Handout for Session 8 (Solutions Only)

Q1-a: Create the following `Series` object using three ways.

```
Fritos     20
Cheetos    15
Lays       25
dtype: int64
```

```
[24]: t=pd.Series({'Fritos':20,'Cheetos':15,'Lays':25})
      t
```

```
Fritos     20
Cheetos    15
Lays       25
dtype: int64
```

```
[25]: t=pd.Series([20,15,25],index=['Fritos','Cheetos','Lays'])
      t
```

```
Fritos     20
Cheetos    15
Lays       25
dtype: int64
```

```
[26]: t=pd.Series()
      t['Fritos']=20
      t['Cheetos']=15
      t['Lays']=25
      t
```

```
Fritos     20
Cheetos    15
Lays       25
dtype: int64
```

Q1-b: Obtain the single element corresponding to `"Lays"` using five ways.

```
[27]: t[2]
```

```
25
```

```
[28]: t[-1]
```

```
25
```

```
[29]: t.iloc[2]
```

```
25
```

```
[30]: t.iloc[-1]
```

```
25
```

```
[31]: t.loc['Lays']
```

```
25
```

Q1-c: Obtain everything but the first element using at least four ways.

```
[32]: t[1:]
```

```
Cheetos    15
Lays       25
dtype: int64
```

```
[33]: t.iloc[1:]
```

```
Cheetos    15
Lays       25
dtype: int64
```

```
[34]: t.loc['Cheetos':]
```

```
Cheetos    15
Lays       25
dtype: int64
```

```
[35]: t[[False,True,True]]
```

```
Cheetos    15
Lays       25
dtype: int64
```

2.3 Manipulating a Series Object

Q2-a: Run the function `describe` on the Series `gdp` (from part 1 of this handout).

```
[51]: gdp.describe()
```

```
count       12.000000
mean      6396.826912
std       3524.169583
min       2204.242423
25%       3664.915073
50%       5554.323909
75%       8606.556438
max      12934.458535
Name: gdp, dtype: float64
```

Q2-b: Write an expression divides the Series `gdp` by 1000 and round to 2 decimal places (using the `round` function).

```
[52]: round(gdp/1000,2)
```

```
year
1952     2.20
1957     2.55
1962     3.02
1967     3.88
```

```
1972     4.58
1977     5.30
1982     5.81
1987     7.26
1992     8.22
1997     9.76
2002    11.25
2007    12.93
Name: gdp, dtype: float64
```

Q2-c: Filter the Series gdp for values above 10000.

```
[53]: gdp[gdp>10000]
```

```
year
2002    11247.278678
2007    12934.458535
Name: gdp, dtype: float64
```

Q2-d: Obtain a Series corresponding to the life expectancy in USA when the GDP is above 10 trillion. (Hint: obtain the life expectancy column using `usa['lifeExp']` and use boolean indexing on `gdp1` as in Q2-b.)

```
[54]: usa['lifeExp'][gdp>10000]
```

```
year
2002    77.310
2007    78.242
Name: lifeExp, dtype: float64
```

Q2-e: Compute the average life expectancy in the data set for USA when the GDP is above 10 trillion. (Hint: call the function `mean` of the above Series.)

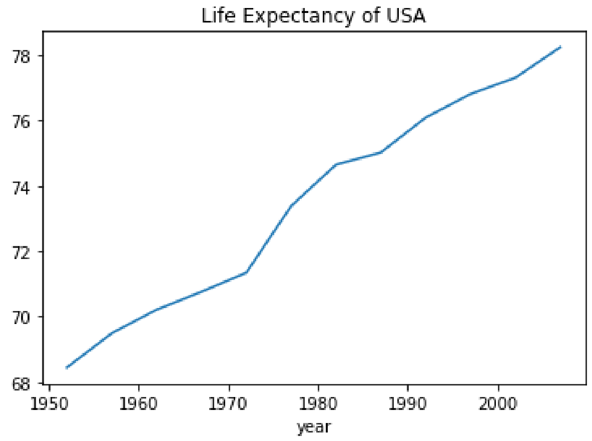
```
[55]: usa['lifeExp'][usa['gdp']>10000].mean()
```

```
77.77600000000001
```

Q2-f: Plot the life expectancy of USA in the data set using a line plot.

```
[56]: usa['lifeExp'].plot(title='Life Expectancy of USA')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb02ebc89b0>
```



3. Pandas DataFrame Basics II

Q3-a: Obtain the second column of the DataFrame `df` in at least three ways.

```
[71]: df['Rank']
```

```
apple     2
orange    1
grape     3
Name: Rank, dtype: int64
```

```
[72]: df.iloc[:,-1]
```

```
apple     M
orange    M
grape     S
Name: Size, dtype: object
```

```
[73]: df.loc[:,'Rank']
```

```
apple     2
orange    1
grape     3
Name: Rank, dtype: int64
```

Q3-b: Obtain the second and third row of the DataFrame `df` in at least five ways.

```
[74]: df.iloc[1:3,:]
```

```
        Number  Rank Size
orange       6     1    M
grape        4     3    S
```

```
[75]: df.iloc[[1,2],:]
```

```
        Number  Rank Size
orange       6     1    M
grape        4     3    S
```

```
[76]: df.loc['orange':'grape',:]
```

```
        Number  Rank Size
orange       6     1    M
grape        4     3    S
```

```
[77]: df.loc[['orange','grape'],:]
```

```
        Number  Rank Size
orange       6     1    M
grape        4     3    S
```

```
[78]: df[[False,True,True]]
```

```
        Number  Rank Size
orange       6     1    M
grape        4     3    S
```

Q3-c: Obtain the rank of orange in at least four ways.

```
[79]: df['Rank']['orange']
```

```
1
```

```
[80]: df['Rank'][2]
```

```
3
```

```
[81]: df.iloc[2,1]
```

```
3
```

```
[82]: df.loc['orange','Rank']
```

```
1
```

Q4-a: Obtain the set of unique continents in the DataFrame `data`. (Hint: use the function `unique` associated with the Series `data['continent']`.)

```
[83]: data['continent'].unique()
```

```
array(['Asia', 'Europe', 'Africa', 'Americas', 'Oceania'], dtype=object)
```

Q4-b: Filter for the rows of the DataFrame `data` for which the continent is "Americas", year is 2007, and GDP is at least 1000. (You can either use the `query` function associated with the DataFrame or boolean indexing.)

```
[84]: data.query('continent=="Americas" and year==2007 and gdp >=1000').head()
```

```
            country continent  lifeExp         pop   gdpPercap           gdp
year
2007         Brazil  Americas   72.390  190.010647    9.065801   1722.598680
2007         Canada  Americas   80.653   33.390141   36.319235   1212.704378
2007         Mexico  Americas   76.195  108.700891   11.977575   1301.973070
2007  United States  Americas   78.242  301.139947   42.951653  12934.458535
```

```
[85]: data[(data['continent']=='Americas') & (data.index==2007) & (data['gdp']>=1000)]
```

```
            country continent  lifeExp         pop   gdpPercap           gdp
year
2007         Brazil  Americas   72.390  190.010647    9.065801   1722.598680
2007         Canada  Americas   80.653   33.390141   36.319235   1212.704378
2007         Mexico  Americas   76.195  108.700891   11.977575   1301.973070
2007  United States  Americas   78.242  301.139947   42.951653  12934.458535
```

Q4-c: Compute the average `gdpPercap` of the countries in the Americas in 1952, and also in 2007. (No need to do population weighted average.)

```
[86]: data.query('continent=="Americas" and year==1952')['gdpPercap'].mean()
```

```
4.0790625522
```

```
[87]: data['gdpPercap'][(data['continent']=='Americas') & (data.index==1952)].mean()
```

```
4.0790625522
```

```
[88]: data.query('continent=="Americas" and year==2007')['gdpPercap'].mean()
```

```
11.00303162536
```

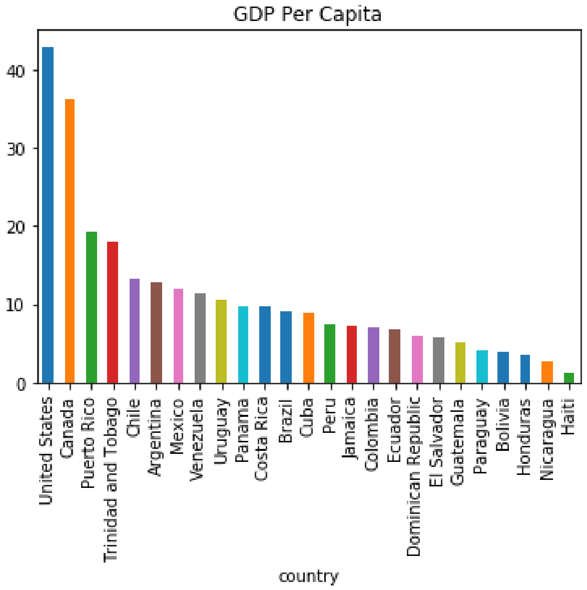
```
[89]: data['gdpPercap'][(data['continent']=='Americas') & (data.index==2007)].mean()
```

```
11.00303162536
```

Q4-d: Create a bar graphs of the gdpPercap of countries in the Americas for the year 2007. (Optional: sort the bars in descending order.)

```
[90]: data.query('continent=="Americas" and year==2007').\
          sort_values(by='gdpPercap',ascending=False).\
          plot(x='country',y='gdpPercap',kind='bar',legend=False,title='GDP Per Capita')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb02ec1f0b8>
```


GDP Per Capita
United States
Canada
Puerto Rico
Trinidad and Tobago
Chile
Argentina
Mexico
Venezuela
Uruguay
Panama
Costa Rica
Brazil
Cuba
Peru
Jamaica
Colombia
Ecuador
Dominican Republic
El Salvador
Guatemala
Paraguay
Bolivia
Honduras
Nicaragua
Haiti
country