Assignment 2 - Pandas Introduction

This is from an assignment in Coursera's Applied Data Science in Python Specialization. It is intended as an introdution to data processing in *pandas* and uses two separate data sets.

The code to read the data and reformat columns was not written by me. However, all other code was.

For purposes of grading, everything was written into a function.

Part 1

The following code loads the olympics dataset (olympics.csv), which was derrived from the Wikipedia entry on All Time Olympic Games Medals, and does some basic data cleaning.

The columns are organized as # of Summer games, Summer medals, # of Winter games, Winter medals, total # number of games, total # of medals. Use this dataset to answer the questions below.

In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('olympics.csv', index_col=0, skiprows=1)
```

In [3]:

```
for col in df.columns:
    if col[:2]=='01':
        df.rename(columns={col:'Gold'+col[4:]}, inplace=True)
    if col[:2]=='02':
        df.rename(columns={col:'Silver'+col[4:]}, inplace=True)
    if col[:2]=='03':
        df.rename(columns={col:'Bronze'+col[4:]}, inplace=True)
    if col[:1]=='№':
        df.rename(columns={col:'#'+col[1:]}, inplace=True)

names_ids = df.index.str.split('\s\(') # split the index by '('

df.index = names_ids.str[0] # the [0] element is the country name (new index)
df['ID'] = names_ids.str[1].str[:3] # the [1] element is the abbreviation or ID (take first 3 chara
cters from that)

df = df.drop('Totals')
df.head()
```

Out[3]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	Bron
Afghanistan	13	0	0	2	2	0	0	0	0	0	13	0	0	2
Algeria	12	5	2	8	15	3	0	0	0	0	15	5	2	8
Argentina	23	18	24	28	70	18	0	0	0	0	41	18	24	28
Armenia	5	1	2	9	12	6	0	0	0	0	11	1	2	9
Australasia	2	3	4	5	12	0	0	0	0	0	2	3	4	5

Question 0 (Example)

What is the first country in df?

This function should return a Series.

You should write your whole answer within the function provided.

In [4]:

```
def answer_zero():
   return df.iloc[0]
answer_zero()
```

Out[4]:

```
# Summer           13
Gold                0
Silver              0
Bronze              2
Total               2
# Winter           0
Gold.1              0
Silver.1            0
Bronze.1            0
Total.1             0
# Games            13
Gold.2              0
Silver.2            0
Bronze.2            2
Combined total      2
ID                AFG
Name: Afghanistan, dtype: object
```

Question 1

Which country has won the most gold medals in summer games?

This function should return a single string value.

In [5]:

```
def answer_one():
    ##dropna() b/c otherwise you get the whole data frame with all rows with NaN values
    ##except the one you want (USA)
    df1=df.where(df['Gold']==df['Gold'].max()).dropna()
    return df1.index[0]
    ## also works df1.iloc[0].name
answer_one()
```

Out[5]:

```
'United States'
```

Question 2

Which country had the biggest difference between their summer and winter gold medal counts?

This function should return a single string value.

In [6]:

```
def answer_two():
    df['diff'] = df['Gold']-df['Gold.1']
    df1=df.where(df['diff']==df['diff'].max()).dropna()
    return df1.index[0]
answer_two()
```

Out[6]:

```
'United States'
```

Question 3

Which country has the biggest difference between their summer gold medal counts and winter gold medal counts relative to their total gold medal count?

$$\frac{Summer~Gold - Winter~Gold}{Total~Gold}$$

Only include countries that have won at least 1 gold in both summer and winter.

This function should return a single string value.

In [10]:

```
def answer_three():
    df1=df[(df['Gold']>0) & (df['Gold.1']>0)]
    df1['diffper']=((df['Gold'])-(df['Gold.1']))/(df['Gold.2'])
    df2=df1.where(df1['diffper']==df1['diffper'].max()).dropna()
    return df2.index[0]

answer_three()
```

```
C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[10]:

```
'Bulgaria'
```

Question 4

Write a function to update the dataframe to include a new column called "Points" which is a weighted value where each gold medal counts for 3 points, silver medals for 2 points, and bronze mdeals for 1 point. The function should return only the column (a Series object) which you created.

This function should return a Series named `Points` *of length 146*

In [11]:

```
def answer_four():
    df['Points']=(df['Gold.2']*3)+(df['Silver.2']*2)+(df['Bronze.2']*1)
    return df['Points']
answer_four()
```

Out[11]:

```
Afghanistan                       2
Algeria                          27
Argentina                       130
Armenia                          16
Australasia                      22
Australia                       923
Austria                         569
Azerbaijan                       43
Bahamas                          24
Bahrain                           1
Barbados                          1
Belarus                         154
Belgium                         276
Bermuda                           1
Bohemia                           5
Botswana                          2
Brazil                          184
British West Indies               2
Bulgaria                        411
Burundi                           3
Cameroon                         12
Canada                          846
Chile                            24
China                          1120
```

```
China                                1120
Colombia                               29
Costa Rica                              7
Ivory Coast                             2
Croatia                                67
Cuba                                  420
Cyprus                                  2
                                      ...
Spain                                 268
Sri Lanka                               4
Sudan                                   2
Suriname                                4
Sweden                               1217
Switzerland                           630
Syria                                   6
Chinese Taipei                         32
Tajikistan                              4
Tanzania                                4
Thailand                               44
Togo                                    1
Tonga                                   2
Trinidad and Tobago                    27
Tunisia                                19
Turkey                                191
Uganda                                 14
Ukraine                               220
United Arab Emirates                    3
United States                        5684
Uruguay                                16
Uzbekistan                             38
Venezuela                              18
Vietnam                                 4
Virgin Islands                          2
Yugoslavia                            171
Independent Olympic Participants        4
Zambia                                  3
Zimbabwe                               18
Mixed team                             38
Name: Points, Length: 146, dtype: int64
```

Part 2

For the next set of questions, we will be using census data from the United States Census Bureau. Counties are political and geographic subdivisions of states in the United States. This dataset contains population data for counties and states in the US from 2010 to 2015. See this document for a description of the variable names.

The census dataset (census.csv) should be loaded as census_df. Answer questions using this as appropriate.

Question 5

Which state has the most counties in it? (hint: consider the sumlevel key carefully! You'll need this for future questions too...)

This function should return a single string value.

In [16]:

```
census_df = pd.read_csv(r'C:/Users/mjdun/Desktop/Applied Data Science in Python/Files/census.csv')
census_df.head()
```

Out[16]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2010	POPEST
0	40	3	6	1	0	Alabama	Alabama	4779736	4780127	4785161
1	50	3	6	1	1	Alabama	Autauga County	54571	54571	54660
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265	183193
3	50	3	6	1	5	Alabama	Barbour County	27457	27457	27341

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2010	POPEST
4	50	3	6	1	7	Alabama	Bibb County	22915	22919	22861

5 rows × 100 columns

In [17]:

```
def answer_five():
    ##limit data frame to county level data
    clevel = census_df.where(census_df['SUMLEV']==50)
    ##count up all the records by the value in 'STNAME' column. Get the index (State name here) of
the maximum count
    return clevel['STNAME'].value_counts().idxmax()
answer_five()
```

Out[17]:

```
'Texas'
```

Question 6

Only looking at the three most populous counties for each state, what are the three most populous states (in order of highest population to lowest population)?

This function should return a list of string values.

In [19]:

```
def answer_six():

    ##limit data frame to county level data
    clevel = census_df.where(census_df['SUMLEV'] == 50)
    ##sort by state name and then population, descending
    a = clevel.sort_values(['STNAME', 'CENSUS2010POP'], ascending = False)
    ##group by state name, get top three counties in each state
    a=a.groupby('STNAME').head(3)
    ##group by state name, sum the populations (there are only 3 records per state)
    a=a.groupby('STNAME')['CENSUS2010POP'].sum()
    ##sort by index (0=index instead of 1=columns), descending
    a=a.sort_values(0, False)
    ##get top three
    s=a.index[0:3]
    ##adds these to a list of string values, also works list(a.head(3).index.values)
    list=[]
    for i in s:
        list.append(i)
    return list

answer_six()
```

Out[19]:

```
['California', 'Texas', 'Illinois']
```

Question 7

Which county has had the largest absolute change in population within the period 2010-2015? (Hint: population values are stored in columns POPESTIMATE2010 through POPESTIMATE2015, you need to consider all six columns.)

e.g. If County Population in the 5 year period is 100, 120, 80, 105, 100, 130, then its largest change in the period would be |130-80| = 50.

This function should return a single string value.

In [21]:

```
def answer_seven():
    clevel = census_df.where(census_df['SUMLEV']==50).dropna()
    ##find the largest value for each record in this list of columns, axis=1 for columns instead o
```

```
f index, create a new column in data frame
    clevel['max']=clevel[['POPESTIMATE2010', 'POPESTIMATE2011', 'POPESTIMATE2012', 'POPESTIMATE2013
', 'POPESTIMATE2014', 'POPESTIMATE2015']].max(axis=1)
    ##find the smallest value for each record in this list of columns, axis=1 for columns instead
of index, create a new column in data frame
    clevel['min']=clevel[['POPESTIMATE2010', 'POPESTIMATE2011', 'POPESTIMATE2012', 'POPESTIMATE2013
', 'POPESTIMATE2014', 'POPESTIMATE2015']].min(axis=1)
    ##create a new column of the difference between max and min for each record
    clevel['change']=(clevel['max']-clevel['min'])
    ##sort data frame by change
    clevel=clevel.sort_values(['change'], ascending=False)
    ##extract value from 'CTYNAME' column from first row
    a=clevel.iloc[0]['CTYNAME']
    return a
answer_seven()
```

Out[21]:

```
'Harris County'
```

Question 8

In this datafile, the United States is broken up into four regions using the "REGION" column.

Create a query that finds the counties that belong to regions 1 or 2, whose name starts with 'Washington', and whose POPESTIMATE2015 was greater than their POPESTIMATE 2014.

This function should return a 5x2 DataFrame with the columns = ['STNAME', 'CTYNAME'] and the same index ID as the census_df (sorted ascending by index).

In [22]:

```
def answer_eight():
    ##only county level
    clevel = census_df[(census_df['SUMLEV']==50)].dropna()
    ##only region 1 or 2
    clevel = clevel[(clevel['REGION']==1) | (clevel['REGION']==2)]
    ##only where 2015 pop. > 2014 pop.
    clevel = clevel[(clevel['POPESTIMATE2015'])>(clevel['POPESTIMATE2014'])]
    ##only where county name starts with Washington
    clevel = clevel[(clevel['CTYNAME'].str.startswith("Washington"))]
    ##return just state name and county name columns as a new data frame
    return clevel[['STNAME', 'CTYNAME']]
answer_eight()
```

Out[22]:

	STNAME	CTYNAME
896	Iowa	Washington County
1419	Minnesota	Washington County
2345	Pennsylvania	Washington County
2355	Rhode Island	Washington County
3163	Wisconsin	Washington County