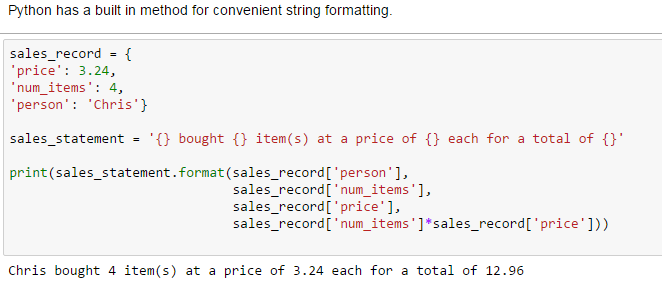
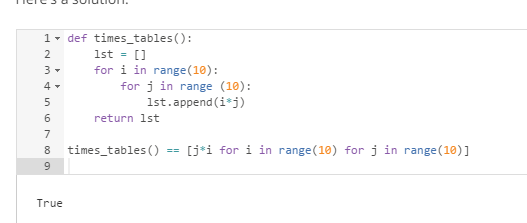
# Grammar

## String formatting



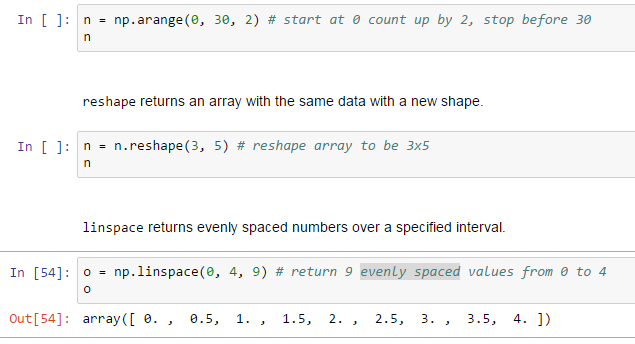
## Convert into list



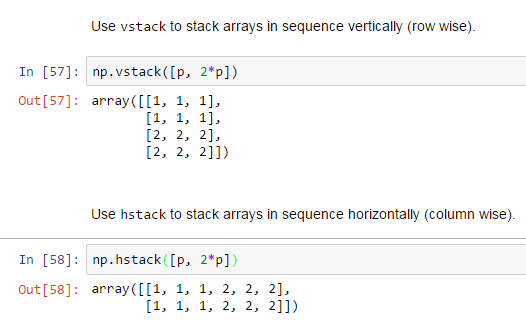
# Numpy

## Numpy basic functions

### Generate arraies:



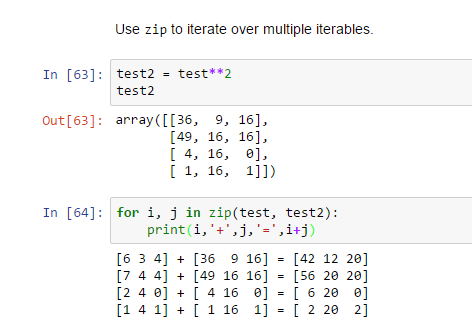
### Mat stack functions:



### Important array feature:

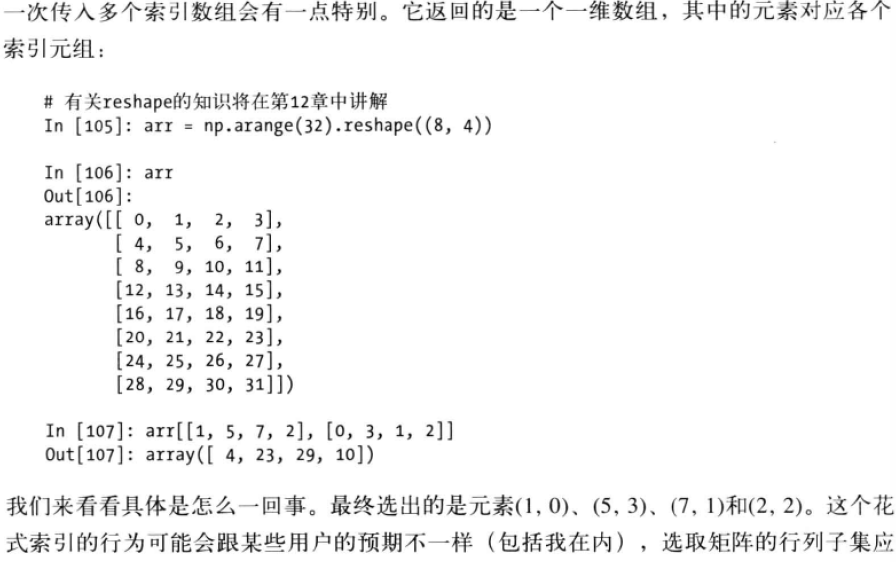


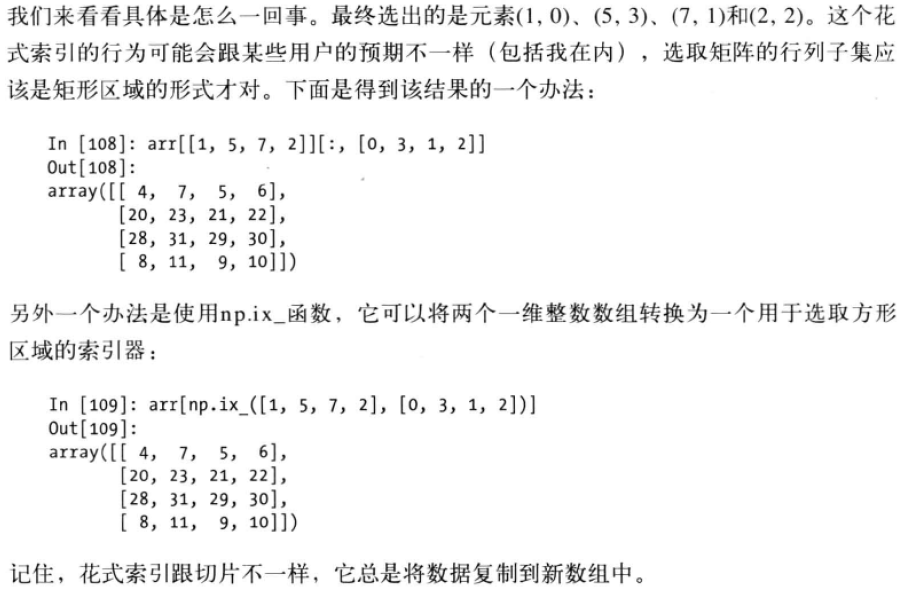
### Zip to iterate multiple iterables:



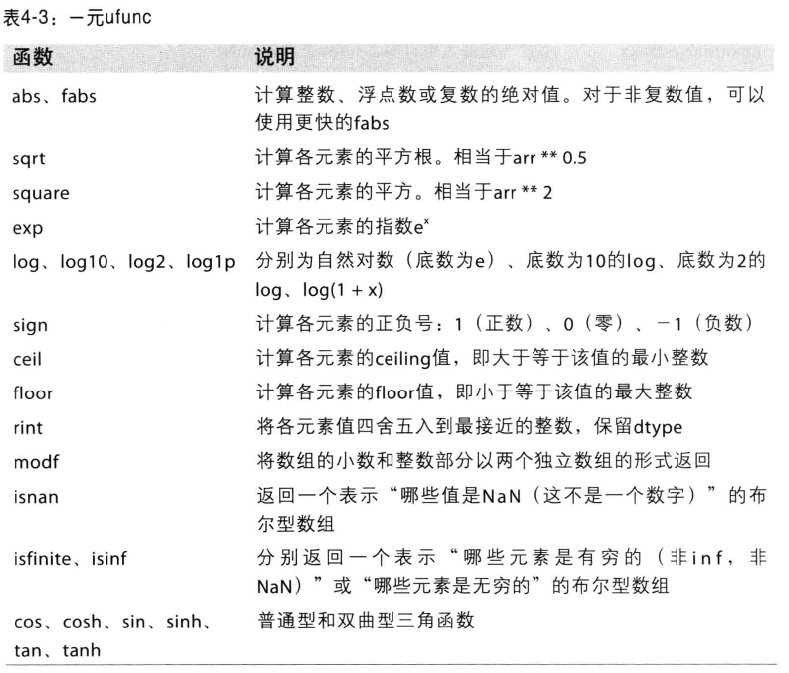
## Numpy tricks

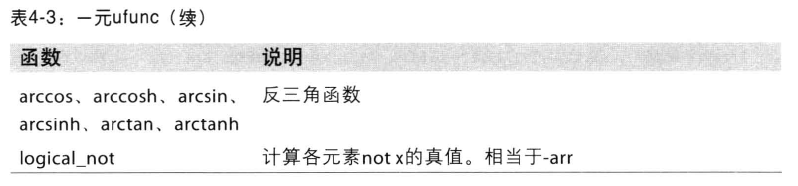
### Fancy indexing:

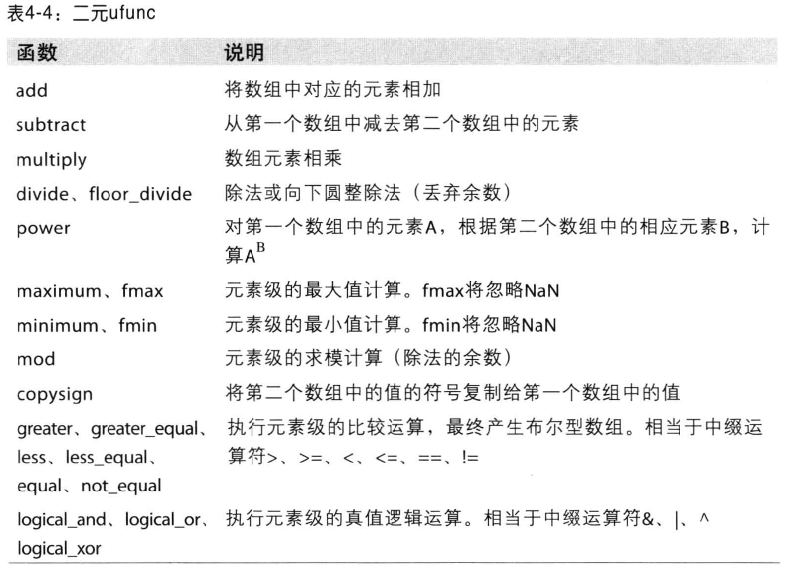




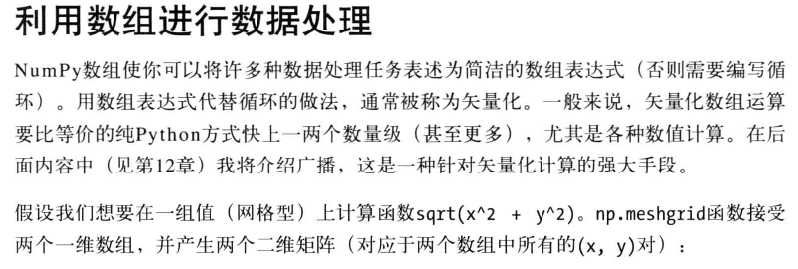
### ufunc

通用函数（即ufunc）是一种对ndarray中的数据执行元素级运算的函数。





### Vectorization array



Examples

**>>>** nx, ny = (3, 2)

**>>>** x = np.linspace(0, 1, nx)

**>>>** y = np.linspace(0, 1, ny)

**>>>** xv, yv = meshgrid(x, y)

**>>>** xv

array([[ 0. , 0.5, 1. ],

[ 0. , 0.5, 1. ]])

**>>>** yv

array([[ 0., 0., 0.],

[ 1., 1., 1.]])

**>>>** xv, yv = meshgrid(x, y, sparse=True) *# make sparse output arrays*

**>>>** xv

array([[ 0. , 0.5, 1. ]])

**>>>** yv

array([[ 0.],

[ 1.]])

[meshgrid](https://docs.scipy.org/doc/numpy/reference/generated/numpy.meshgrid.html#numpy.meshgrid) is very useful to evaluate functions on a grid.

>>>

**>>>** x = np.arange(-5, 5, 0.1)

**>>>** y = np.arange(-5, 5, 0.1)

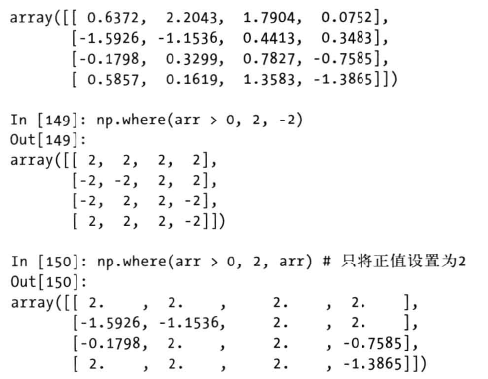
**>>>** xx, yy = meshgrid(x, y, sparse=True)

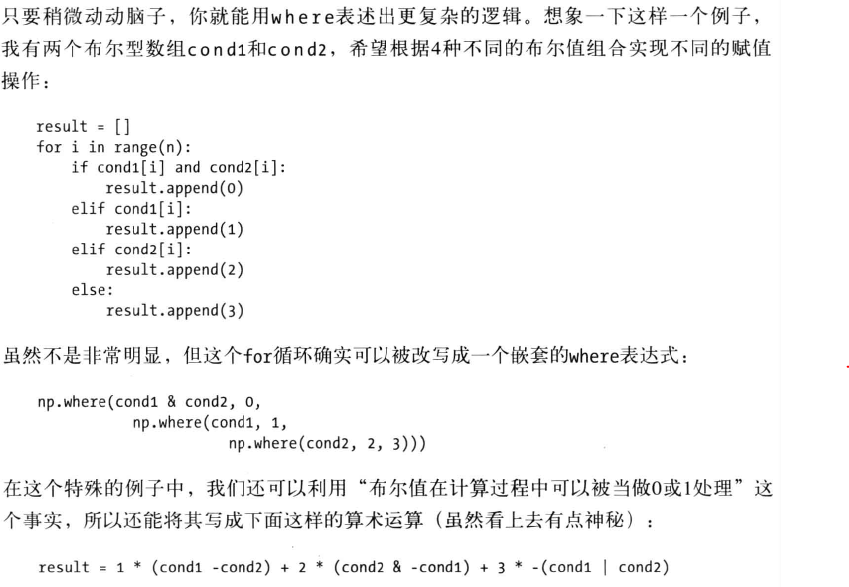
**>>>** z = np.sin(xx\*\*2 + yy\*\*2) / (xx\*\*2 + yy\*\*2)

**>>>** h = plt.contourf(x,y,z)

### Numpy.where

np.where(condition, xarr, yarr)

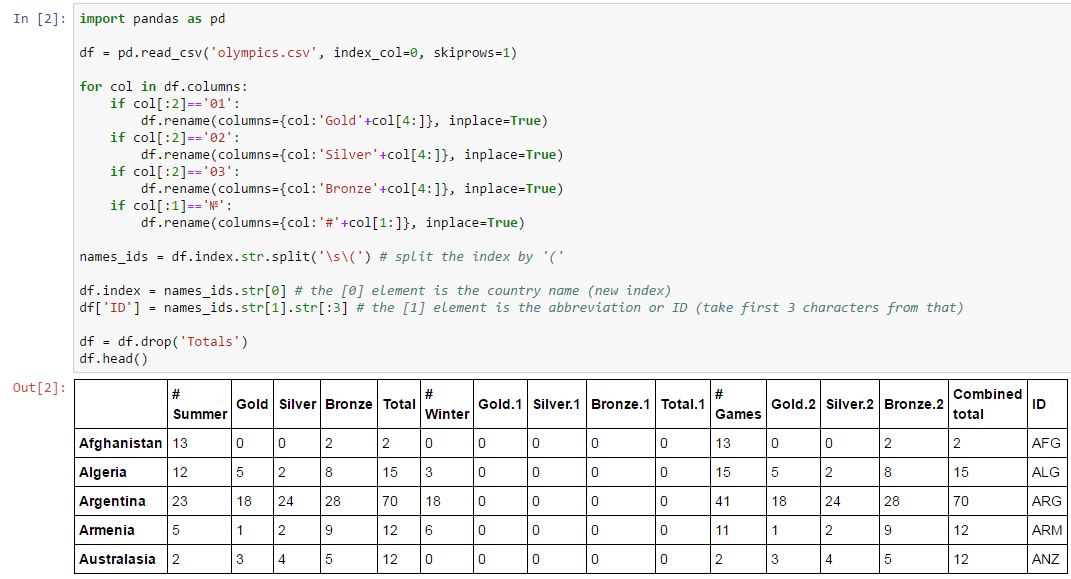


Complex :

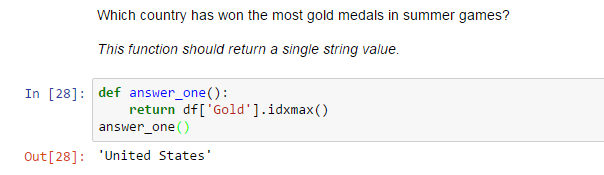
# Manipulate Series and Dataframe

## Basic Tech

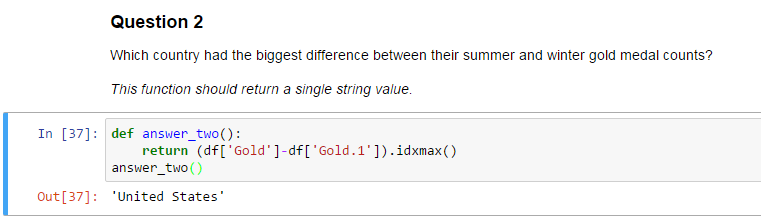
### Data:



### Select max id:



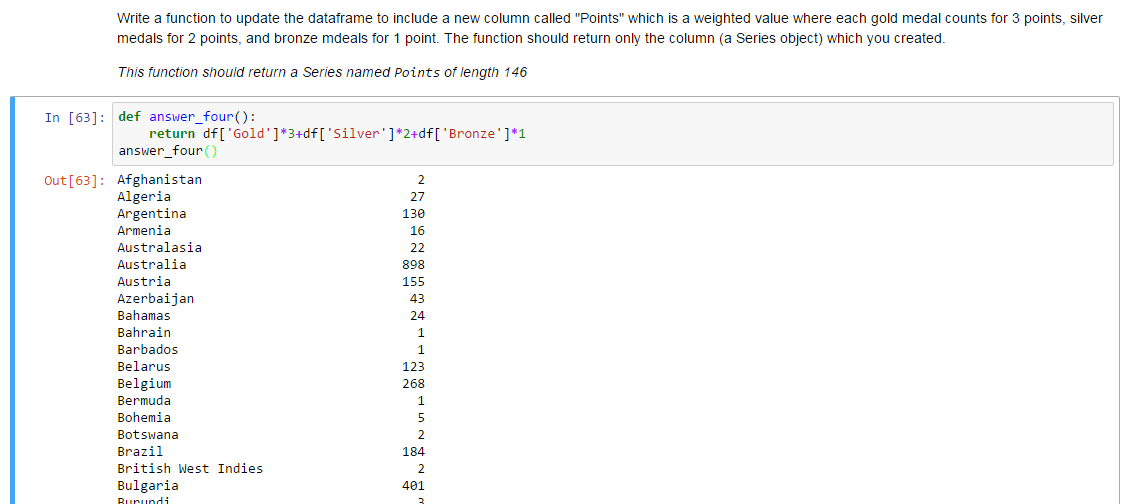
### Max difference:



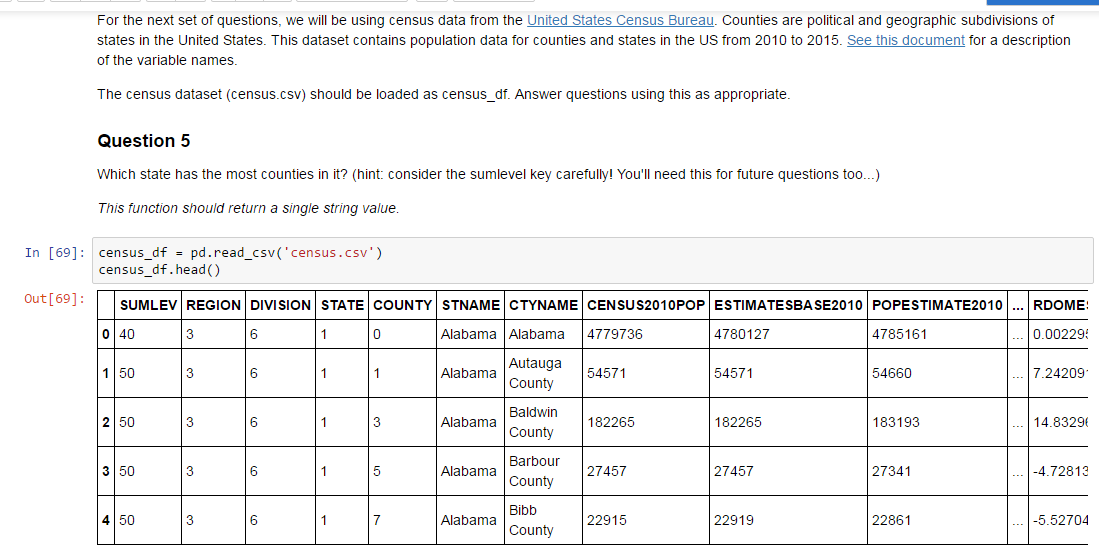
### Max difference abs percent:



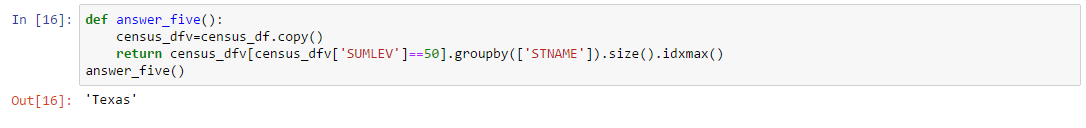
### Weighted value:



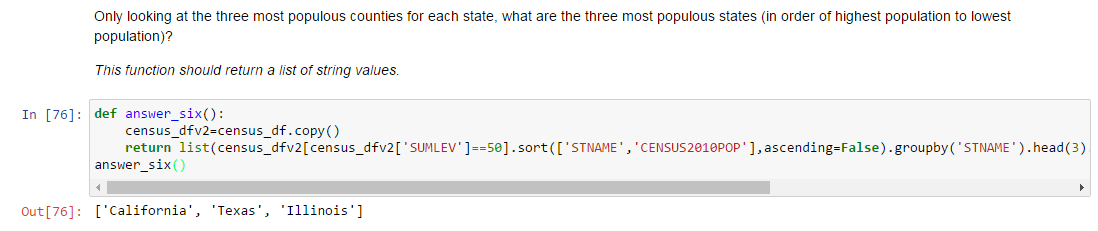
Data2:



### Groupby Count Max:



### Top 3 most populous states of top 3 counties:



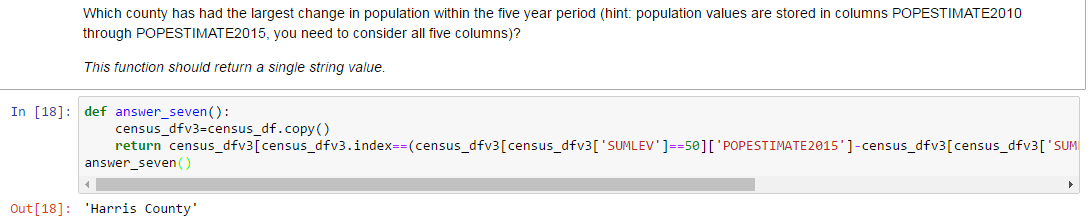
def answer\_six():

census\_dfv2=census\_df.copy()

return list(census\_dfv2[census\_dfv2['SUMLEV']==50].sort(['STNAME','CENSUS2010POP'],ascending=False).groupby('STNAME').head(3).groupby('STNAME')['CENSUS2010POP'].sum().nlargest(3).index)

answer\_six()

### Largest difference in population:



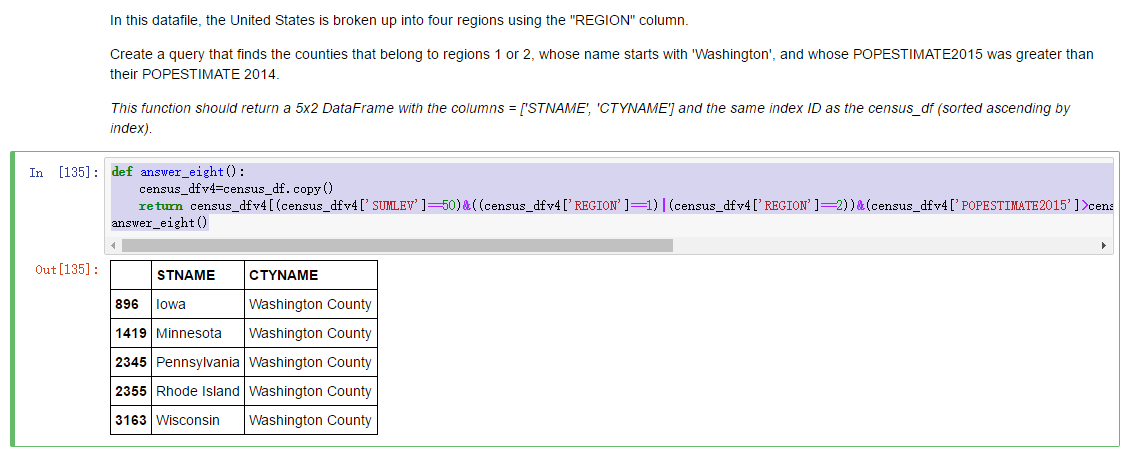
def answer\_seven():

census\_dfv3=census\_df.copy()

return census\_dfv3[census\_dfv3.index==(census\_dfv3[census\_dfv3['SUMLEV']==50]['POPESTIMATE2015']-census\_dfv3[census\_dfv3['SUMLEV']==50]['POPESTIMATE2010']).idxmax()]['CTYNAME'].values[0]

answer\_seven()

### Multiply condition query:



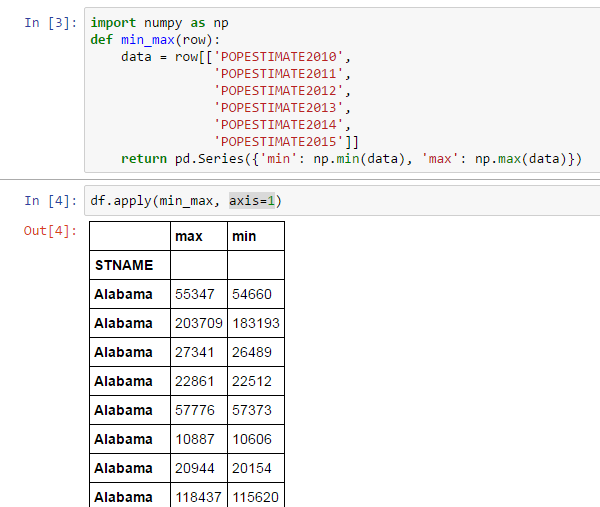
def answer\_eight():

census\_dfv4=census\_df.copy()

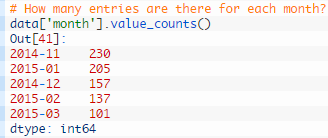
return census\_dfv4[(census\_dfv4['SUMLEV']==50)&((census\_dfv4['REGION']==1)|(census\_dfv4['REGION']==2))&(census\_dfv4['POPESTIMATE2015']>census\_dfv4['POPESTIMATE2014'])&(census\_dfv4['CTYNAME'].str.startswith('Washington'))].loc[:,['STNAME','CTYNAME']]

answer\_eight()

### Apply function:



### Value\_counts():count entries for each month



## Advanced Tech

### Data

Load the energy data from the file Energy Indicators.xls, which is a list of indicators of [energy supply and renewable electricity production](https://hub.coursera-notebooks.org/user/qaddobjikidpfmgyjqasxi/notebooks/Energy%20Indicators.xls) from the [United Nations](http://unstats.un.org/unsd/environment/excel_file_tables/2013/Energy%20Indicators.xls) for the year 2013, and should be put into a DataFrame with the variable name of **energy**.

Keep in mind that this is an Excel file, and not a comma separated values file. Also, make sure to exclude the footer and header information from the datafile. The first two columns are unneccessary, so you should get rid of them, and you should change the column labels so that the columns are:

['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable's]

Convert the energy supply and the energy supply per capita to gigajoules (there are 1,000,000 gigajoules in a petajoule). For all countries which have missing data (e.g. data with "...") make sure this is reflected as np.NaN values.

Rename the following list of countries (for use in later questions):

"Republic of Korea": "South Korea", "United States of America": "United States", "United Kingdom of Great Britain and Northern Ireland": "United Kingdom", "China, Hong Kong Special Administrative Region": "Hong Kong"

There are also several countries with parenthesis in their name. Be sure to remove these, e.g. 'Bolivia (Plurinational State of)' should be 'Bolivia'.

Next, load the GDP data from the file world\_bank.csv, which is a csv containing countries' GDP from 1960 to 2015 from [World Bank](http://data.worldbank.org/indicator/NY.GDP.MKTP.CD). Call this DataFrame **GDP**.

Make sure to skip the header, and rename the following list of countries:

"Korea, Rep.": "South Korea", "Iran, Islamic Rep.": "Iran", "Hong Kong SAR, China": "Hong Kong"

Finally, load the [Sciamgo Journal and Country Rank data for Energy Engineering and Power Technology](http://www.scimagojr.com/countryrank.php?category=2102), which ranks countries based on their journal contributions in the aforementioned area. Call this DataFrame **ScimEn**.

Join the three datasets: GDP, Energy, and ScimEn into a new dataset (using the intersection of country names). Use only the last 10 years (2006-2015) of GDP data and only the top 15 countries by Scimagojr 'Rank' (Rank 1 through 15).

The index of this DataFrame should be the name of the country.

### Merge

def answer\_one():

import pandas as pd

import numpy as np

energy=pd.read\_excel('Energy Indicators.xls',skiprows=16,skip\_footer=38).drop(['Unnamed: 0','Unnamed: 1'],1).rename(columns={'Unnamed: 2':'Country','Renewable Electricity Production':'% Renewable','Energy Supply per capita':'Energy Supply per Capita'})

energy.drop(energy.index[0],inplace=True)

energy.replace(r'\(.\*\)','',inplace=True,regex=True)

energy['Country']=energy['Country'].map(lambda x: x.rstrip(' 1234567890'))

energy.replace({'...':np.NaN,

"Republic of Korea": "South Korea",

"United States of America": "United States",

"United Kingdom of Great Britain and Northern Ireland": "United Kingdom",

"China, Hong Kong Special Administrative Region": "Hong Kong"},

inplace=True)

energy['Energy Supply']\*=1000000

GDP=pd.read\_csv('world\_bank.csv',skiprows=4)

GDP.replace({"Korea, Rep.": "South Korea",

"Iran, Islamic Rep.": "Iran",

"Hong Kong SAR, China": "Hong Kong"},

inplace=True)

ScimEn=pd.read\_excel('scimagojr-3.xlsx')

ScimEn=ScimEn[ScimEn['Rank']<=15]

Merge=(pd.merge(energy, GDP, how='inner', left\_on=['Country'], right\_on=['Country Name'])

.merge(ScimEn,how='inner',on=['Country'])

.set\_index('Country')

.sort('Rank'))[['Rank',

'Documents',

'Citable documents',

'Citations',

'Self-citations',

'Citations per document',

'H index',

'Energy Supply',

'Energy Supply per Capita',

'% Renewable',

'2006',

'2007',

'2008',

'2009',

'2010',

'2011',

'2012',

'2013',

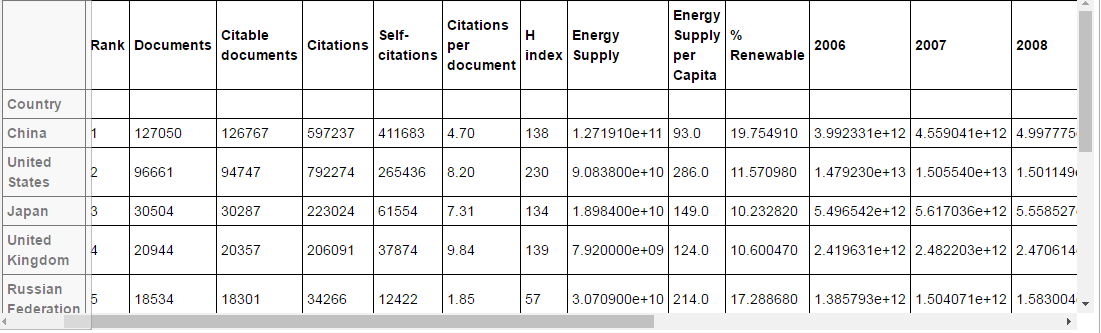
'2014',

'2015'

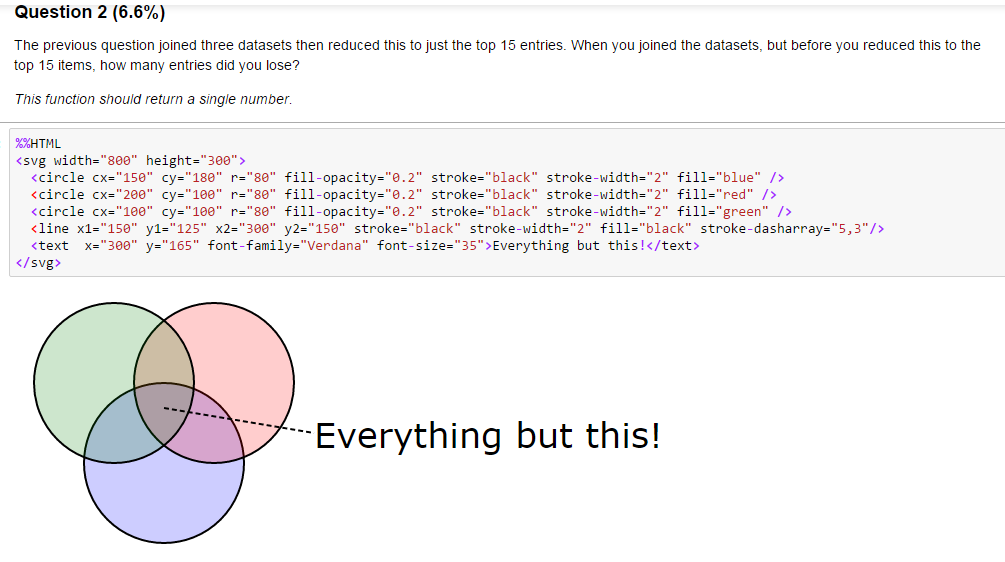
]]

return Merge

answer\_one()



### Replace and [Intersection](http://www.baidu.com/link?url=3EZt9TkpSB3l7AfLWYPZwQcZVYCtQoQuvxDxOSGKB-z6lYFQDFRKbKUj-BMn78X15S8x3LiBPdYoX37m2lxmqPj9_D7Rr6qjNMcupnDjJU_)



def answer\_two():

import pandas as pd

import numpy as np

energy=pd.read\_excel('Energy Indicators.xls',skiprows=16,skip\_footer=38).drop(['Unnamed: 0','Unnamed: 1'],1).rename(columns={'Unnamed: 2':'Country','Renewable Electricity Production':'% Renewable'})

energy.drop(energy.index[0],inplace=True)

energy.replace(r'\(.\*\)','',inplace=True,regex=True)

energy['Country']=energy['Country'].map(lambda x: x.rstrip(' 1234567890'))

energy.replace({'...':np.NaN,

"Republic of Korea": "South Korea",

"United States of America": "United States",

"United Kingdom of Great Britain and Northern Ireland": "United Kingdom",

"China, Hong Kong Special Administrative Region": "Hong Kong"},

inplace=True)

energy['Energy Supply']\*=1000000

GDP=pd.read\_csv('world\_bank.csv',skiprows=4)

GDP.replace({"Korea, Rep.": "South Korea",

"Iran, Islamic Rep.": "Iran",

"Hong Kong SAR, China": "Hong Kong"},

inplace=True)

ScimEn=pd.read\_excel('scimagojr-3.xlsx')

I=(pd.merge(energy, GDP, how='inner', left\_on=['Country'], right\_on=['Country Name'])

.merge(ScimEn,how='inner',on=['Country']))

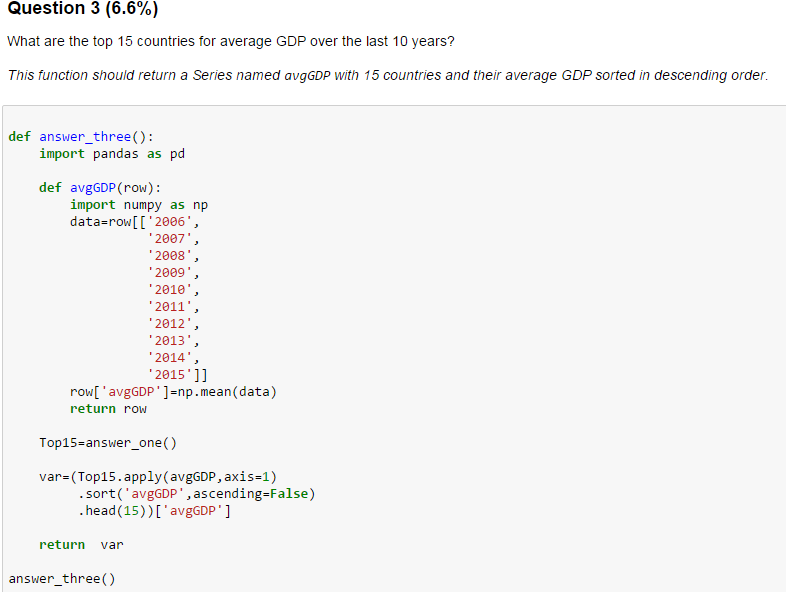
O=len((pd.merge(energy, GDP, how='outer', left\_on=['Country'], right\_on=['Country Name'])

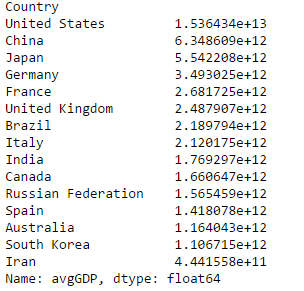
.merge(ScimEn,how='outer',on=['Country'])).index)

return O-len(I.index)-4

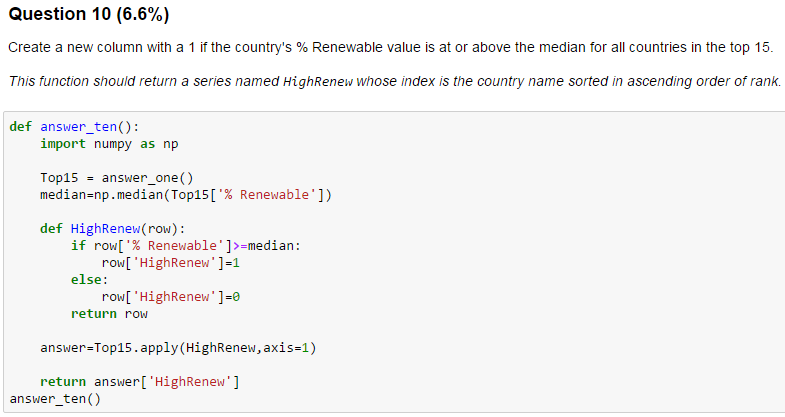
answer\_two()

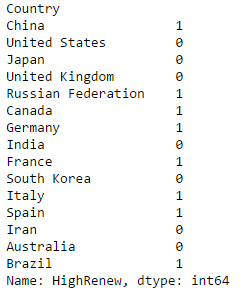
### Apply function



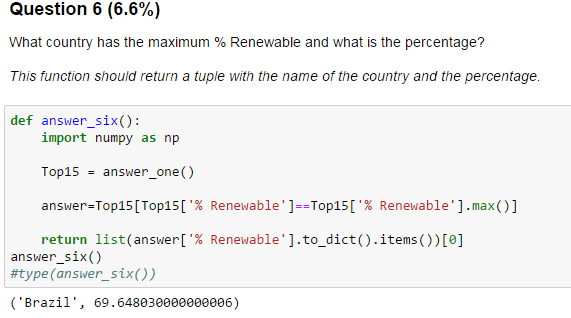


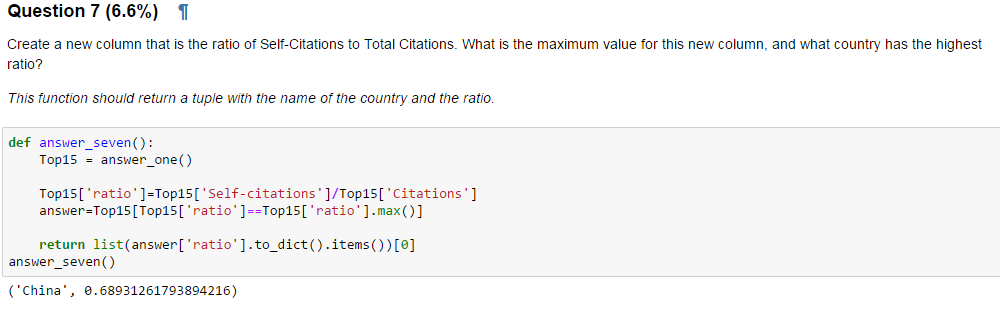


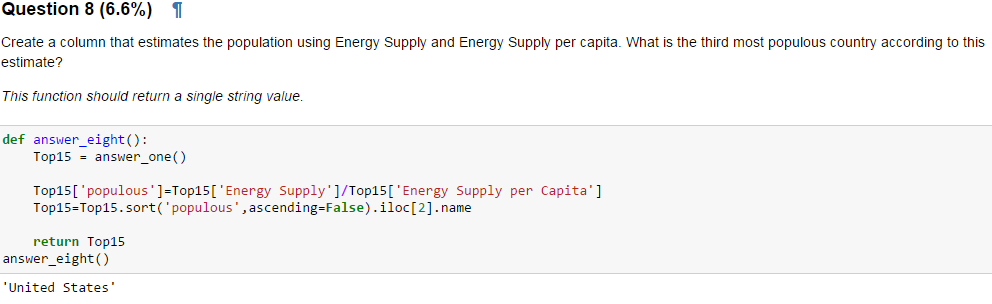




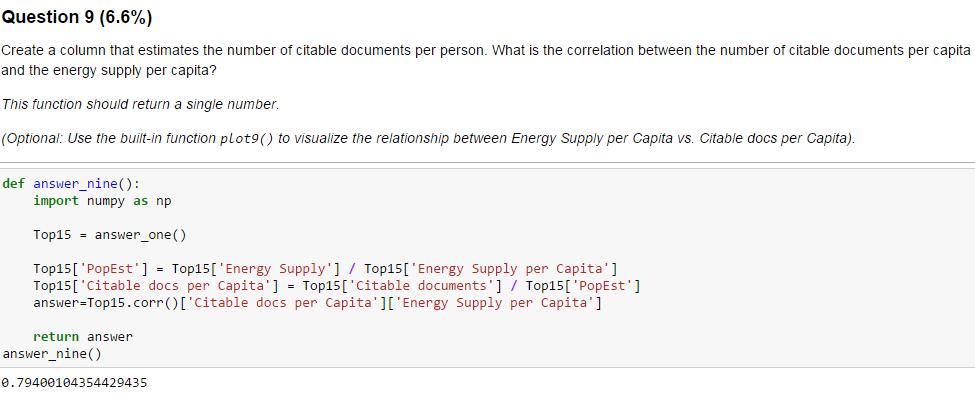
### Convert Max item

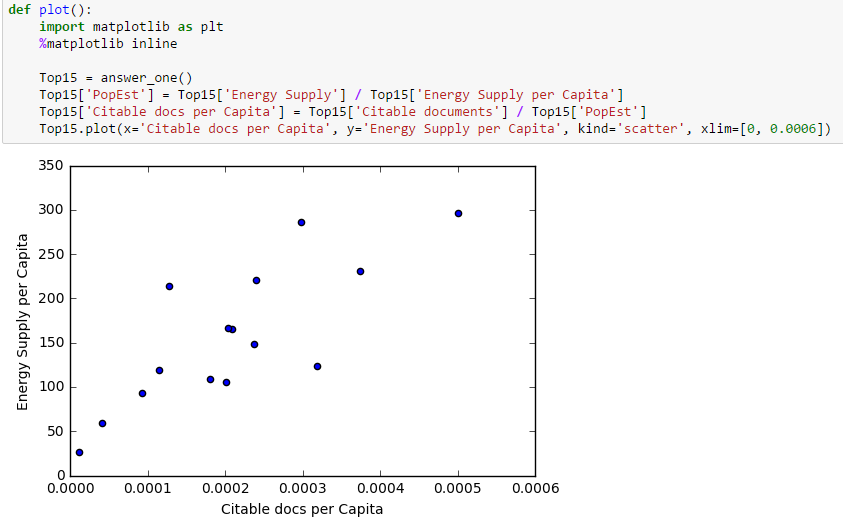






### Correlation





### Dictionary Group by



def answer\_eleven():

import pandas as pd

import numpy as np

Top15 = answer\_one()

ContinentDict = {'China':'Asia',

'United States':'North America',

'Japan':'Asia',

'United Kingdom':'Europe',

'Russian Federation':'Europe',

'Canada':'North America',

'Germany':'Europe',

'India':'Asia',

'France':'Europe',

'South Korea':'Asia',

'Italy':'Europe',

'Spain':'Europe',

'Iran':'Asia',

'Australia':'Australia',

'Brazil':'South America'}

Top15=Top15.reset\_index()

Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']

Top15['Continent']=Top15['Country'].map(ContinentDict)

answer=Top15.groupby('Continent').agg({'Country':{'size':np.count\_nonzero},

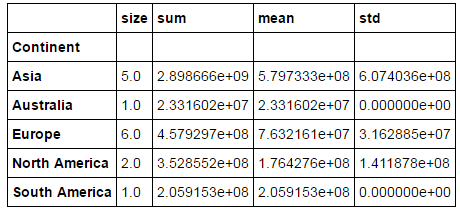
'PopEst':{'sum':np.sum,'mean':np.mean,'std':np.nanstd}}).astype(float)

answer.columns=['sum', 'mean', 'std', 'size']

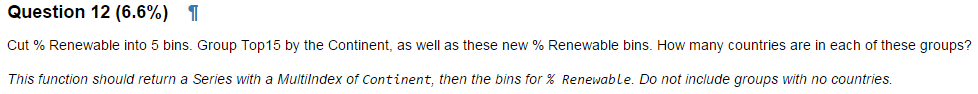
answer=answer[['size','sum', 'mean', 'std']]

return answer

answer\_eleven()



### Scale



def answer\_twelve():

import pandas as pd

import numpy as np

Top15 = answer\_one()

ContinentDict = {'China':'Asia',

'United States':'North America',

'Japan':'Asia',

'United Kingdom':'Europe',

'Russian Federation':'Europe',

'Canada':'North America',

'Germany':'Europe',

'India':'Asia',

'France':'Europe',

'South Korea':'Asia',

'Italy':'Europe',

'Spain':'Europe',

'Iran':'Asia',

'Australia':'Australia',

'Brazil':'South America'}

Top15=Top15.reset\_index()

Top15['Continent']=Top15['Country'].map(ContinentDict)

Top15['Bin']=pd.cut(Top15['% Renewable'],5)

answer=Top15.groupby(['Continent','Bin']).agg({'Country':{'Count':np.count\_nonzero}})

#answer2=answer.set\_index(answer.index.get\_level\_values(1))

return answer

answer\_twelve()

### Number Formating

