Pandas Example

In [3]:
```
import pandas as pd
pd.Series?
```

In [4]:
```
Init signature:
pd.Series(
    data=None,
    index=None,
    dtype=None,
    name=None,
    copy=False,
    fastpath=False,
)
Docstring:
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
methods for performing operations involving the index. Statistical
methods from ndarray have been overridden to automatically exclude
missing data (currently represented as NaN).

Operations between Series (+, -, /, *, **) align values based on their
associated index values-- they need not be the same length. The result
index will be the sorted union of the two indexes.

Parameters
----------
data : array-like, Iterable, dict, or scalar value
    Contains data stored in Series.

    .. versionchanged:: 0.23.0
       If data is a dict, argument order is maintained for Python 3.6
       and later.

index : array-like or Index (1d)
    Values must be hashable and have the same length as `data`.
    Non-unique index values are allowed. Will default to
    RangeIndex (0, 1, 2, ..., n) if not provided. If both a dict and index
    sequence are used, the index will override the keys found in the
    dict.
dtype : str, numpy.dtype, or ExtensionDtype, optional
    Data type for the output Series. If not specified, this will be
    inferred from `data`.
    See the :ref:`user guide <basics.dtypes>` for more usages.
name : str, optional
    The name to give to the Series.
copy : bool, default False
    Copy input data.
File:           c:\users\jayvant\anaconda3\lib\site-packages\pandas\core\series.py
Type:           type
Subclasses:     SubclassedSeries
```

```
  File "<ipython-input-4-08f33bb2262c>", line 1
    Init signature:
                 ^
SyntaxError: invalid syntax
```

```
In [5]: animals =['Tiger','Bear','Moose']   #string series
        pd.Series(animals)

Out[5]: 0    Tiger
        1     Bear
        2    Moose
        dtype: object

In [6]: numbers =[1,2,3]     #integer series
        pd.Series(numbers)

Out[6]: 0    1
        1    2
        2    3
        dtype: int64

In [7]: animals =['Tiger','Bear',None]   #string series with string and None type (None  -
        string)
        pd.Series(animals)

Out[7]: 0    Tiger
        1     Bear
        2     None
        dtype: object

In [8]: numbers =[1,2,None]     #integer series and None type  (NaN - integer)
        pd.Series(numbers)

Out[8]: 0    1.0
        1    2.0
        2    NaN
        dtype: float64

In [9]: import numpy as np        #import numpy
        np.nan==None

Out[9]: False

In [10]: np.nan==1

Out[10]: False

In [11]: np.nan == np.nan

Out[11]: False

In [12]: np.isnan(np.nan)

Out[12]: True

In [13]: np.isnan(1)

Out[13]: False
```

```
In [15]: sports = {'Archery': 'Bhutan',    #key - value pair
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s=pd.Series(sports)
         s
```

```
Out[15]: Archery           Bhutan
         Golf            Scotland
         Sumo               Japan
         Taekwondo    South Korea
         dtype: object
```

```
In [16]: s.index   #returns index(key)
```

```
Out[16]: Index(['Archery', 'Golf', 'Sumo', 'Taekwondo'], dtype='object')
```

```
In [17]: s = pd.Series(['Tiger','Bear','Moose'],index=['India','America','Canada'])
         s
```

```
Out[17]: India      Tiger
         America     Bear
         Canada     Moose
         dtype: object
```

```
In [18]: sports = {'Archery': 'Bhutan',
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s = pd.Series(sports, index=['Golf', 'Sumo', 'Hockey'])   #display values of listed
         index
         s
```

```
Out[18]: Golf      Scotland
         Sumo         Japan
         Hockey         NaN
         dtype: object
```

Querying a Series

```
In [19]: sports = {'Archery': 'Bhutan',
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s = pd.Series(sports)
         s
```

```
Out[19]: Archery           Bhutan
         Golf            Scotland
         Sumo               Japan
         Taekwondo    South Korea
         dtype: object
```

```
In [20]: s.iloc[2]    #accessing using index number
```

```
Out[20]: 'Japan'
```

In [21]:
```
s.iloc[0]
```
Out[21]:
```
'Bhutan'
```

In [24]:
```
s.loc['Sumo']   #accessing using index values
```
Out[24]:
```
'Japan'
```

In [25]:
```
s.loc['Archery']
```
Out[25]:
```
'Bhutan'
```

In [26]:
```
s[2]  #index number
```
Out[26]:
```
'Japan'
```

In [28]:
```
s[0]
```
Out[28]:
```
'Bhutan'
```

In [30]:
```
s['Archery']
```
Out[30]:
```
'Bhutan'
```

In [31]:
```
sports = {99: 'Bhutan',
          100: 'Scotland',
          101: 'Japan',
          102: 'South Korea'}
s = pd.Series(sports)
s
```
Out[31]:
```
99          Bhutan
100       Scotland
101          Japan
102    South Korea
dtype: object
```

In [32]:
```
s.iloc[2]
```
Out[32]:
```
'Japan'
```

In [34]:
```
s.loc[101]
```
Out[34]:
```
'Japan'
```

In [36]:
```
s[101]
```
Out[36]:
```
'Japan'
```

In [38]:
```
s = pd.Series([100.00, 120.00, 101.00, 3.00])
s
```
Out[38]:
```
0    100.0
1    120.0
2    101.0
3      3.0
dtype: float64
```

In [39]:
```
total = 0
for i in s:
    total+=i
print(total)
```

```
324.0
```

In [40]:
```
import numpy as np
total = np.sum(s)
total
```

Out[40]:
```
324.0
```

In [41]:
```
max=np.max(s)
max
```

Out[41]:
```
120.0
```

In [42]:
```
s = pd.Series(np.random.randint(0,1000,10000))
s
```

Out[42]:
```
0       649
1       216
2       877
3       796
4       271
       ...
9995    572
9996    566
9997    573
9998    756
9999    380
Length: 10000, dtype: int32
```

In [43]:
```
s.head()
```

Out[43]:
```
0    649
1    216
2    877
3    796
4    271
dtype: int32
```

In [44]:
```
len(s)
```

Out[44]:
```
10000
```

In [45]:
```
%%timeit -n 100  #timeit - it will detects number of interations   (100 number of t
imes)
summary = 0
for item in s:
    summary+=item
```

```
3.38 ms ± 83.8 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [46]:
```
%%timeit -n 50
summary = 0
for item in s:
    summary+=item
```

```
3.19 ms ± 225 µs per loop (mean ± std. dev. of 7 runs, 50 loops each)
```

In [47]:
```
%%timeit -n 1000
summary = 0
for item in s:
    summary+=item
```
```
3.84 ms ± 13.7 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [51]:
```
%%timeit -n 100
summary = np.sum(s)
```
```
262 µs ± 22.4 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [49]:
```
%%timeit -n 50
summary = np.sum(s)
```
```
257 µs ± 17.2 µs per loop (mean ± std. dev. of 7 runs, 50 loops each)
```

In [58]:
```
s.head()
```
Out[58]:
```
0    653
1    220
2    881
3    800
4    275
dtype: int32
```

In [57]:
```
s+=2    #adds 2 in each item
s.head()
```
Out[57]:
```
0    653
1    220
2    881
3    800
4    275
dtype: int32
```

In [64]:
```
%%timeit -n 10
s = pd.Series(np.random.randint(0,1000,10000))
for label, value in s.iteritems():
    s.loc[label]= value+2
```
```
1.49 s ± 27.9 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [65]:
```
%%timeit -n 10
s = pd.Series(np.random.randint(0,1000,10000))
s+=2
```
```
1.17 ms ± 279 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [66]:
```
s = pd.Series([1, 2, 3])
s.loc['Animal'] = 'Bears'
s
```
Out[66]:
```
0             1
1             2
2             3
Animal    Bears
dtype: object
```

In [70]:
```
original_sports = pd.Series({'Archery': 'Bhutan',
                             'Golf': 'Scotland',
                             'Sumo': 'Japan',
                             'Taekwondo': 'South Korea'})
cricket_loving_countries = pd.Series(['Australia',
                                      'Barbados',
                                      'India',
                                      'England'],
                                   index=['Cricket',
                                          'Cricket',
                                          'Cricket',
                                          'Cricket'])
all_countries = original_sports.append(cricket_loving_countries)
all_countries
```

Out[70]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
Cricket        Australia
Cricket         Barbados
Cricket            India
Cricket          England
dtype: object
```

In [71]:
```
all_countries['Cricket']
```

Out[71]:
```
Cricket    Australia
Cricket     Barbados
Cricket        India
Cricket      England
dtype: object
```

In [72]:
```
all_countries[6]
```

Out[72]:
```
'India'
```

In [73]:
```
all_countries.loc['Cricket']
```

Out[73]:
```
Cricket    Australia
Cricket     Barbados
Cricket        India
Cricket      England
dtype: object
```

The DataFrame Data Structure

In [85]:
```
#creating Dataframe from various series
import pandas as pd
purchase_1 = pd.Series({'Name': 'Jayvant',
                        'Item Purchased': 'Dog Food',
                        'Cost': 22.50})
purchase_2 = pd.Series({'Name': 'Rahul',
                        'Item Purchased': 'Kitty Litter',
                        'Cost': 2.50})
purchase_3 = pd.Series({'Name': 'Vinod',
                        'Item Purchased': 'Bird Seed',
                        'Cost': 5.00})
df = pd.DataFrame([purchase_1, purchase_2, purchase_3], index=['Store 1', 'Store 1
', 'Store 2'])
df.head()
```

Out[85]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

In [86]:
```
df.head()
```

Out[86]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

In [104]:
```
df.head()
```

Out[104]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

In [105]:
```
df.loc['Store 2']
```

Out[105]:
```
Name                      Vinod
Item Purchased        Bird Seed
Cost                          5
Name: Store 2, dtype: object
```

In [106]:
```
type(df.loc['Store 2'])
```

Out[106]:
```
pandas.core.series.Series
```

In [107]:
```
df.loc['Store 1']
```

Out[107]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5

In [108]:
```
df.loc['Store 1', 'Cost']
```

Out[108]:
```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [109]:
```
df.loc['Store 1', 'Name']
```

Out[109]:
```
Store 1    Jayvant
Store 1      Rahul
Name: Name, dtype: object
```

In [110]:
```
df.T
```

Out[110]:

	Store 1	Store 1	Store 2
Name	Jayvant	Rahul	Vinod
Item Purchased	Dog Food	Kitty Litter	Bird Seed
Cost	22.5	2.5	5

In [111]:
```
df.head()
```

Out[111]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

In [112]:
```
df.T.loc['Cost']
```

Out[112]:
```
Store 1    22.5
Store 1     2.5
Store 2       5
Name: Cost, dtype: object
```

In [117]:
```
df.loc['Store 1', 'Name']
```

Out[117]:
```
Store 1    Jayvant
Store 1      Rahul
Name: Name, dtype: object
```

In [118]:
```
df.loc['Store 1', 'Cost']
```

Out[118]:
```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [119]:
```
df.loc['Store 1']
```

Out[119]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5

In [120]:
```
df.T
```
Out[120]:

	Store 1	Store 1	Store 2
Name	Jayvant	Rahul	Vinod
Item Purchased	Dog Food	Kitty Litter	Bird Seed
Cost	22.5	2.5	5

In [121]:
```
df.T.loc['Name']
```
Out[121]:
```
Store 1    Jayvant
Store 1      Rahul
Store 2      Vinod
Name: Name, dtype: object
```

In [122]:
```
df.loc[:,['Name', 'Cost']]
```
Out[122]:

	Name	Cost
Store 1	Jayvant	22.5
Store 1	Rahul	2.5
Store 2	Vinod	5.0

In [123]:
```
df.drop('Store 1')
```
Out[123]:

	Name	Item Purchased	Cost
Store 2	Vinod	Bird Seed	5.0

In [124]:
```
df
```
Out[124]:

	Name	Item Purchased	Cost
Store 1	Jayvant	Dog Food	22.5
Store 1	Rahul	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

In [125]:
```
copy_df = df.copy()
copy_df = copy_df.drop('Store 1')
copy_df
```
Out[125]:

	Name	Item Purchased	Cost
Store 2	Vinod	Bird Seed	5.0

In [126]:
```
del copy_df['Name']
copy_df
```
Out[126]:

	Item Purchased	Cost
Store 2	Bird Seed	5.0

In [127]:
```
df['Location'] = None   #add one more col value
df
```

Out[127]:

	Name	Item Purchased	Cost	Location
Store 1	Jayvant	Dog Food	22.5	None
Store 1	Rahul	Kitty Litter	2.5	None
Store 2	Vinod	Bird Seed	5.0	None

Dataframe Indexing and Loading

In [128]:
```
costs = df['Cost']
costs
```

Out[128]:
```
Store 1    22.5
Store 1     2.5
Store 2     5.0
Name: Cost, dtype: float64
```

In [129]:
```
costs+=2
costs
```

Out[129]:
```
Store 1    24.5
Store 1     4.5
Store 2     7.0
Name: Cost, dtype: float64
```

In [130]:
```
df
```

Out[130]:

	Name	Item Purchased	Cost	Location
Store 1	Jayvant	Dog Food	24.5	None
Store 1	Rahul	Kitty Litter	4.5	None
Store 2	Vinod	Bird Seed	7.0	None

In [131]:
```
!cat olympics.csv
```

```
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

In [133]:
```
df = pd.read_csv('F:/Python_Programs/olympics.csv')  #reading fromcsv file
df.head()
```

Out[133]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NaN	№ Summer	01 !	02 !	03 !	Total	№ Winter	01 !	02 !	03 !	Total	№ Games	01 !	02 !	03 !	Combined total
1	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
2	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
3	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
4	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12

In [134]:
```
df
```

Out[134]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	NaN	№ Summer	01 !	02 !	03 !	Total	№ Winter	01 !	02 !	03 !	Total	№ Games	01 !	02 !	(
1	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	
2	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	
3	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	
4	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
143	Independent Olympic Participants (IOP) [IOP]	1	0	1	2	3	0	0	0	0	0	1	0	1	
144	Zambia (ZAM) [ZAM]	12	0	1	1	2	0	0	0	0	0	12	0	1	
145	Zimbabwe (ZIM) [ZIM]	12	3	4	1	8	1	0	0	0	0	13	3	4	
146	Mixed team (ZZX) [ZZX]	3	8	5	4	17	0	0	0	0	0	3	8	5	
147	Totals	27	4809	4775	5130	14714	22	959	958	948	2865	49	5768	5733	6(

148 rows × 16 columns

In [137]:
```
df = pd.read_csv('F:/Python_Programs/olympics.csv', index_col = 0, skiprows=1) #re
move col 1 and row0
df.head()
```

Out[137]:

	№ Summer	01 !	02 !	03 !	Total	№ Winter	01 !.1	02 !.1	03 !.1	Total.1	№ Games	01 !.2	02 !.2	03 !.2	Combined total
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3	4	5	12

In [138]:
```
df.columns
```

Out[138]:
```
Index(['№ Summer', '01 !', '02 !', '03 !', 'Total', '№ Winter', '01 !.1',
       '02 !.1', '03 !.1', 'Total.1', '№ Games', '01 !.2', '02 !.2', '03 !.2',
       'Combined total'],
      dtype='object')
```

In [139]:
```
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)

df.head()
```

Out[139]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	G
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	

In [144]:
```
df.head()
```

Out[144]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	G
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	

Querying a DataFrame

In [145]:
```
df['Gold'] > 0
```

Out[145]:
```
Afghanistan (AFG)                               False
Algeria (ALG)                                    True
Argentina (ARG)                                  True
Armenia (ARM)                                    True
Australasia (ANZ) [ANZ]                          True
                                                ...
Independent Olympic Participants (IOP) [IOP]    False
Zambia (ZAM) [ZAM]                              False
Zimbabwe (ZIM) [ZIM]                             True
Mixed team (ZZX) [ZZX]                           True
Totals                                           True
Name: Gold, Length: 147, dtype: bool
```

In [146]:
```
only_gold = df.where(df['Gold'] > 0)
only_gold.head()
```

Out[146]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	G
Afghanistan (AFG)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
Algeria (ALG)	12.0	5.0	2.0	8.0	15.0	3.0	0.0	0.0	0.0	0.0	15.0	
Argentina (ARG)	23.0	18.0	24.0	28.0	70.0	18.0	0.0	0.0	0.0	0.0	41.0	
Armenia (ARM)	5.0	1.0	2.0	9.0	12.0	6.0	0.0	0.0	0.0	0.0	11.0	
Australasia (ANZ) [ANZ]	2.0	3.0	4.0	5.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	

In [147]:
```
only_gold
```

Out[147]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	Game
Afghanistan (AFG)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Algeria (ALG)	12.0	5.0	2.0	8.0	15.0	3.0	0.0	0.0	0.0	0.0	15.
Argentina (ARG)	23.0	18.0	24.0	28.0	70.0	18.0	0.0	0.0	0.0	0.0	41.
Armenia (ARM)	5.0	1.0	2.0	9.0	12.0	6.0	0.0	0.0	0.0	0.0	11.
Australasia (ANZ) [ANZ]	2.0	3.0	4.0	5.0	12.0	0.0	0.0	0.0	0.0	0.0	2.
...	...	...	...	...	...	...	...	...	...	...	.
Independent Olympic Participants (IOP) [IOP]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Zambia (ZAM) [ZAM]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
Zimbabwe (ZIM) [ZIM]	12.0	3.0	4.0	1.0	8.0	1.0	0.0	0.0	0.0	0.0	13.
Mixed team (ZZX) [ZZX]	3.0	8.0	5.0	4.0	17.0	0.0	0.0	0.0	0.0	0.0	3.
Totals	27.0	4809.0	4775.0	5130.0	14714.0	22.0	959.0	958.0	948.0	2865.0	49.

147 rows × 15 columns

In [148]:
```
only_gold['Gold'].count()
```

Out[148]:
```
100
```

In [149]:
```
df['Gold'].count()
```

Out[149]:
```
147
```

In [150]:
```
only_gold = only_gold.dropna()
only_gold.head()
```

Out[150]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	G
Algeria (ALG)	12.0	5.0	2.0	8.0	15.0	3.0	0.0	0.0	0.0	0.0	15.0	
Argentina (ARG)	23.0	18.0	24.0	28.0	70.0	18.0	0.0	0.0	0.0	0.0	41.0	
Armenia (ARM)	5.0	1.0	2.0	9.0	12.0	6.0	0.0	0.0	0.0	0.0	11.0	
Australasia (ANZ) [ANZ]	2.0	3.0	4.0	5.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	
Australia (AUS) [AUS] [Z]	25.0	139.0	152.0	177.0	468.0	18.0	5.0	3.0	4.0	12.0	43.0	

In [151]:
```
only_gold = df[df['Gold'] > 0]
only_gold.head()
```

Out[151]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gc
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	
Australia (AUS) [AUS] [Z]	25	139	152	177	468	18	5	3	4	12	43	

In [152]:
```
len(df[(df['Gold'] > 0) | (df['Gold.1'] > 0)])
```

Out[152]:
```
101
```

In [153]:
```
df[(df['Gold.1'] > 0) & (df['Gold'] == 0)]
```

Out[153]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	G
Liechtenstein (LIE)	16	0	0	0	0	18	2	2	5	9	34	

In [156]:
```
df[(df['Gold.2'] > 0) & (df['Gold'] == 3)]
```

Out[156]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gc
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	
Cameroon (CMR)	13	3	1	1	5	1	0	0	0	0	14	
Dominican Republic (DOM)	13	3	2	1	6	0	0	0	0	0	13	
Latvia (LAT)	10	3	11	5	19	10	0	4	3	7	20	
Nigeria (NGR)	15	3	8	12	23	0	0	0	0	0	15	
Pakistan (PAK)	16	3	3	4	10	2	0	0	0	0	18	
Tunisia (TUN)	13	3	3	4	10	0	0	0	0	0	13	
Zimbabwe (ZIM) [ZIM]	12	3	4	1	8	1	0	0	0	0	13	

Indexing Dataframes

In [157]:
```
df.head()
```

Out[157]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	G
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	

In [158]:
```
df['country'] = df.index
df = df.set_index('Gold')
df.head()
```

Out[158]:

	# Summer	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	Bro
Gold													
0	13	0	2	2	0	0	0	0	0	13	0	0	
5	12	2	8	15	3	0	0	0	0	15	5	2	
18	23	24	28	70	18	0	0	0	0	41	18	24	
1	5	2	9	12	6	0	0	0	0	11	1	2	
3	2	4	5	12	0	0	0	0	0	2	3	4	

In [159]:
```
df = df.reset_index()
df.head()
```

Out[159]:

	Gold	# Summer	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	E
0	0	13	0	2	2	0	0	0	0	0	13	0	0	
1	5	12	2	8	15	3	0	0	0	0	15	5	2	
2	18	23	24	28	70	18	0	0	0	0	41	18	24	
3	1	5	2	9	12	6	0	0	0	0	11	1	2	
4	3	2	4	5	12	0	0	0	0	0	2	3	4	

In [161]:
```
df = pd.read_csv('F:/Python_Programs/census.csv')
df.head()
```

Out[161]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE20
0	40	3	6	1	0	Alabama	Alabama	4779736	47801
1	50	3	6	1	1	Alabama	Autauga County	54571	545
2	50	3	6	1	3	Alabama	Baldwin County	182265	1822
3	50	3	6	1	5	Alabama	Barbour County	27457	274
4	50	3	6	1	7	Alabama	Bibb County	22915	229

5 rows × 100 columns

In [162]:
```
df['SUMLEV'].unique()
```

Out[162]:
```
array([40, 50], dtype=int64)
```

In [163]:
```
df['REGION'].unique()
```

Out[163]:
```
array([3, 4, 1, 2], dtype=int64)
```

In [164]:
```
df=df[df['SUMLEV'] == 50]
df.head()
```

Out[164]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE20
1	50	3	6	1	1	Alabama	Autauga County	54571	545
2	50	3	6	1	3	Alabama	Baldwin County	182265	1822
3	50	3	6	1	5	Alabama	Barbour County	27457	274
4	50	3	6	1	7	Alabama	Bibb County	22915	229
5	50	3	6	1	9	Alabama	Blount County	57322	573

5 rows × 100 columns

In [165]:
```python
columns_to_keep = ['STNAME',
                   'CTYNAME',
                   'BIRTHS2010',
                   'BIRTHS2011',
                   'BIRTHS2012',
                   'BIRTHS2013',
                   'BIRTHS2014',
                   'BIRTHS2015',
                   'POPESTIMATE2010',
                   'POPESTIMATE2011',
                   'POPESTIMATE2012',
                   'POPESTIMATE2013',
                   'POPESTIMATE2014',
                   'POPESTIMATE2015']
df = df[columns_to_keep]
df.head()
```

Out[165]:

	STNAME	CTYNAME	BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPI
1	Alabama	Autauga County	151	636	615	574	623	600	
2	Alabama	Baldwin County	517	2187	2092	2160	2186	2240	
3	Alabama	Barbour County	70	335	300	283	260	269	
4	Alabama	Bibb County	44	266	245	259	247	253	
5	Alabama	Blount County	183	744	710	646	618	603	

In [166]:
```python
df = df.set_index(['STNAME', 'CTYNAME'])
df.head()
```

Out[166]:

		BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPEST
STNAME	**CTYNAME**							
Alabama	**Autauga County**	151	636	615	574	623	600	
	Baldwin County	517	2187	2092	2160	2186	2240	
	Barbour County	70	335	300	283	260	269	
	Bibb County	44	266	245	259	247	253	
	Blount County	183	744	710	646	618	603	

In [167]:
```
df.loc['Michigan', 'Washtenaw County']
```

Out[167]:
```
BIRTHS2010             977
BIRTHS2011            3826
BIRTHS2012            3780
BIRTHS2013            3662
BIRTHS2014            3683
BIRTHS2015            3709
POPESTIMATE2010     345563
POPESTIMATE2011     349048
POPESTIMATE2012     351213
POPESTIMATE2013     354289
POPESTIMATE2014     357029
POPESTIMATE2015     358880
Name: (Michigan, Washtenaw County), dtype: int64
```

In [168]:
```
df.loc[ [('Michigan', 'Washtenaw County'),
         ('Michigan', 'Wayne County')] ]
```

Out[168]:

		BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPES
STNAME	**CTYNAME**							
Michigan	**Washtenaw County**	977	3826	3780	3662	3683	3709	
	Wayne County	5918	23819	23270	23377	23607	23586	

In [ ]: