*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
```

In [2]:
```
animals = ['Tiger', 'Bear', 'Moose']
```

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

In [3]:
```
numbers = [1, 2, 3]
```

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

In [4]:
```
animals = ['Tiger', 'Bear', None]
```

Out[4]:
```
0    Tiger
1     Bear
2     None
dtype: object
```

In [5]:
```
numbers = [1, 2, None]
```

Out[5]:
```
0    1.0
1    2.0
2    NaN
dtype: float64
```

In [6]:
```
import numpy as np
```

Out[6]:
```
False
```

In [7]:
```

```

Out[7]:
```
False
```

In [8]:
```

```

Out[8]:
```
True
```

In [9]:
```
sports = {'Archery': 'Bhutan',
          'Golf': 'Scotland',
          'Sumo': 'Japan',
          'Taekwondo': 'South Korea'}
s = pd.Series(sports)
print(s)
```

```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
{'Archery': 'Bhutan', 'Golf': 'Scotland', 'Sumo': 'Japan', 'Taekwondo': 'South K
orea'}
```

In [10]:

Out[10]: Index(['Archery', 'Golf', 'Sumo', 'Taekwondo'], dtype='object')

In [11]:
```
s = pd.Series(['Tiger', 'Bear', 'Moose'], index=['India', 'America', 'Canada'])
```

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=['Golf', 'Sumo', 'Hockey'])
```

Out[12]:
```
Golf      Scotland
Sumo         Japan
Hockey         NaN
dtype: object
```

Querying a Series

In [13]:
```
sports = {'Archery': 'Bhutan',
          'Golf': 'Scotland',
          'Sumo': 'Japan',
          'Taekwondo': 'South Korea'}
s = pd.Series(sports)
```

Out[13]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [14]:

Out[14]: 'South Korea'

In [15]:

Out[15]: 'Scotland'

In [16]:

Out[16]: 'South Korea'

In [17]:

Out[17]: 'Scotland'

In [18]:

Out[18]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [19]:
```
sports = {99: 'Bhutan',
          100: 'Scotland',
          101: 'Japan',
          102: 'South Korea'}
```

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

/opt/conda/lib/python3.6/site-packages/pandas/core/series.py in __getitem__(sel
f, key)
    601         key = com._apply_if_callable(key, self)
    602         try:
--> 603             result = self.index.get_value(self, key)
    604
    605             if not is_scalar(result):

/opt/conda/lib/python3.6/site-packages/pandas/indexes/base.py in get_value(self,
series, key)
   2167         try:
   2168             return self._engine.get_value(s, k,
-> 2169                                           tz=getattr(series.dtype, 'tz',
```

In [21]:
```
s = pd.Series([100.00, 120.00, 101.00, 3.00])
```

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
```
```
324.0
```

In [23]:
```
import numpy as np

total = np.sum(s)
```
```
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     30
1     50
2    621
3    603
4    406
dtype: int64
```

In [25]:
```
```

Out[25]:
```
10000
```

In [26]:
```
%%timeit -n 10
summary = 0
for item in s:
```

```
The slowest run took 7.23 times longer than the fastest. This could mean that an
intermediate result is being cached.
1.91 ms ± 1.88 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [27]:
```
%%timeit -n 100
```

```
The slowest run took 6.95 times longer than the fastest. This could mean that an
intermediate result is being cached.
165 µs ± 181 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [28]:
```
s+=2 #adds two to each item in s using broadcasting
```

Out[28]:
```
0     32
1     52
2    623
3    605
4    408
dtype: int64
```

In [29]:
```
for label, value in s.iteritems():
    s.set_value(label, value+2)
```

Out[29]:
```
0     34
1     54
2    625
3    607
4    410
dtype: int64
```

In [30]:
```
%%timeit -n 10
s = pd.Series(np.random.randint(0,1000,10000))
for label, value in s.iteritems():
```

```
1.23 s ± 18.3 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [31]:
```
%%timeit -n 10
s = pd.Series(np.random.randint(0,1000,10000))
s+=2
```

```
The slowest run took 27.51 times longer than the fastest. This could mean that a
n intermediate result is being cached.
1.13 ms ± 2.09 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [32]:
```
s = pd.Series([1, 2, 3])
s.loc['Animal'] = 'Bears'
```

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'])
```

In [34]:

Out[34]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [35]:

Out[35]:
```
Cricket    Australia
Cricket     Barbados
Cricket     Pakistan
Cricket      England
dtype: object
```

In [36]:

Out[36]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
Cricket        Australia
Cricket         Barbados
Cricket         Pakistan
Cricket          England
dtype: object
```

In [37]:

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'
```

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]:

Out[39]:
```
Cost                          5
Item Purchased    Bird Seed
Name                    Vinod
Name: Store 2, dtype: object
```

In [40]:

Out[40]:
```
pandas.core.series.Series
```

In [41]:

Out[41]:

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

In [42]:

Out[42]:
```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [43]:

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]:

Out[44]:
```
Store 1    22.5
Store 1     2.5
Store 2       5
Name: Cost, dtype: object
```

In [45]:

Out[45]:
```
Store 1    22.5
Store 1     2.5
Store 2     5.0
Name: Cost, dtype: float64
```

In [46]:

Out[46]:
```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

In [47]:

Out[47]:

	Name	Cost
Store 1	Chris	22.5
Store 1	Kevyn	2.5
Store 2	Vinod	5.0

In [48]:

Out[48]:

	Cost	Item Purchased	Name
Store 2	5.0	Bird Seed	Vinod

In [49]:

Out[49]:

	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 [50]:
```
copy_df = df.copy()
copy_df = copy_df.drop('Store 1')
```

Out[50]:

	Cost	Item Purchased	Name
Store 2	5.0	Bird Seed	Vinod

In [51]:

In [52]:
```
del copy_df['Name']
```

Out[52]:

	Cost	Item Purchased
Store 2	5.0	Bird Seed

In [53]:
```
df['Location'] = None
```

Out[53]:

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

Dataframe Indexing and Loading

In [54]:
```
costs = df['Cost']
```

Out[54]:
```
Store 1    22.5
Store 1     2.5
Store 2     5.0
Name: Cost, dtype: float64
```

In [55]:
```
costs+=2
```

Out[55]:
```
Store 1    24.5
Store 1     4.5
Store 2     7.0
Name: Cost, dtype: float64
```

In [56]:
```

```

Out[56]:

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

In [57]:
```

```

```
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 !,0
3 !,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],25,139,152,177,468,18,5,3,4,12,43,144,155,181,480
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
```

In [58]:
```
import pandas as pd
df = pd.read_csv('olympics.csv')
```

Out[58]:

	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 [59]:
```
df = pd.read_csv('olympics.csv', index_col = 0, skiprows=1)
```

Out[59]:

	№ 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 [60]:
```

```

Out[60]:
```
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 [61]:
```
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)
```

Out[61]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Go
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 [62]:
```
for col in df.columns:
```

```
mmer

er
ze
l
nter
.1
er.1
ze.1
l.1
mes
.2
er.2
ze.2
ined total
```

Querying a DataFrame

In [63]:

Out[63]:

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
Totals	True

Name: Gold, dtype: bool

In [64]:
```
only_gold = df.where(df['Gold'] > 0)
```

Out[64]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Go
Afghanistan (AFG)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
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	1
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 [65]:

Out[65]: 100

In [66]:

Out[66]: 147

In [67]:
```
only_gold = only_gold.dropna()
```

Out[67]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gol
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	1
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	14

In [68]:
```
only_gold = df[df['Gold'] > 0]
```

Out[68]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gol
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	1

In [69]:

Out[69]: 101

In [70]:

Out[70]:

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

Indexing Dataframes

In [71]:

Out[71]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Go
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 [72]:
```
df['country'] = df.index
df = df.set_index('Gold')
```

Out[72]:

Gold	# Summer	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	Bron
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 [73]:
```
df = df.reset_index()
```

Out[73]:

	Gold	# Summer	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	Br
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 [74]:
```
import pandas as pd
df = pd.read_csv('census.csv')
```

Out[74]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2010
0	40	3	6	1	0	Alabama	Alabama	4779736	4780127
1	50	3	6	1	1	Alabama	Autauga County	54571	54571
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265
3	50	3	6	1	5	Alabama	Barbour County	27457	27457
4	50	3	6	1	7	Alabama	Bibb County	22915	22919

5 rows × 100 columns

In [75]:
```
df['SUMLEV'].unique()
```

Out[75]:
```
array([40, 50])
```

In [76]:
```
df=df[df['SUMLEV'] == 50]
```

Out[76]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2010
1	50	3	6	1	1	Alabama	Autauga County	54571	54571
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265
3	50	3	6	1	5	Alabama	Barbour County	27457	27457
4	50	3	6	1	7	Alabama	Bibb County	22915	22919
5	50	3	6	1	9	Alabama	Blount County	57322	57322

5 rows × 100 columns

In [77]:
```
columns_to_keep = ['STNAME',
                   'CTYNAME',
                   'BIRTHS2010',
                   'BIRTHS2011',
                   'BIRTHS2012',
                   'BIRTHS2013',
                   'BIRTHS2014',
                   'BIRTHS2015',
                   'POPESTIMATE2010',
                   'POPESTIMATE2011',
                   'POPESTIMATE2012',
                   'POPESTIMATE2013',
                   'POPESTIMATE2014',
                   'POPESTIMATE2015']
df = df[columns_to_keep]
```

Out[77]:

	STNAME	CTYNAME	BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPES
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 [78]:
```
df = df.set_index(['STNAME', 'CTYNAME'])
```

Out[78]:

		BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPESTIM
STNAME	**CTYNAME**							
	Autauga County	151	636	615	574	623	600	
	Baldwin County	517	2187	2092	2160	2186	2240	
Alabama	**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 [79]:

Out[79]:

STNAME	CTYNAME	BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPESTIM
	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	
	Bullock County	39	169	122	132	118	123	
	Butler County	65	276	241	240	267	257	

In [80]:

Out[80]:
```
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 [81]:
```
df.loc[ [('Michigan', 'Washtenaw County'),
```

Out[81]:

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

In [82]:
```
import pandas as pd
purchase1 = pd.Series({'Name':'Bhavesh',
                       'Item purchased':'doog food',
                       'cost':22.50})
purchase2 = pd.Series({'Name':'Kiran',
                       'Item purchased':'cat litter',
                       'cost':2.50})
purchase3 = pd.Series({'Name':'Nitin',
                       'Item purchased':'bird seeds',
                       'cost':5.00})
df = pd.DataFrame([purchase1, purchase2, purchase3], index=['Store 1', 'Store 1', '
#df["Location"] = df.index
#df = df.reset_index()
#df = df.set_index(["Location", "Name"])
#df["Location", "Name", "Item purchased", "cost"] = ["Store 2", "Kevyn", "Kitty foo
#df
df = df.set_index([df.index, 'Name'])
df.index.names = ['Location', 'Name']
#df = df.append(pd.Series(data={'cost': 3.00, 'Item purchased': 'Kitty Food'}, name
df = df.append(pd.Series({'Location':'Store 2',
                          'Name':'Kevyin',
                         'Item purchased': 'Cat food',
                         'cost': 3.00}))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-82-1d0d967c19e9> in <module>()
     21                           'Name':'Kevyin',
     22                          'Item purchased': 'Cat food',
---> 23                          'cost': 3.00}))
     24 df

/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in append(self, othe
r, ignore_index, verify_integrity)
   4403                 other = Series(other)
   4404             if other.name is None and not ignore_index:
-> 4405                 raise TypeError('Can only append a Series if ignore_inde
x=True'
   4406                                 ' or if the Series has a name')
   4407

TypeError: Can only append a Series if ignore_index=True or if the Series has a
name
```

Missing values

In [83]:
```
import pandas as pd
df = pd.read_csv('log.csv')
```

Out[83]:

	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 [ ]:

In [84]:
```
df = df.set_index('time')
df = df.sort_index()
```

Out[84]:

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

In [85]:
```
df = df.reset_index()
df = df.set_index(['time', 'user'])
```

Out[85]:

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

In [86]:
```
df = df.fillna(method='ffill')
```

Out[86]:

time	user	video	playback position	paused	volume
1469974424	cheryl	intro.html	5	False	10.0
	sue	advanced.html	23	False	10.0
1469974454	cheryl	intro.html	6	False	10.0
	sue	advanced.html	24	False	10.0
1469974484	cheryl	intro.html	7	False	10.0