*You are currently looking at **version 1.2** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ (https://www.coursera.org/learn/python-data-analysis/resources/0dhYG) course resource.*

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 (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]=='№':
        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 inde
x)
df['ID'] = names_ids.str[1].str[:3] # the [1] element is the abbreviation o
r 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	Game
Afghanistan	13	0	0	2	2	0	0	0	0	0	
Algeria	12	5	2	8	15	3	0	0	0	0	
Argentina	23	18	24	28	70	18	0	0	0	0	
Armenia	5	1	2	9	12	6	0	0	0	0	
Australasia	2	3	4	5	12	0	0	0	0	0	

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 auto
grader will call
# this function and compare the return value against the correct solution v
alue

#def answer_zero():
    # This function returns the row for Afghanistan, which is a Series obje
ct. 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 FAQ
s

#answer_zero()
```

Question 1

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

This function should return a single string value.

计算过程

In [3]:
```
# 直接对 'Gold' 列使用 idxmax() 找出最大值对应的 index
df['Gold'].idxmax()
```

Out[3]:
```
'United States'
```

YC 答案

In [4]:
```
# yjc
def answer_one():
    return df['Gold'].idxmax()
```

标准答案 **1**

In [5]:
```
def answer_one():
    max_gold = 0
    for country in df.index:
        gold = df.loc[country,'Gold']
        if gold > max_gold:
            max_gold = gold
            max_country = country

    return max_country
```

Question 2

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

This function should return a single string value.

计算过程

In [6]:
```
# 直接对 'Gold' 列 与 'Gold.1' 列的差，使用 idxmax() 找出最大值对应的 index
(df['Gold'] - df['Gold.1']).idxmax()
```

Out[6]: 'United States'

YC 答案

In [7]:
```
# yjc
def answer_two():
    return (df['Gold'] - df['Gold.1']).idxmax()
```

标准答案 2

In [8]:
```
def answer_two():
    gold_max_diff = 0
    for country in df.index:
        gold_diff = df.loc[country,'Gold'] - df.loc[country,'Gold.1']
        if gold_diff > gold_max_diff:
            gold_max_diff = gold_diff
            country_saved = country

    return country_saved
```

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.

计算过程

In [9]:
```
# 排除夏冬都是 0 金牌的国家，避免除数(Total Gold)为 0.
# 需要将夏冬 0 金牌国家剔除。
gold_ctry = df[(df['Gold'] > 0) & (df['Gold.1'] > 0)]
gold_ctry.head()
```

Out[9]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games
Australia	25	139	152	177	468	18	5	3	4	12	43
Austria	26	18	33	35	86	22	59	78	81	218	48
Belarus	5	12	24	39	75	6	6	4	5	15	11
Belgium	25	37	52	53	142	20	1	1	3	5	45
Bulgaria	19	51	85	78	214	19	1	2	3	6	38

In [10]:
```
# 计算差值，并提取最大值的 index
(abs(gold_ctry['Gold'] - gold_ctry['Gold.1']) / (gold_ctry['Gold'] + gold_c
try['Gold.1'])).idxmax()
```

Out[10]: 'Bulgaria'

YC 答案

In [11]:
```
# yjc
# 排除夏冬都是 0 金牌的国家，避免除数(Total Gold)为 0.
def answer_three():
    f1 = df[df['Gold'] > 0]
    f2 = f1[f1['Gold.1'] > 0]
    summer = f2['Gold']
    winter = f2['Gold.1']
    return (abs(summer - winter) / (summer + winter)).idxmax()
```

标准答案 **3**

In [12]:
```
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]
```

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

计算过程

In [13]:
```
gold2 = df['Gold.2'] * 3
gold2.head(2)
```

Out[13]:
```
Afghanistan     0
Algeria        15
Name: Gold.2, dtype: int64
```

In [14]:
```
silver2 = df['Silver.2'] * 2
silver2.head(2)
```

Out[14]:
```
Afghanistan    0
Algeria        4
Name: Silver.2, dtype: int64
```

In [15]:
```
bronze2 = df['Bronze.2']
bronze2.head(2)
```

Out[15]:
```
Afghanistan    2
Algeria        8
Name: Bronze.2, dtype: int64
```

In [16]:
```
Points = gold2 + silver2 + bronze2
Points
```

Out[16]:
```
Afghanistan                          2
Algeria                             27
Argentina                          130
Armenia                             16
Australasia                         22
                                   ...
Yugoslavia                         171
Independent Olympic Participants     4
Zambia                               3
Zimbabwe                            18
Mixed team                          38
Length: 146, dtype: int64
```

YC 答案

In [17]:
```
# yjc
def answer_four():
    gold2 = df['Gold.2'] * 3
    silver2 = df['Silver.2'] * 2
    bronze2 = df['Bronze.2']
    Points = gold2 + silver2 + bronze2
    return Points
```

标准答案 4

In [18]:
```
def answer_four():
    dict = {}
    for country in df.index:
        point = df.loc[country,'Gold.2'] * 3 + df.loc[country,'Silver.2'] *
2 + df.loc[country,'Bronze.2']
        dict[country] = point
        Points = pd.Series(dict)
    return Points
```

Part 2

For the next set of questions, we will be using census data from the United States Census Bureau (http://www.census.gov/popest/data/counties/totals/2015/CO-EST2015-alldata.html). 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 (http://www.census.gov/popest/data/counties/totals/2015/files/CO-EST2015-alldata.pdf) for a description of the variable names.

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

In [19]:
```
census_df = pd.read_csv('census.csv')
census_df.head()
```

Out[19]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATES
0	40	3	6	1	0	Alabama	Alabama	4779736	
1	50	3	6	1	1	Alabama	Autauga County	54571	
2	50	3	6	1	3	Alabama	Baldwin County	182265	
3	50	3	6	1	5	Alabama	Barbour County	27457	
4	50	3	6	1	7	Alabama	Bibb County	22915	

5 rows × 100 columns

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.

计算过程

In [20]:
```
# 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计）
census_clean = census_df[census_df['STNAME'] != census_df['CTYNAME']]
census_clean.head()
```

Out[20]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATES
1	50	3	6	1	1	Alabama	Autauga County	54571	
2	50	3	6	1	3	Alabama	Baldwin County	182265	
3	50	3	6	1	5	Alabama	Barbour County	27457	
4	50	3	6	1	7	Alabama	Bibb County	22915	
5	50	3	6	1	9	Alabama	Blount County	57322	

5 rows × 100 columns

In [21]:
```
# 以STNAME为依据分组，并 count。产生新 DataFrame
county = census_clean.groupby('STNAME').count()
county.head()
```

Out[21]:

STNAME	SUMLEV	REGION	DIVISION	STATE	COUNTY	CTYNAME	CENSUS2010POP	ESTIMATESBA
Alabama	67	67	67	67	67	67	67	
Alaska	29	29	29	29	29	29	29	
Arizona	15	15	15	15	15	15	15	
Arkansas	75	75	75	75	75	75	75	
California	58	58	58	58	58	58	58	

5 rows × 99 columns

In [22]:
```
# 从 [COUNTY] 列的值中找出最大值的 index
county['COUNTY'].idxmax()
```

Out[22]: 'Texas'

YC 答案

In [23]:
```
# yjc
def answer_five():
    # 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计）
    census_clean = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    # 以STNAME为依据分组，并 count。产生新 DataFrame
    county = census_clean.groupby('STNAME').count()
    # 从 [COUNTY] 列的值中找出最大值的 index
    return county['COUNTY'].idxmax()
```

标准答案 **5**

In [24]:
```
def answer_five():
    census_clear = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    census_gp = census_clear.groupby('STNAME')
    county_num = census_gp.count()['COUNTY']
    state_most_county = county_num.idxmax()

    return state_most_county
```

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.

计算过程

In [25]:
```
# 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计，但可能统计错误）
census_state = census_df[census_df['STNAME'] != census_df['CTYNAME']]
```

In [26]:
```
# 找出每州人口最多的 3 个 county
top3_county = census_state.groupby('STNAME')['CENSUS2010POP'].nlargest(3)
top3_county
```

Out[26]:
```
STNAME
Alabama     37      658466
            49      412992
            45      334811
Alaska      71      291826
            76       97581
                      ...
Wisconsin   3109    488073
            3164    389891
Wyoming     3180     91738
            3182     75450
            3172     46133
Name: CENSUS2010POP, Length: 150, dtype: int64
```

In [27]:
```
# 按照州名来分组统计人口最多 3 个 county 总和
census_pop = top3_county.groupby('STNAME').sum()
census_pop.head()
```

Out[27]:
```
STNAME
Alabama        1406269
Alaska          478402
Arizona        5173150
Arkansas        807152
California    15924150
Name: CENSUS2010POP, dtype: int64
```

In [28]:
```
# 找到最大人口的 3 个州。并将 DataFrame 中，index 读取出来，并转换成 list
census_pop.nlargest(3).index.tolist()
```

Out[28]:
```
['California', 'Texas', 'Illinois']
```

YC 答案

In [29]:
```
# yjc
def answer_six():
    # 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计，但可能统计错误)
    census_state = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    # 找出每州人口最多的 3 个 county
    top3_county = census_state.groupby('STNAME')['CENSUS2010POP'].nlargest(
3)
    # 按照州名来分组统计人口最多 3 个 county 总和
    census_pop = top3_county.groupby('STNAME').sum()
    # 找到最大人口的 3 个州。并将 DataFrame 中，index 读取出来，并转换成 list
    return census_pop.nlargest(3).index.tolist()
```

标准答案 **6**

In [30]:
```
def answer_six():
    census_clear = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    return census_clear.groupby('STNAME')['CENSUS2010POP'].apply(lambda x:
x.nlargest(3).sum()).nlargest(3).index.tolist()
```

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.

计算过程

In [31]:
```
# 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计，但可能统计错误）
census_state = census_df[census_df['STNAME'] != census_df['CTYNAME']]
```

In [32]:
```
# 以州和郡名组成双 index，并提取相关列，组成新的 DataFrame。
county_pop = census_state.set_index(['STNAME','CTYNAME']).loc[:,['POPESTIMATE2010',
                                    'POPESTIMATE2011','POPESTIMATE2012','POPESTIMATE2013',
                                    'POPESTIMATE2014','POPESTIMATE2015']]
county_pop.head()
```

Out[32]:

		POPESTIMATE2010	POPESTIMATE2011	POPESTIMATE2012	POPESTIMATE2013
STNAME	**CTYNAME**				
Alabama	**Autauga County**	54660	55253	55175	55038
	Baldwin County	183193	186659	190396	195126
	Barbour County	27341	27226	27159	26973
	Bibb County	22861	22733	22642	22512
	Blount County	57373	57711	57776	57734

In [33]:
```
# 将人口数值列堆积起来(变成竖列，横列无法计算 max, min)
county_pop_stack = county_pop.stack()
county_pop_stack
```

Out[33]:
```
STNAME    CTYNAME
Alabama   Autauga County  POPESTIMATE2010    54660
                          POPESTIMATE2011    55253
                          POPESTIMATE2012    55175
                          POPESTIMATE2013    55038
                          POPESTIMATE2014    55290
                                             ...
Wyoming   Weston County    POPESTIMATE2011     7114
                          POPESTIMATE2012     7065
                          POPESTIMATE2013     7160
                          POPESTIMATE2014     7185
                          POPESTIMATE2015     7234
Length: 18846, dtype: int64
```

In [34]:
```
# 当 index 是 multi-index 时，使用 max(level) 来确定需要最大值的数据位置。
pop_diff = county_pop_stack.max(level = ['STNAME','CTYNAME']) - county_pop_
stack.min(level = ['STNAME','CTYNAME'])
pop_diff
```

Out[34]:
```
STNAME    CTYNAME
Alabama   Autauga County         687
          Baldwin County       20516
          Barbour County         852
          Bibb County            349
          Blount County          403
                                ...
Wyoming   Sweetwater County     1569
          Teton County         1828
          Uinta County          280
          Washakie County       229
          Weston County         169
Length: 3141, dtype: int64
```

In [35]:
```
# 找出最大值的 index 中的 county name
pop_diff.idxmax()[1]
```

Out[35]:
```
'Harris County'
```

YC 答案

In [36]:
```
# yjc
def answer_seven():
    # 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计，但可能统计错误)
    census_state = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    # 以州和郡名组成双 index，将人口数值列堆积起来(变成竖列，横列无法计算 max, mi
n)
    county_pop = census_state.set_index(['STNAME','CTYNAME']).loc[:,['POPES
TIMATE2010',
                                        'POPESTIMATE2011','POPESTIMATE201
2','POPESTIMATE2013',
                                        'POPESTIMATE2014','POPESTIMATE201
5']].stack()
    pop_diff = county_pop.max(level = ['STNAME','CTYNAME']) - county_pop.mi
n(level = ['STNAME','CTYNAME'])
    return pop_diff.idxmax()[1]
```

标准答案 7

In [37]:
```
def answer_seven():
    census_clear = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    f1 = census_clear.set_index(['STNAME','CTYNAME']).loc[:,['POPESTIMATE2010',
                            'POPESTIMATE2011','POPESTIMATE2012','POPESTIMATE2013',
                            'POPESTIMATE2014','POPESTIMATE2015']].stack()
    f2 = f1.max(level=['STNAME','CTYNAME']) - f1.min(level=['STNAME','CTYNAME'])
    return f2.idxmax()[1]
```

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).

计算过程

In [38]:
```
# 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计，但可能统计错误）
census_state = census_df[census_df['STNAME'] != census_df['CTYNAME']]
census_state.head(2)
```

Out[38]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATES
1	50	3	6	1	1	Alabama	Autauga County	54571	
2	50	3	6	1	3	Alabama	Baldwin County	182265	

2 rows × 100 columns

In [39]:
```
# 仅提取需要的列组成新 DataFrame
county_reg = census_state[['STNAME','CTYNAME','REGION','POPESTIMATE2014','POPESTIMATE2015']]
county_reg.head(2)
```

Out[39]:

	STNAME	CTYNAME	REGION	POPESTIMATE2014	POPESTIMATE2015
1	Alabama	Autauga County	3	55290	55347
2	Alabama	Baldwin County	3	199713	203709

In [40]:
```
# finds the counties that belong to regions 1 or 2
county_12 = county_reg[(county_reg['REGION'] == 1) | (county_reg['REGION']
== 2)]
county_12.head(2)
```

Out[40]:

	STNAME	CTYNAME	REGION	POPESTIMATE2014	POPESTIMATE2015
315	Connecticut	Fairfield County	1	945816	948053
316	Connecticut	Hartford County	1	896871	895841

In [41]:
```
# County Name starts with 'Washington'
county_wash = county_12[county_12['CTYNAME'].str.startswith('Washington')]
county_wash.head(2)
```

Out[41]:

	STNAME	CTYNAME	REGION	POPESTIMATE2014	POPESTIMATE2015
703	Illinois	Washington County	2	14394	14270
799	Indiana	Washington County	2	27904	27827

In [42]:
```
# POPESTIMATE2015 was greater than their POPESTIMATE 2014
# 然后提取 DataFrame 中 ['STNAME', 'CTYNAME'] 的值
county_wash[county_wash['POPESTIMATE2015'] > county_wash['POPESTIMATE2014'
]].loc[:,['STNAME', 'CTYNAME']]
```

Out[42]:

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

YC 答案

In [43]:
```
# yjc
def answer_eight():
    # 清除州名和郡名相同的错误信息（郡名和州名相同，表示该州的小计，但可能统计错误）
    census_state = census_df[census_df['STNAME'] != census_df['CTYNAME']]
    # 仅提取需要的列组成新 DataFrame
    county_reg = census_state[['STNAME','CTYNAME','REGION','POPESTIMATE2014','POPESTIMATE2015']]
    # finds the counties that belong to regions 1 or 2
    county_12 = county_reg[(county_reg['REGION'] == 1) | (county_reg['REGION'] == 2)]
    # County Name starts with 'Washington'
    county_wash = county_12[county_12['CTYNAME'].str.startswith('Washington')]
    # POPESTIMATE2015 was greater than their POPESTIMATE 2014
    # 然后提取 DataFrame 中 ['STNAME', 'CTYNAME'] 的值
    return county_wash[county_wash['POPESTIMATE2015'] > county_wash['POPESTIMATE2014']].loc[:,['STNAME', 'CTYNAME']]
```

标准答案 **8**

In [44]:
```
def answer_eight():
    census_clear = census_df[(census_df['STNAME']!=census_df['CTYNAME'])&((census_df['REGION']==1)|(census_df['REGION']==2))]
    census_rewrt = census_clear[['STNAME','CTYNAME','REGION','POPESTIMATE2015','POPESTIMATE2014']]
    f1 = census_rewrt[census_rewrt['POPESTIMATE2015']>census_rewrt['POPESTIMATE2014']]
    f2 = f1[f1['CTYNAME'].str.startswith('Washington')]

    return f2.loc[:,['STNAME','CTYNAME']]
```