Assignment 1(a)

All questions are weighted the same in this assignment.

Part 1

The following code loads the olympics dataset (olympics.csv), which was derived 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.

Note: Please open the olympic.csv file and go through the code in the below block to see how cleaning has been done to view the data in understandable format.

In [49]:
```
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(10)
```

Out[49]:

	Summer	Gold	Silver	Bronze	Total	Winter	Gold.1	Silver.1	Bronze.1	Total.1	Games
Afghanistan	13	0	0	2	2	0	0	0	0	0	1
Algeria	12	5	2	8	15	3	0	0	0	0	1
Argentina	23	18	24	28	70	18	0	0	0	0	1
Armenia	5	1	2	9	12	6	0	0	0	0	1
Australasia	2	3	4	5	12	0	0	0	0	0	1
Australia	25	139	152	177	468	18	5	3	4	12	4
Austria	26	18	33	35	86	22	59	78	81	218	4
Azerbaijan	5	6	5	15	26	5	0	0	0	0	1
Bahamas	15	5	2	5	12	0	0	0	0	0	1
Bahrain	8	0	0	1	1	0	0	0	0	0	1

In [ ]:
```

```

Question 0 (Example)

What is the first country in df?

This function should return a Series.

In [52]:
```
#This function returns the first row of the series
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(10)

#This function returns the first row of the series
def answer_zero():
    #Write your code below
    return df.iloc[0]

df.iloc[0].name
```

Out[52]:
```
'Afghanistan'
```

Question 1

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

This function should return a single string value.

In [19]:
```
def answer_one():
    #Write your code below
    return df['Gold'].idxmax()

answer_one()
```

Out[19]:
```
'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 [21]:
```
def answer_two():
    #Write your code below
    diff = df['Gold']-df['Gold.1']
    return diff.idxmax()
answer_two()
```

Out[21]:
```
'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 [63]:
```
def answer_three():
    eligible = df[(df['Gold'] >= 1) & (df['Gold.1'] >= 1)]
    ratio = (eligible['Gold'] - eligible['Gold.1']).abs()/eligible['Gold.2']
    return ratio.idxmax()

answer_three()
    #Write your code below
```

Out[63]:
```
'Bulgaria'
```

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 [29]:
```
import pandas as pd
def answer_four():
    #write your code below
    Points = pd.Series(df['Gold.2']*3+df['Silver.2']*2+df['Bronze.2']*1)
    return Points
answer_four()
```

Out[29]:
```
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             4
Brazil             184
British West Indies  7
Bulgaria           411
Burundi              3
Cameroon            12
Canada             846
Chile               24
China             1120
Colombia            29
Costa Rica           7
Ivory Coast          2
Croatia             67
Cuba               420
Cyprus               2
                  ...
Spain              268
Sri Lanka            1
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               10
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
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 [70]:
```
import pandas as pd
census_df = pd.read_csv('census.csv')
census_df.head()
class CensusVariables:
    state_name = "STNAME"
    county_name = "CTYNAME"
    region = "REGION"
    census_pop2010 = "CENSUS2010POP"
    population_2014 = "POPESTIMATE2014"
    population_2015 = "POPESTIMATE2015"
    population_estimates = ["POPESTIMATE2010",
                            "POPESTIMATE2011",
                            "POPESTIMATE2012",
                            "POPESTIMATE2013",
                            population_2014,
                            population_2015]
    county_level = 50
    summary_level = "SUMLEV"

counties = census_df[census_df[
    CensusVariables.summary_level]==CensusVariables.county_level]
# this throws off the numeric index for the argmax method so reset it
counties = counties.reset_index()

# but the last question wants the original index
counties_original_index = census_df[census_df[
    CensusVariables.summary_level]==CensusVariables.county_level]
```

In [72]:
```
def answer_five():
    return counties.groupby(
        CensusVariables.state_name).count().COUNTY.idxmax()

answer_five()
```

Out[72]:
```
'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)? Use `CENSUS2010POP`.

This function should return a list of string values.

In [99]:
```
def answer_six():

    #write your code below
    top_threes = counties.groupby(
        CensusVariables.state_name
    )[CensusVariables.census_population].nlargest(3)
    states = top_threes.groupby(level=0).sum()
    return list(states.nlargest(3).index)

answer_six()
```

Out[99]:
```
['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 [86]:
```
def answer_seven():
    #write your code below
    return counties.iloc[
        (counties[
            CensusVariables.population_estimates].max(axis=1) -
         counties[
            CensusVariables.population_estimates].min(axis=1)
        ).idxmax()][CensusVariables.county_name]

answer_seven()
```

Out[86]:
```
'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 [111]:
```
#could not import tabulate fan.
from tabulate import tabulate
def answer_eight():
    #Write your code below
    regions = counties_original_index[
        (counties_original_index[CensusVariables.region]==1) |
        (counties_original_index[CensusVariables.region]==2)
    ][CensusVariables.state_name].isin(
        regions[CensusVariables.county_name].str.startswith("Washington")]
    grew = washingtons[washingtons[CensusVariables.population_2015] >
                       washingtons[CensusVariables.population_2014]]
    return grew[[CensusVariables.state_name,
                 CensusVariables.county_name]]

outcome = answer_eight()
assert outcome.shape == (3,2)
assert list(outcome.columns) == ['STNAME', 'CTYNAME']
print(tabulate(outcome, headers="keys", tablefmt="plain"))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-111-57f14e625427> in <module>
----> 2 from tabulate import tabulate
      3 def answer_eight():
      4     #Write your code below
```