Assignment 2 - Pandas Introduction

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.

```
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()
```

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. 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()
```

Question 1

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

This function should return a single string value.

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

```
def answer_two():
    summer = df['Gold']
    winter = df['Gold.1']
    delt = summer - winter
    return delt[delt == max(delt)].index[0]
answer_two()
```

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.

```
def answer_three():
    f1 = df[df['Gold']>0]
    f2 = f1[f1['Gold.1']>0]
    summer = f2['Gold']
    winter = f2['Gold.1']
    total = summer + winter
    relative = (summer - winter) / total
    return relative[relative == max(relative)].index[0]
answer_three()
```

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.

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

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

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.

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

```
census_df.head()
def answer_five():
    grouped =
census_df.groupby('STNAME').count().sort_index(by='SUMLEV',ascending=False).i
x[0].name
    return grouped
answer_five()
```

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.

```
def answer_six():
    f1 = census_df[census_df['SUMLEV'] ==
50].groupby('STNAME')['CENSUS2010POP'].apply(lambda x:
x.nlargest(3).sum()).nlargest(3).index.values.tolist()
    return f1
answer_six()
```

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.

```
def answer_seven():
    f1 = census_df[census_df['SUMLEV'] ==
50].set_index(['STNAME','CTYNAME']).ix[:,['POPESTIMATE2010','POPESTIMATE2011'
,'POPESTIMATE2012','POPESTIMATE2013','POPESTIMATE2014','POPESTIMATE2015']].st
ack()
    f2 = f1.max(level=['STNAME','CTYNAME']) -
f1.min(level=['STNAME','CTYNAME'])
    return f2.idxmax()[1]
answer_seven()
```

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).

```
def answer_eight():
    f1 = census_df[(census_df['POPESTIMATE2015'] >
census_df['POPESTIMATE2014']) & ((census_df['REGION'] == 1) |
(census_df['REGION'] == 2))]
    f1 = f1[f1['CTYNAME'].str.startswith('Washington')]
    return f1.ix[:,['STNAME', 'CTYNAME']]
answer_eight()
```

def answer_seven():

```
        columns_to_keep = ['POPESTIMATE2010',
                        'POPESTIMATE2011',
                        'POPESTIMATE2012',
                        'POPESTIMATE2013',
                        'POPESTIMATE2014',
                        'POPESTIMATE2015']
        df_cal=census_df[columns_to_keep].T
        columns_to_keep = ['STNAME',
                        'CTYNAME',
                        'COUNTY']
        df_result = census_df[columns_to_keep]
        df_result['Diff'] = 0
        for i in range(0,len(df_result)):
                df_result.loc[i,['Diff']] = df_cal[i].max()- df_cal[i].min()
                a=df_result.ix[df_result['Diff'].argmax()]['CTYNAME']
        return a
answer_seven()
```