*You are currently looking at **version 1.0** 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.*

The Series Data Structure

In [1]:

```
import pandas as pd
pd.Series?
```

In [2]:

```
animals = ['Tiger', 'Bear', 'Moose']
pd.Series(animals)
```

Out[2]:

```
0    Tiger
1     Bear
2    Moose
dtype: object
```

In [3]:

```
numbers = [1, 2, 3]
pd.Series(numbers)
```

Out[3]:

```
0    1
1    2
2    3
dtype: int64
```

In [4]:

```
animals = ['Tiger', 'Bear', None]
pd.Series(animals)
```

Out[4]:

```
0    Tiger
1     Bear
2     None
dtype: object
```

In [5]:

```
numbers = [1, 2, None]
pd.Series(numbers)
```

Out[5]:

```
0    1.0
1    2.0
2    NaN
dtype: float64
```

In [6]:

```
import numpy as np
np.nan == None
```

Out[6]:

```
False
```

In [7]:

```
np.nan == np.nan
```

Out[7]:

```
False
```

In [8]:

```
np.isnan(np.nan)
```

Out[8]:

```
True
```

In [9]:

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

Out[9]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [10]:

```
s.index
```

Out[10]:

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

In [11]:

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

Out[11]:

```
India      Tiger
America     Bear
Canada     Moose
dtype: object
```

In [12]:

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

Out[12]:

```
Sumo         Japan
Golf      Scotland
Hockey         NaN
dtype: object
```

Querying a Series

In [13]:

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

Out[13]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [14]:

```
s.iloc[3]
```

Out[14]:

```
'South Korea'
```

In [15]:

```
s.loc['Golf']
```

Out[15]:

```
'Scotland'
```

In [16]:

```
s[3]
```

Out[16]:

```
'South Korea'
```

In [17]:

```
s['Golf']
```

Out[17]:

```
'Scotland'
```

In [19]:

```
sports = {99: 'Bhutan',
          100: 'Scotland',
          101: 'Japan',
          102: 'South Korea'}
s = pd.Series(sports)
```

In [20]:

```
s[0] #This won't call s.iloc[0] as one might expect, it generates an error instead
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-20-a5f43d492595> in <module>()
----> 1 s[0] #This won't call s.iloc[0] as one might expect, it generates
 an error instead

/opt/conda/lib/python3.5/site-packages/pandas/core/series.py in __getitem_
_(self, key)
    581         key = com._apply_if_callable(key, self)
    582         try:
--> 583             result = self.index.get_value(self, key)
    584 
    585             if not lib.isscalar(result):

/opt/conda/lib/python3.5/site-packages/pandas/indexes/base.py in
get_value(self, series, key)
   1978         try:
   1979             return self._engine.get_value(s, k,
-> 1980                                           tz=getattr(series.dtype,
 'tz', None))
   1981         except KeyError as e1:
   1982             if len(self) > 0 and self.inferred_type in ['integer',
 'boolean']:

pandas/index.pyx in pandas.index.IndexEngine.get_value (pandas/index.c:333
2)()

pandas/index.pyx in pandas.index.IndexEngine.get_value (pandas/index.c:303
5)()

pandas/index.pyx in pandas.index.IndexEngine.get_loc (pandas/index.c:4018)
()

pandas/hashtable.pyx in pandas.hashtable.Int64HashTable.get_item (pandas/h
ashtable.c:6610)()

pandas/hashtable.pyx in pandas.hashtable.Int64HashTable.get_item (pandas/h
ashtable.c:6554)()

KeyError: 0
```

In [21]:

```
s = pd.Series([100.00, 120.00, 101.00, 3.00])
s
```

Out[21]:

```
0    100.0
1    120.0
2    101.0
3      3.0
dtype: float64
```

In [22]:

```
total = 0
for item in s:
    total+=item
print(total)
```

```
324.0
```

In [23]:

```
import numpy as np

total = np.sum(s)
print(total)
```

```
324.0
```

In [24]:

```
#this creates a big series of random numbers
s = pd.Series(np.random.randint(0,1000,10000))
s.head()
```

Out[24]:

```
0    817
1    604
2    533
3     90
4    548
dtype: int64
```

In [25]:

```
len(s)
```

Out[25]:

```
10000
```

In [26]:

```
%%timeit -n 10
summary = 0
for item in s:
    summary+=item
```

```
10 loops, best of 3: 3.08 ms per loop
```

In [27]:

```
%%timeit -n 10
summary = np.sum(s)
```

```
10 loops, best of 3: 161 µs per loop
```

In [28]:

```
s+=2 #adds two to each item in s using broadcasting
s.head()
```

Out[28]:

```
0    819
1    606
2    535
3     92
4    550
dtype: int64
```

In [29]:

```
for label, value in s.iteritems():
    s.set_value(label, value+2)
s.head()
```

Out[29]:

```
0    821
1    608
2    537
3     94
4    552
dtype: int64
```

In [30]:

```
%%timeit -n 10
s = pd.Series(np.random.randint(0,100,1000))
for label, value in s.iteritems():
    s.loc[label]= value+2
```

```
10 loops, best of 3: 158 ms per loop
```

In [31]:

```
%%timeit -n 10
s = pd.Series(np.random.randint(0,100,1000))
s+=2
```

```
10 loops, best of 3: 341 µs per loop
```

In [32]:

```
s = pd.Series([1, 2, 3])
s.loc['Animal'] = 'Bears'
s
```

Out[32]:

```
0             1
1             2
2             3
Animal    Bears
dtype: object
```

In [33]:

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

In [34]:

```
original_sports
```

Out[34]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [35]:

```
cricket_loving_countries
```

Out[35]:

```
Cricket    Australia
Cricket     Barbados
Cricket     Pakistan
Cricket      England
dtype: object
```

In [36]:

```
all_countries
```

Out[36]:

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
Cricket        Australia
Cricket         Barbados
Cricket         Pakistan
Cricket          England
dtype: object
```

In [37]:

```
all_countries.loc['Cricket']
```

Out[37]:

```
Cricket    Australia
Cricket     Barbados
Cricket     Pakistan
Cricket      England
dtype: object
```

The DataFrame Data Structure

In [38]:

```
import pandas as pd
purchase_1 = pd.Series({'Name': 'Chris',
                        'Item Purchased': 'Dog Food',
                        'Cost': 22.50})
purchase_2 = pd.Series({'Name': 'Kevyn',
                        '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', 'S
tore 2'])
df.head()
```

Out[38]:

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

In [39]:

```
df.loc['Store 2']
```

Out[39]:

```
Cost                      5
Item Purchased    Bird Seed
Name                  Vinod
Name: Store 2, dtype: object
```

In [40]:

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

Out[40]:

```
pandas.core.series.Series
```

In [41]:

```
df.loc['Store 1']
```

Out[41]:

	Cost	Item Purchased	Name
Store 1	22.5	Dog Food	Chris
Store 1	2.5	Kitty Litter	Kevyn

In [42]:

```
df.loc['Store 1','Cost']
```

Out[42]:

```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [43]:

```
df.T
```

Out[43]:

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

In [44]:

```
df.T.loc['Cost']
```

Out[44]:

```
Store 1    22.5
Store 1     2.5
Store 2       5
Name: Cost, dtype: object
```

In [45]:

```
df['Cost'] = df['Cost']*.8
df['Cost']
```

Out[45]:

```
Store 1    18.0
Store 1     2.0
Store 2     4.0
Name: Cost, dtype: float64
```

In [46]:

```
df.loc['Store 1']['Cost']
```

Out[46]:

```
Store 1    18.0
Store 1     2.0
Name: Cost, dtype: float64
```

In [47]:

```
df.loc[:,['Name', 'Cost']]
```

Out[47]:

	Name	Cost
Store 1	Chris	18.0
Store 1	Kevyn	2.0
Store 2	Vinod	4.0

In [48]:

```
df.drop('Store 1')
```

Out[48]:

	Cost	Item Purchased	Name
Store 2	4.0	Bird Seed	Vinod

In [49]:

```
df
```

Out[49]:

	Cost	Item Purchased	Name
Store 1	18.0	Dog Food	Chris
Store 1	2.0	Kitty Litter	Kevyn
Store 2	4.0	Bird Seed	Vinod

In [50]:

```
copy_df = df.copy()
copy_df = copy_df.drop('Store 1')
copy_df
```

Out[50]:

	Cost	Item Purchased	Name
Store 2	4.0	Bird Seed	Vinod

In [51]:

```
copy_df.drop?
```

In [52]:

```
del copy_df['Name']
copy_df
```

Out[52]:

	Cost	Item Purchased
Store 2	4.0	Bird Seed

In [55]:

```
df['Location'] = None
df
```

Out[55]:

	Cost	Item Purchased	Name	Location
Store 1	18.0	Dog Food	Chris	None
Store 1	2.0	Kitty Litter	Kevyn	None
Store 2	4.0	Bird Seed	Vinod	None

Dataframe Indexing and Loading

In [56]:

```
costs = df['Cost']
costs
```

Out[56]:

```
Store 1    18.0
Store 1     2.0
Store 2     4.0
Name: Cost, dtype: float64
```

In [57]:

```
costs+=2
costs
```

Out[57]:

```
Store 1    20.0
Store 1     4.0
Store 2     6.0
Name: Cost, dtype: float64
```

In [58]:

```
df
```

Out[58]:

	Cost	Item Purchased	Name	Location
Store 1	20.0	Dog Food	Chris	None
Store 1	4.0	Kitty Litter	Kevyn	None
Store 2	6.0	Bird Seed	Vinod	None

In [59]:

```
!cat olympics.csv
```

```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
,№ Summer,01 !,02 !,03 !,Total,№ Winter,01 !,02 !,03 !,Total,№ Games,01 !,
02 !,03 !,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
Australia (AUS) [AUS] [Z],25,139,152,177,468,18,5,3,4,12,43,144,155,181,48
0
Austria (AUT),26,18,33,35,86,22,59,78,81,218,48,77,111,116,304
Azerbaijan (AZE),5,6,5,15,26,5,0,0,0,0,10,6,5,15,26
Bahamas (BAH),15,5,2,5,12,0,0,0,0,0,15,5,2,5,12
Bahrain (BRN),8,0,0,1,1,0,0,0,0,0,8,0,0,1,1
Barbados (BAR) [BAR],11,0,0,1,1,0,0,0,0,0,11,0,0,1,1
Belarus (BLR),5,12,24,39,75,6,6,4,5,15,11,18,28,44,90
Belgium (BEL),25,37,52,53,142,20,1,1,3,5,45,38,53,56,147
Bermuda (BER),17,0,0,1,1,7,0,0,0,0,24,0,0,1,1
Bohemia (BOH) [BOH] [Z],3,0,1,3,4,0,0,0,0,0,3,0,1,3,4
Botswana (BOT),9,0,1,0,1,0,0,0,0,0,9,0,1,0,1
Brazil (BRA),21,23,30,55,108,7,0,0,0,0,28,23,30,55,108
British West Indies (BWI) [BWI],1,0,0,2,2,0,0,0,0,0,1,0,0,2,2
Bulgaria (BUL) [H],19,51,85,78,214,19,1,2,3,6,38,52,87,81,220
Burundi (BDI),5,1,0,0,1,0,0,0,0,0,5,1,0,0,1
Cameroon (CMR),13,3,1,1,5,1,0,0,0,0,14,3,1,1,5
Canada (CAN),25,59,99,121,279,22,62,56,52,170,47,121,155,173,449
Chile (CHI) [I],22,2,7,4,13,16,0,0,0,0,38,2,7,4,13
China (CHN) [CHN],9,201,146,126,473,10,12,22,19,53,19,213,168,145,526
Colombia (COL),18,2,6,11,19,1,0,0,0,0,19,2,6,11,19
Costa Rica (CRC),14,1,1,2,4,6,0,0,0,0,20,1,1,2,4
Ivory Coast (CIV) [CIV],12,0,1,0,1,0,0,0,0,0,12,0,1,0,1
Croatia (CRO),6,6,7,10,23,7,4,6,1,11,13,10,13,11,34
Cuba (CUB) [Z],19,72,67,70,209,0,0,0,0,0,19,72,67,70,209
Cyprus (CYP),9,0,1,0,1,10,0,0,0,0,19,0,1,0,1
Czech Republic (CZE) [CZE],5,14,15,15,44,6,7,9,8,24,11,21,24,23,68
Czechoslovakia (TCH) [TCH],16,49,49,45,143,16,2,8,15,25,32,51,57,60,168
Denmark (DEN) [Z],26,43,68,68,179,13,0,1,0,1,39,43,69,68,180
Djibouti (DJI) [B],7,0,0,1,1,0,0,0,0,0,7,0,0,1,1
Dominican Republic (DOM),13,3,2,1,6,0,0,0,0,0,13,3,2,1,6
Ecuador (ECU),13,1,1,0,2,0,0,0,0,0,13,1,1,0,2
Egypt (EGY) [EGY] [Z],21,7,9,10,26,1,0,0,0,0,22,7,9,10,26
Eritrea (ERI),4,0,0,1,1,0,0,0,0,0,4,0,0,1,1
Estonia (EST),11,9,9,15,33,9,4,2,1,7,20,13,11,16,40
Ethiopia (ETH),12,21,7,17,45,2,0,0,0,0,14,21,7,17,45
Finland (FIN),24,101,84,117,302,22,42,62,57,161,46,143,146,174,463
France (FRA) [O] [P] [Z],27,202,223,246,671,22,31,31,47,109,49,233,254,29
3,780
Gabon (GAB),9,0,1,0,1,0,0,0,0,0,9,0,1,0,1
Georgia (GEO),5,6,5,14,25,6,0,0,0,0,11,6,5,14,25
Germany (GER) [GER] [Z],15,174,182,217,573,11,78,78,53,209,26,252,260,270,
782
United Team of Germany (EUA) [EUA],3,28,54,36,118,3,8,6,5,19,6,36,60,41,13
7
East Germany (GDR) [GDR],5,153,129,127,409,6,39,36,35,110,11,192,165,162,5
19
West Germany (FRG) [FRG],5,56,67,81,204,6,11,15,13,39,11,67,82,94,243
Ghana (GHA) [GHA],13,0,1,3,4,1,0,0,0,0,14,0,1,3,4
Great Britain (GBR) [GBR] [Z],27,236,272,272,780,22,10,4,12,26,49,246,276,
284,806
Greece (GRE) [Z],27,30,42,39,111,18,0,0,0,0,45,30,42,39,111
Grenada (GRN),8,1,0,0,1,0,0,0,0,0,8,1,0,0,1
```

```
Guatemala (GUA),13,0,1,0,1,1,0,0,0,0,14,0,1,0,1
Guyana (GUY) [GUY],16,0,0,1,1,0,0,0,0,0,16,0,0,1,1
Haiti (HAI) [J],14,0,1,1,2,0,0,0,0,0,14,0,1,1,2
Hong Kong (HKG) [HKG],15,1,1,1,3,4,0,0,0,0,19,1,1,1,3
Hungary (HUN),25,167,144,165,476,22,0,2,4,6,47,167,146,169,482
Iceland (ISL),19,0,2,2,4,17,0,0,0,0,36,0,2,2,4
India (IND) [F],23,9,6,11,26,9,0,0,0,0,32,9,6,11,26
Indonesia (INA),14,6,10,11,27,0,0,0,0,0,14,6,10,11,27
Iran (IRI) [K],15,15,20,25,60,10,0,0,0,0,25,15,20,25,60
Iraq (IRQ),13,0,0,1,1,0,0,0,0,0,13,0,0,1,1
Ireland (IRL),20,9,8,12,29,6,0,0,0,0,26,9,8,12,29
Israel (ISR),15,1,1,5,7,6,0,0,0,0,21,1,1,5,7
Italy (ITA) [M] [S],26,198,166,185,549,22,37,34,43,114,48,235,200,228,663
Jamaica (JAM) [JAM],16,17,30,20,67,7,0,0,0,0,23,17,30,20,67
Japan (JPN),21,130,126,142,398,20,10,17,18,45,41,140,143,160,443
Kazakhstan (KAZ),5,16,17,19,52,6,1,3,3,7,11,17,20,22,59
Kenya (KEN),13,25,32,29,86,3,0,0,0,0,16,25,32,29,86
North Korea (PRK),9,14,12,21,47,8,0,1,1,2,17,14,13,22,49
South Korea (KOR),16,81,82,80,243,17,26,17,10,53,33,107,99,90,296
Kuwait (KUW),12,0,0,2,2,0,0,0,0,0,12,0,0,2,2
Kyrgyzstan (KGZ),5,0,1,2,3,6,0,0,0,0,11,0,1,2,3
Latvia (LAT),10,3,11,5,19,10,0,4,3,7,20,3,15,8,26
Lebanon (LIB),16,0,2,2,4,16,0,0,0,0,32,0,2,2,4
Liechtenstein (LIE),16,0,0,0,0,18,2,2,5,9,34,2,2,5,9
Lithuania (LTU),8,6,5,10,21,8,0,0,0,0,16,6,5,10,21
Luxembourg (LUX) [O],22,1,1,0,2,8,0,2,0,2,30,1,3,0,4
Macedonia (MKD),5,0,0,1,1,5,0,0,0,0,10,0,0,1,1
Malaysia (MAS) [MAS],12,0,3,3,6,0,0,0,0,0,12,0,3,3,6
Mauritius (MRI),8,0,0,1,1,0,0,0,0,0,8,0,0,1,1
Mexico (MEX),22,13,21,28,62,8,0,0,0,0,30,13,21,28,62
Moldova (MDA),5,0,2,5,7,6,0,0,0,0,11,0,2,5,7
Mongolia (MGL),12,2,9,13,24,13,0,0,0,0,25,2,9,13,24
Montenegro (MNE),2,0,1,0,1,2,0,0,0,0,4,0,1,0,1
Morocco (MAR),13,6,5,11,22,6,0,0,0,0,19,6,5,11,22
Mozambique (MOZ),9,1,0,1,2,0,0,0,0,0,9,1,0,1,2
Namibia (NAM),6,0,4,0,4,0,0,0,0,0,6,0,4,0,4
Netherlands (NED) [Z],25,77,85,104,266,20,37,38,35,110,45,114,123,139,376
Netherlands Antilles (AHO) [AHO] [I],13,0,1,0,1,2,0,0,0,0,15,0,1,0,1
New Zealand (NZL) [NZL],22,42,18,39,99,15,0,1,0,1,37,42,19,39,100
Niger (NIG),11,0,0,1,1,0,0,0,0,0,11,0,0,1,1
Nigeria (NGR),15,3,8,12,23,0,0,0,0,0,15,3,8,12,23
Norway (NOR) [Q],24,56,49,43,148,22,118,111,100,329,46,174,160,143,477
Pakistan (PAK),16,3,3,4,10,2,0,0,0,0,18,3,3,4,10
Panama (PAN),16,1,0,2,3,0,0,0,0,0,16,1,0,2,3
Paraguay (PAR),11,0,1,0,1,1,0,0,0,0,12,0,1,0,1
Peru (PER) [L],17,1,3,0,4,2,0,0,0,0,19,1,3,0,4
Philippines (PHI),20,0,2,7,9,4,0,0,0,0,24,0,2,7,9
Poland (POL),20,64,82,125,271,22,6,7,7,20,42,70,89,132,291
Portugal (POR),23,4,8,11,23,7,0,0,0,0,30,4,8,11,23
Puerto Rico (PUR),17,0,2,6,8,6,0,0,0,0,23,0,2,6,8
Qatar (QAT),8,0,0,4,4,0,0,0,0,0,8,0,0,4,4
Romania (ROU),20,88,94,119,301,20,0,0,0,1,1,40,88,94,120,302
Russia (RUS) [RUS],5,132,121,142,395,6,49,40,35,124,11,181,161,177,519
Russian Empire (RU1) [RU1],3,1,4,3,8,0,0,0,0,0,3,1,4,3,8
Soviet Union (URS) [URS],9,395,319,296,1010,9,78,57,59,194,18,473,376,355,
1204
Unified Team (EUN) [EUN],1,45,38,29,112,1,9,6,8,23,2,54,44,37,135
Saudi Arabia (KSA),10,0,1,2,3,0,0,0,0,0,10,0,1,2,3
Senegal (SEN),13,0,1,0,1,5,0,0,0,0,18,0,1,0,1
Serbia (SRB) [SRB],3,1,2,4,7,2,0,0,0,0,5,1,2,4,7
Serbia and Montenegro (SCG) [SCG],3,2,4,3,9,3,0,0,0,0,6,2,4,3,9
```

```
Singapore (SIN),15,0,2,2,4,0,0,0,0,0,15,0,2,2,4
Slovakia (SVK) [SVK],5,7,9,8,24,6,2,2,1,5,11,9,11,9,29
Slovenia (SLO),6,4,6,9,19,7,2,4,9,15,13,6,10,18,34
South Africa (RSA),18,23,26,27,76,6,0,0,0,0,24,23,26,27,76
Spain (ESP) [Z],22,37,59,35,131,19,1,0,1,2,41,38,59,36,133
Sri Lanka (SRI) [SRI],16,0,2,0,2,0,0,0,0,0,16,0,2,0,2
Sudan (SUD),11,0,1,0,1,0,0,0,0,0,11,0,1,0,1
Suriname (SUR) [E],11,1,0,1,2,0,0,0,0,0,11,1,0,1,2
Sweden (SWE) [Z],26,143,164,176,483,22,50,40,54,144,48,193,204,230,627
Switzerland (SUI),27,47,73,65,185,22,50,40,48,138,49,97,113,113,323
Syria (SYR),12,1,1,1,3,0,0,0,0,0,12,1,1,1,3
Chinese Taipei (TPE) [TPE] [TPE2],13,2,7,12,21,11,0,0,0,0,24,2,7,12,21
Tajikistan (TJK),5,0,1,2,3,4,0,0,0,0,9,0,1,2,3
Tanzania (TAN) [TAN],12,0,2,0,2,0,0,0,0,0,12,0,2,0,2
Thailand (THA),15,7,6,11,24,3,0,0,0,0,18,7,6,11,24
Togo (TOG),9,0,0,1,1,1,0,0,0,0,10,0,0,1,1
Tonga (TGA),8,0,1,0,1,1,0,0,0,0,9,0,1,0,1
Trinidad and Tobago (TRI) [TRI],16,2,5,11,18,3,0,0,0,0,19,2,5,11,18
Tunisia (TUN),13,3,3,4,10,0,0,0,0,0,13,3,3,4,10
Turkey (TUR),21,39,25,24,88,16,0,0,0,0,37,39,25,24,88
Uganda (UGA),14,2,3,2,7,0,0,0,0,0,14,2,3,2,7
Ukraine (UKR),5,33,27,55,115,6,2,1,4,7,11,35,28,59,122
United Arab Emirates (UAE),8,1,0,0,1,0,0,0,0,0,8,1,0,0,1
United States (USA) [P] [Q] [R] [Z],26,976,757,666,2399,22,96,102,84,282,4
8,1072,859,750,2681
Uruguay (URU),20,2,2,6,10,1,0,0,0,0,21,2,2,6,10
Uzbekistan (UZB),5,5,5,10,20,6,1,0,0,1,11,6,5,10,21
Venezuela (VEN),17,2,2,8,12,4,0,0,0,0,21,2,2,8,12
Vietnam (VIE),14,0,2,0,2,0,0,0,0,0,14,0,2,0,2
Virgin Islands (ISV),11,0,1,0,1,7,0,0,0,0,18,0,1,0,1
Yugoslavia (YUG) [YUG],16,26,29,28,83,14,0,3,1,4,30,26,32,29,87
Independent Olympic Participants (IOP) [IOP],1,0,1,2,3,0,0,0,0,0,1,0,1,2,3
Zambia (ZAM) [ZAM],12,0,1,1,2,0,0,0,0,0,12,0,1,1,2
Zimbabwe (ZIM) [ZIM],12,3,4,1,8,1,0,0,0,0,13,3,4,1,8
Mixed team (ZZX) [ZZX],3,8,5,4,17,0,0,0,0,0,3,8,5,4,17
Totals,27,4809,4775,5130,14714,22,959,958,948,2865,49,5768,5733,6078,17579
```

In [60]:

```
df = pd.read_csv('olympics.csv')
df.head()
```

Out[60]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	1
0	NaN	№ Summer	01 !	02 !	03 !	Total	№ Winter	01 !	02 !	03 !	Total	№ Games	01 !	0 !
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	2
4	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2

In [61]:

```
df = pd.read_csv('olympics.csv', index_col = 0, skiprows=1)
df.head()
```

Out[61]:

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

In [62]:

```
df.columns
```

Out[62]:

```
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 [63]:

```
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[63]:

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

Querying a DataFrame

In [64]:

```
df['Gold'] > 0
```

```
Out[64]:

Afghanistan (AFG)                               False
Algeria (ALG)                                    True
Argentina (ARG)                                  True
Armenia (ARM)                                    True
Australasia (ANZ) [ANZ]                          True
Australia (AUS) [AUS] [Z]                        True
Austria (AUT)                                    True
Azerbaijan (AZE)                                 True
Bahamas (BAH)                                    True
Bahrain (BRN)                                   False
Barbados (BAR) [BAR]                            False
Belarus (BLR)                                    True
Belgium (BEL)                                    True
Bermuda (BER)                                   False
Bohemia (BOH) [BOH] [Z]                         False
Botswana (BOT)                                  False
Brazil (BRA)                                     True
British West Indies (BWI) [BWI]                 False
Bulgaria (BUL) [H]                               True
Burundi (BDI)                                    True
Cameroon (CMR)                                   True
Canada (CAN)                                     True
Chile (CHI) [I]                                  True
China (CHN) [CHN]                                True
Colombia (COL)                                   True
Costa Rica (CRC)                                 True
Ivory Coast (CIV) [CIV]                         False
Croatia (CRO)                                    True
Cuba (CUB) [Z]                                   True
Cyprus (CYP)                                    False
                                                ...  
Sri Lanka (SRI) [SRI]                           False
Sudan (SUD)                                     False
Suriname (SUR) [E]                               True
Sweden (SWE) [Z]                                 True
Switzerland (SUI)                                True
Syria (SYR)                                      True
Chinese Taipei (TPE) [TPE] [TPE2]                True
Tajikistan (TJK)                                False
Tanzania (TAN) [TAN]                            False
Thailand (THA)                                   True
Togo (TOG)                                      False
Tonga (TGA)                                     False
Trinidad and Tobago (TRI) [TRI]                  True
Tunisia (TUN)                                    True
Turkey (TUR)                                     True
Uganda (UGA)                                     True
Ukraine (UKR)                                    True
United Arab Emirates (UAE)                       True
United States (USA) [P] [Q] [R] [Z]              True
Uruguay (URU)                                    True
Uzbekistan (UZB)                                 True
Venezuela (VEN)                                  True
Vietnam (VIE)                                   False
Virgin Islands (ISV)                            False
Yugoslavia (YUG) [YUG]                           True
Independent Olympic Participants (IOP) [IOP]    False
Zambia (ZAM) [ZAM]                              False
Zimbabwe (ZIM) [ZIM]                             True
Mixed team (ZZX) [ZZX]                           True
Mixed team (ZZX) [ZZX]                          True

Totals                                          True
Name: Gold, dtype: bool
```

In [65]:

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

Out[65]:

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

In [66]:

```
only_gold['Gold'].count()
```

Out[66]:

```
100
```

In [67]:

```
df['Gold'].count()
```

Out[67]:

```
147
```

In [68]:

```
only_gold = only_gold.dropna()
only_gold.head()
```

Out[68]:

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

In [69]:

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

Out[69]:

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

In [70]:

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

Out[70]:

```
101
```

In [71]:

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

Out[71]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bro
Liechtenstein (LIE)	16	0	0	0	0	18	2	2	5

Indexing Dataframes

In [72]:

```
df.head()
```

Out[72]:

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

In [73]:

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

Out[73]:

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

In [74]:

```
df = df.reset_index()
df.head()
```

Out[74]:

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

In [75]:

```
df = pd.read_csv('census.csv')
df.head()
```

Out[75]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010I
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

In [80]:

```
df['SUMLEV'].unique()
```

Out[80]:

```
array([40, 50])
```

In [81]:

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

Out[81]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010I
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 [82]:

```
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[82]:

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

In [83]:

```
df = df.set_index(['STNAME', 'CTYNAME'])
df.head()
```

Out[83]:

		BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2
STNAME	**CTYNAME**					
Alabama	**Autauga County**	151	636	615	574	623
	Baldwin County	517	2187	2092	2160	2186
	Barbour County	70	335	300	283	260
	Bibb County	44	266	245	259	247
	Blount County	183	744	710	646	618

In [84]:

```
df.loc['Michigan', 'Washtenaw County']
```

Out[84]:

```
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 [85]:

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

Out[85]:

		BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTH
STNAME	CTYNAME					
Michigan	Washtenaw County	977	3826	3780	3662	3683
	Wayne County	5918	23819	23270	23377	23607

Missing values

In [86]:

```
df = pd.read_csv('log.csv')
df
```

```
Out[86]:
```

	time	user	video	playback position	paused	volume
0	1469974424	cheryl	intro.html	5	False	10.0
1	1469974454	cheryl	intro.html	6	NaN	NaN
2	1469974544	cheryl	intro.html	9	NaN	NaN
3	1469974574	cheryl	intro.html	10	NaN	NaN
4	1469977514	bob	intro.html	1	NaN	NaN
5	1469977544	bob	intro.html	1	NaN	NaN
6	1469977574	bob	intro.html	1	NaN	NaN
7	1469977604	bob	intro.html	1	NaN	NaN
8	1469974604	cheryl	intro.html	11	NaN	NaN
9	1469974694	cheryl	intro.html	14	NaN	NaN
10	1469974724	cheryl	intro.html	15	NaN	NaN
11	1469974454	sue	advanced.html	24	NaN	NaN
12	1469974524	sue	advanced.html	25	NaN	NaN
13	1469974424	sue	advanced.html	23	False	10.0
14	1469974554	sue	advanced.html	26	NaN	NaN
15	1469974624	sue	advanced.html	27	NaN	NaN
16	1469974654	sue	advanced.html	28	NaN	5.0
17	1469974724	sue	advanced.html	29	NaN	NaN
18	1469974484	cheryl	intro.html	7	NaN	NaN
19	1469974514	cheryl	intro.html	8	NaN	NaN
20	1469974754	sue	advanced.html	30	NaN	NaN
21	1469974824	sue	advanced.html	31	NaN	NaN
22	1469974854	sue	advanced.html	32	NaN	NaN
23	1469974924	sue	advanced.html	33	NaN	NaN
24	1469977424	bob	intro.html	1	True	10.0
25	1469977454	bob	intro.html	1	NaN	NaN
26	1469977484	bob	intro.html	1	NaN	NaN
27	1469977634	bob	intro.html	1	NaN	NaN
28	1469977664	bob	intro.html	1	NaN	NaN
29	1469974634	cheryl	intro.html	12	NaN	NaN
30	1469974664	cheryl	intro.html	13	NaN	NaN
31	1469977694	bob	intro.html	1	NaN	NaN
32	1469977724	bob	intro.html	1	NaN	NaN

In [87]:

```
df.fillna?
```