

## 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 derived from the Wikipedia entry on [All Time Olympic Games Medals](https://en.wikipedia.org/wiki/All-time\_Olympic\_Games\_medal\_table), 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.

## 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 derived from the Wikipedia entry on [All Time Olympic Games Medals](https://en.wikipedia.org/wiki/All-time_Olympic_Games_medal_table), 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]=='N':
        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]:

	# S u m m e r	G o l d	S i l v e r	B r o n z e	T o t a l	# W i n t e r	G o l d. 1	S i l v e r. 1	B r o n z e .1	T o t a l.1	# G a m e s	G o l d. 2	S i l v e r .2	B r o n z e .2	C o m b i n e d t o t a l	ID
Af g h a n i s t a n	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2	AF G
Al ge ri a	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15	AL G
Ar ge nt in a	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70	A R G
Ar m en ia	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12	A R M

A us tr al as ia	2	3	4	5	12	0	0	0	0	0	2	3	4	5	12	A N Z
---------------------------------	---	---	---	---	----	---	---	---	---	---	---	---	---	---	----	-------------

### 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?

```
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.

	S U M L E V	R E G I O N	D I V I S I O N	S T A T E	C O U N T Y	S T N A M E	C T Y N A M E	C E N S U S 2 0 1 0 P O P	E S T I M A T E S B A S E 2 0 1 0	P O P E S T I M A T E 2 0 1 0	.. .	R D O M E S T I C M I G 2 0 1 1	R D O M E S T I C M I G 2 0 1 2	R D O M E S T I C M I G 2 0 1 3	R D O M E S T I C M I G 2 0 1 4	R D O M E S T I C M I G 2 0 1 5	R N E T M I G 2 0 1 1	R N E T M I G 2 0 1 2	R N E T M I G 2 0 1 3	R N E T M I G 2 0 1 4	R N E T M I G 2 0 1 5
0	40	3	6	1	0	A l a b a m a	A l a b a m a	4779736	4780127	4785161	.. .	0. 002295	- 0. 1916	0. 38106	0. 582002	- 0. 467369	1. 03015	0. 82644	1. 383282	1. 724718	0. 712594

1	50	3	6	1	1	Alabama	Autauga County	54571	54571	54660	..	7.242091	-22.915927	-3.012349	2.265371	-2.606016	-2.672204	2.592200	-2.18733
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265	183193	..	14.832960	17.647293	21.845730	19.129841	17.585967	22.7266	20.317742	18.293499
3	50	3	6	1	5	Alabama	Barbour County	27457	27457	27341	..	-4.728132	-2.506890	-7.05824	-3.905432	-1.075813	-4.87114	-3.97664	-1.05883

4	5	3	6	1	7	A	B	2	2	2	..	-	-	-	-	0	-	-	-	0	1.
	0					I	i	2	2	2	.	5	5	6	0	.	5	4	5	.	1
						a	b	9	9	8		.	.	.	.	1	.	.	.	7	0
						b	b	1	1	6		5	0	2	1	7	0	3	4	5	7
						a	C	5	9	1		2	6	0	7	7	8	6	0	4	8
						m	o					7	8	1	7	2	8	3	3	5	6
						a	u					0	8	0	5	5	3	6	7	3	1
						n						4	7	0	3	8	8	3	2	3	
						t						3	1	1	7		9	6	9		
						y															

5 rows x 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]:

ERROR::

```
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', 'POPESTIMATE2014', 'POPESTIMATE2013', 'POPESTIMATE2012', 'POPESTIMATE2011', 'POPESTIMATE2010']]
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']]

```