ed out is more a name than a number)
    qrt_bfr_rec_start=gdp.loc[qrt_bfr_rec_start]['Quarter']

    #create PricingRatio column of quarter before / quarter bottom
    hdf['PricingRatio']=hdf[qrt_bfr_rec_start].div(hdf[rec_bottom])
    #make university towns into a list of tuples
    ut_list=ul.to_records(index=False).tolist()
    #group from gdp data frame that is in university towns
    group1=hdf.loc[hdf.index.isin(ut_list)]
    #group from gdp data frame that is not in university towns
    group2=hdf.loc[~hdf.index.isin(ut_list)]
    #run ttest, omit NaN
    ttest=ttest_ind(group1['PricingRatio'], group2['PricingRatio'], nan_policy='omit')
    different=ttest.pvalue<.01
    p=ttest.pvalue
    if group1['PricingRatio'].mean()>group2['PricingRatio'].mean():
        better='non-university towns'
    else:
        better='university town'
    return (different, p, better)
run_ttest()
```

Out[34]:

```
(False, 0.03117926533616027, 'university town')
```