Assignment 2

Assignment 2 - Pandas Introduction¶

All questions are weighted the same in this assignment.

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

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

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 characters from that)

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

Out[1]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	Bronze.2	Combined total	ID
Afghanistan	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2	AFG
Algeria	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15	ALG
Argentina	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70	ARG
Armenia	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12	ARM
Australasia	2	3	4	5	12	0	0	0	0	0	2	3	4	5	12	ANZ

```
### Question 0 (Example)

What is the first country in df?

*This function should return a Series.*
```

Question 0 (Example)¶
What is the first country in df?
This function should return a Series.

In [2]:

```
# You should write your whole answer within the function
provided. The autograder will call
# this function and compare the return value against the
correct solution value
def answer_zero():
    # This function returns the row for Afghanistan, which is
a Series object. The assignment
    # question description will tell you the general format
the autograder is expecting
    return df.iloc[0]

# You can examine what your function returns by calling it in
the cell. If you have questions
# about the assignment formats, check out the discussion
forums for any FAQs
answer_zero()
```

Out[2]:

```
# 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 [3]:

```
def answer_one():
    return df['Gold'].argmax()
answer_one()
```

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 [4]:

```
def answer_two():
    return (df['Gold']-df['Gold.1']).argmax()
```

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?

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

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

Only include countries that have won at least 1 gold in both summer and winter.
This function should return a single string value.
In [5]:

```
def answer_three():
    xdf = df.copy()
    xdf = xdf[(xdf['Gold']>0) & (xdf['Gold.1']>0)]
    return((xdf['Gold']-xdf['Gold.1'])/xdf['Gold.2']).argmax()
```

Question 4¶
Write a function that creates a Series called "Points" which is a weighted value where each gold medal (`Gold.2`) counts for 3 points, silver medals (`Silver.2`) for 2 points, and bronze medals (`Bronze.2`) for 1 point. The function should return only the column (a Series object) which you created, with the country names as indices.
This function should return a Series named `Points` of length 146
In [6]:

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

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 [7]:

```
census_df = pd.read_csv('census.csv')
census_df.head()
```

Out[7]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2010	POPESTIMATE2010	...	RDOMESTICMIG2011	RDOMESTICMIG2012	RDOMESTICMIG2013	RDOMESTICMIG2014	RDOMESTICMIG2015	RNETMIG2011	RNETMIG2012	RNETMIG2013	RNETMIG2014	RNETMIG2015
0	40	3	6	1	0	Alabama	Alabama	4779736	4780127	4785161	...	0.002295	-0.193196	0.381066	0.582002	-0.467369	1.030015	0.826644	1.383282	1.724718	0.712594
1	50	3	6	1	1	Alabama	Autauga County	54571	54571	54660	...	7.242091	-2.915927	-3.012349	2.265971	-2.530799	7.606016	-2.662146	-2.722002	2.592270	-2.187333

2	50	3	6	1	3	Alabama	Baldwin County	182265	182265	183193	...	14.8329600	17.6472933	21.8457005	19.2432887	17.1978722	15.8441776	18.5596227	22.7276266	20.3171142	18.2934499
3	50	3	6	1	5	Alabama	Barbour County	27457	27457	27341	...	-4.7281132	-2.5006900	-7.0568224	-3.9042117	-10.5432999	-4.8747411	-2.7581113	-7.1676764	-3.9785833	-10.5432999
4	50	3	6	1	7	Alabama	Bibb County	22915	22919	22861	...	-5.5270043	-5.0688871	-6.2010001	-0.1775375	0.1772258	-5.0883889	-4.3636366	-5.4037229	0.7545333	1.1078861

5 rows × 100 columns

In [8]:

```
def answer_five():
    return census_df['STNAME'].value_counts().argmax()
```

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)? Use `CENSUS2010POP`.

This function should return a list of string values.
In [9]:

```
def answer_six():
    df1 = census_df.copy()
    df1 = df1.groupby(['STNAME'])
    statePop = pd.DataFrame(columns=['pop'])
    for i, c in df1:

statePop.loc[i]=[c.sort_values(by='CENSUS2010POP',ascending =
False)[1:4]['CENSUS2010POP'].sum()]
    top = statePop.nlargest(3,'pop')
    return list (top.index)
```

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 [10]:

```
def answer_seven():
    pop =
census_df[['STNAME','CTYNAME','POPESTIMATE2015','POPESTIMATE20
14','POPESTIMATE2013','POPESTIMATE2012','POPESTIMATE2011','POP
ESTIMATE2010']]
    pop = pop[pop['STNAME']!= pop['CTYNAME']]
    index = (pop.max(axis=1)-pop.min(axis=1)).argmax()
    return census_df.loc[index]['CTYNAME']
```

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 [11]:

```
def answer_eight():
    return census_df[(census_df['REGION']<3 ) &
(census_df['CTYNAME'] == 'Washington County') &
(census_df['POPESTIMATE2015']>census_df['POPESTIMATE2014'])]
```

```
[['STNAME','CTYNAME']]

END
```