Week 2

May 5, 2020

---

*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 course resource.*

---

1 The Series Data Structure

Pandas `Series([data, index, dtype, name, copy, ...])` is a constructor for Series. Its a one-dimensional ndarray with axis labels (including time series). Series contains:**Attributes** like `series.index,series.size`.... **Conversions** , **Iteration/Indexing** like `series.loc,series.iloc,series.iteritems()`... **Computation/Stat**, **Missing Data Handling**... and many more.

```
In [39]: import pandas as pd

         #Here '?' will give a documentation of the Data structure, Methods, Attribute we using
         pd.Series.loc?
```

Here series gives an one-dimensional ndarray with axis labels(0,1,2) (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`).

```
In [40]: animals = ['Tiger', 'Bear', 'Moose']
         pd.Series(animals)

Out[40]: 0    Tiger
         1     Bear
         2    Moose
         dtype: object

In [41]: numbers = [1, 2, 3]
         pd.Series(numbers)
```

```
Out[41]: 0    1
         1    2
         2    3
         dtype: int64

In [42]: animals = ['Tiger', 'Bear', None]
         pd.Series(animals)

Out[42]: 0    Tiger
         1     Bear
         2     None
         dtype: object

In [43]: numbers = [1, 2, None]
         pd.Series(numbers)

Out[43]: 0    1.0
         1    2.0
         2    NaN
         dtype: float64
```

Within pandas, a missing value is denoted by **NaN**. **Series.isna()**: Detect missing values. Return a boolean same-sized object indicating if the values are `NA`. NA values, such as None or **numpy.NaN**, gets mapped to `True` values. Everything else gets mapped to `False` values. Characters such as empty strings `''` or `numpy.inf` are not considered NA values (unless you set `pandas.options.mode.use_inf_as_na = True`).

```
In [44]: import numpy as np
         np.nan == None

Out[44]: False

In [45]: np.nan == np.nan

Out[45]: False

In [46]: np.isnan(np.nan)

Out[46]: True
```

pandas.Series *(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)* **data** : array-like, Iterable, dict, or scalar value**index** : array-like or Index (1d). Values must be hashable and have the same length as data. Non-unique index values are allowed

```
In [47]: sports = {'Archery': 'Bhutan',
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s = pd.Series(sports)
         s
```

```
Out[47]: Archery           Bhutan
         Golf            Scotland
         Sumo               Japan
         Taekwondo    South Korea
         dtype: object

In [48]: s.index             # index is an attribute of series

Out[48]: Index(['Archery', 'Golf', 'Sumo', 'Taekwondo'], dtype='object')

In [49]: s = pd.Series(['Tiger', 'Bear', 'Moose'], index=['India', 'America', 'Canada'])
         s

Out[49]: India      Tiger
         America     Bear
         Canada     Moose
         dtype: object

In [50]: sports = {'Archery': 'Bhutan',
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s = pd.Series(sports, index=['Golf', 'Sumo', 'Hockey'])
         s

Out[50]: Golf      Scotland
         Sumo         Japan
         Hockey         NaN
         dtype: object
```

2 Querying a Series

```
In [51]: sports = {'Archery': 'Bhutan',
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s = pd.Series(sports)
         s

Out[51]: Archery           Bhutan
         Golf            Scotland
         Sumo               Japan
         Taekwondo    South Korea
         dtype: object
```

Series.loc Property : Access a group of rows and columns by label(s) or a boolean array.
.loc[ ] is primarily label based, but may also be used with a boolean array.
Series.iloc Property : Purely integer-location based indexing for selection by position.
.iloc[ ] is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array.

```
In [52]: s.iloc[3]

Out[52]: 'South Korea'

In [53]: s.loc['Golf']

Out[53]: 'Scotland'

In [54]: s[3]

Out[54]: 'South Korea'

In [55]: s['Golf']

Out[55]: 'Scotland'

In [56]: sports = {99: 'Bhutan',
                   100: 'Scotland',
                   101: 'Japan',
                   102: 'South Korea'}
         s = pd.Series(sports)
         s

Out[56]: 99          Bhutan
         100       Scotland
         101          Japan
         102    South Korea
         dtype: object

In [57]: s[0] #This won't call s.iloc[0] as one might expect, it generates an error instead


         ---------------------------------------------------------------------------

         KeyError                                  Traceback (most recent call last)

         <ipython-input-57-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/python3.6/site-packages/pandas/core/series.py in __getitem__(self, 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,
        2167         try:
```

```
   2168             return self._engine.get_value(s, k,
-> 2169                                           tz=getattr(series.dtype, 'tz', None))
   2170         except KeyError as e1:
   2171             if len(self) > 0 and self.inferred_type in ['integer', 'boolean']:


pandas/index.pyx in pandas.index.IndexEngine.get_value (pandas/index.c:3557)()


pandas/index.pyx in pandas.index.IndexEngine.get_value (pandas/index.c:3240)()


pandas/index.pyx in pandas.index.IndexEngine.get_loc (pandas/index.c:4279)()


pandas/src/hashtable_class_helper.pxi in pandas.hashtable.Int64HashTable.get_item (panda


pandas/src/hashtable_class_helper.pxi in pandas.hashtable.Int64HashTable.get_item (panda


KeyError: 0
```

```
In [59]: s = pd.Series([100.00, 120.00, 101.00, 3.00])
         s

Out[59]: 0    100.0
         1    120.0
         2    101.0
         3      3.0
         dtype: float64

In [60]: total = 0
         for item in s:
             total+=item
         print(total)
```

```
324.0
```

```
In [61]: import numpy as np

         total = np.sum(s)                                    #using sum() operator it take less
         print(total)
```

```
324.0
```

```
In [62]: #this creates a big series of random numbers
         s = pd.Series(np.random.randint(0,1000,10000))
         s.head()                                          #.head() will return only  the to

Out[62]: 0    195
         1    780
         2    516
         3    820
         4    694
         dtype: int64

In [63]: len(s)

Out[63]: 10000

In [64]: %%timeit -n 100
         summary = 0
         for item in s:
             summary+=item

2.14 ms ś 343 ţs per loop (mean ś std. dev. of 7 runs, 100 loops each)


In [65]: %%timeit -n 100
         summary = np.sum(s)

The slowest run took 7.32 times longer than the fastest. This could mean that an intermediate re
266 ţs ś 258 ţs per loop (mean ś std. dev. of 7 runs, 100 loops each)


In [66]: s+=2                                      #adds two to each item in s using broadcast
         s.head()

Out[66]: 0    197
         1    782
         2    518
         3    822
         4    696
         dtype: int64

In [ ]: for label, value in s.iteritems():#Lazily iterate over (index, value) tuples.This method
             s.set_value(label, value+2)
        s.head()

Out[ ]: 0    199
        1    784
        2    520
        3    824
        4    698
        dtype: int64
```

```
In [ ]: %%timeit -n 10
        s = pd.Series(np.random.randint(0,1000,10000))
        for label, value in s.iteritems():
            s.loc[label]= value+2
```

```
1.47 s ś 6.57 ms per loop (mean ś std. dev. of 7 runs, 10 loops each)
```

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

```
378 ţs ś 59.4 ţs per loop (mean ś std. dev. of 7 runs, 10 loops each)
```

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

```
Out[ ]: 0             1
        1             2
        2             3
        Animal    Bears
        dtype: object
```

```
In [ ]: 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 [ ]: original_sports
```

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

```
In [ ]: cricket_loving_countries
```

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

In [ ]: all_countries

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

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

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

3 The DataFrame Data Structure

pandas.DataFrame *(data=None, index: Optional[Collection] = None, columns: Optional[Collection] = None, dtype: Union[str, numpy.dtype, ExtensionDtype, None] = None, copy: bool = False)*

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

- **data** : ndarray (structured or homogeneous), Iterable, dict, or DataFrame. Dict can contain Series, arrays, constants, or list-like objects.
- **index** : Index or array-like. Index to use for resulting frame. Will default to RangeIndex if no indexing information part of input data and no index provided.
- **columns** : Index or array-like. Column labels to use for resulting frame.

```
In [ ]: 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', 'St
        df.head()
```

Out[ ]:

	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 [ ]: df.loc['Store 2']                              #Single label. Note this returns the row
```

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

```
In [ ]: type(df.loc['Store 2'])                    # return datatype
```

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

```
In [ ]: df.loc[['Store 1','Store 2']]                     #List of labels. Note using [[]] retur
```

Out[ ]:

	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 [ ]: df.loc['Store 1', 'Cost']                              # Single label for row and colum
```

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

```
In [ ]: df.T                                                  # Transpose index and columns.
```

Out[ ]:

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

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

```
In [ ]: df['Cost']

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

In [ ]: df.loc['Store 1']['Cost']

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

- Slice with labels for row and single label for column. As mentioned above, `note that both the start and stop of the slice are included.`

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

Out[ ]:           Name  Cost
        Store 1  Chris  22.5
        Store 1  Kevyn   2.5
        Store 2  Vinod   5.0
```

- **DataFrame.drop** *(self, labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')* Drop specified labels from rows or columns.

Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names. When using a multi-index, labels on different levels can be removed by specifying the level.

```
In [ ]: df.drop('Store 1')                                    # Drop specified labels from rows or c

Out[ ]:          Cost Item Purchased   Name
        Store 2   5.0      Bird Seed  Vinod

In [ ]: df

Out[ ]:          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 [ ]: copy_df = df.copy()                                   #Make a copy of this objects indices an
        copy_df = copy_df.drop('Store 1')
        copy_df

Out[ ]:          Cost Item Purchased   Name
        Store 2   5.0      Bird Seed  Vinod

In [ ]: copy_df.drop?
```

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

Out[ ]:          Cost Item Purchased
        Store 2   5.0      Bird Seed

In [ ]: df['Location'] = None                # new column with vale 'None'
        df

Out[ ]:          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
```

4 Dataframe Indexing and Loading

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

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

- Here we can increase the cost in this series using **broadcasting**. Now if we look at our original DataFrame, we see those costs have risen as well.

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

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

In [ ]: df

Out[ ]:          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
```

- Pandas has built-in support for delimited files such as `CSV files` as well as a variety of other data formats including `relational databases, Excel, and HTML tables.`
- We can take a look at this file using the **shell command cat**. Which we can invoke directly using the `exclamation` point.

```
In [ ]: !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
Afghanistană(AFG),13,0,0,2,2,0,0,0,0,0,13,0,0,2,2
Algeriaă(ALG),12,5,2,8,15,3,0,0,0,0,15,5,2,8,15
Argentinaă(ARG),23,18,24,28,70,18,0,0,0,0,41,18,24,28,70
Armeniaă(ARM),5,1,2,9,12,6,0,0,0,0,11,1,2,9,12
Australasiaă(ANZ) [ANZ],2,3,4,5,12,0,0,0,0,0,2,3,4,5,12
Australiaă(AUS) [AUS] [Z],25,139,152,177,468,18,5,3,4,12,43,144,155,181,480
Austriaă(AUT),26,18,33,35,86,22,59,78,81,218,48,77,111,116,304
Azerbaijană(AZE),5,6,5,15,26,5,0,0,0,0,10,6,5,15,26
Bahamasă(BAH),15,5,2,5,12,0,0,0,0,0,15,5,2,5,12
Bahrainăă(BRN),8,0,0,1,1,0,0,0,0,0,8,0,0,1,1
Barbadosă(BAR) [BAR],11,0,0,1,1,0,0,0,0,0,11,0,0,1,1
Belarusă(BLR),5,12,24,39,75,6,6,4,5,15,11,18,28,44,90
Belgiumă(BEL),25,37,52,53,142,20,1,1,3,5,45,38,53,56,147
Bermudaă(BER),17,0,0,1,1,7,0,0,0,0,24,0,0,1,1
Bohemiaă(BOH) [BOH] [Z],3,0,1,3,4,0,0,0,0,0,3,0,1,3,4
Botswanaă(BOT),9,0,1,0,1,0,0,0,0,0,9,0,1,0,1
Braziliă(BRA),21,23,30,55,108,7,0,0,0,0,28,23,30,55,108
British West Indiesă(BWI) [BWI],1,0,0,2,2,0,0,0,0,0,1,0,0,2,2
Bulgariaă(BUL) [H],19,51,85,78,214,19,1,2,3,6,38,52,87,81,220
Burundiă(BDI),5,1,0,0,1,0,0,0,0,0,5,1,0,0,1
Cameroonă(CMR),13,3,1,1,5,1,0,0,0,0,14,3,1,1,5
Canadaă(CAN),25,59,99,121,279,22,62,56,52,170,47,121,155,173,449
Chileă(CHI) [I],22,2,7,4,13,16,0,0,0,0,38,2,7,4,13
Chinaă(CHN) [CHN],9,201,146,126,473,10,12,22,19,53,19,213,168,145,526
Colombiaă(COL),18,2,6,11,19,1,0,0,0,0,19,2,6,11,19
Costa Ricaă(CRC),14,1,1,2,4,6,0,0,0,0,20,1,1,2,4
Ivory Coastă(CIV) [CIV],12,0,1,0,1,0,0,0,0,0,12,0,1,0,1
Croatiaă(CRO),6,6,7,10,23,7,4,6,1,11,13,10,13,11,34
Cubaă(CUB) [Z],19,72,67,70,209,0,0,0,0,0,19,72,67,70,209
Cyprusă(CYP),9,0,1,0,1,10,0,0,0,0,19,0,1,0,1
Czech Republicăă(CZE) [CZE],5,14,15,15,44,6,7,9,8,24,11,21,24,23,68
Czechoslovakiaă(TCH) [TCH],16,49,49,45,143,16,2,8,15,25,32,51,57,60,168
Denmarkă(DEN) [Z],26,43,68,68,179,13,0,1,0,1,39,43,69,68,180
Djiboutiă(DJI) [B],7,0,0,1,1,0,0,0,0,0,7,0,0,1,1
Dominican Republicăă(DOM),13,3,2,1,6,0,0,0,0,0,13,3,2,1,6
Ecuadorăă(ECU),13,1,1,0,2,0,0,0,0,0,13,1,1,0,2
Egyptă(EGY) [EGY] [Z],21,7,9,10,26,1,0,0,0,0,22,7,9,10,26
Eritreaă(ERI),4,0,0,1,1,0,0,0,0,0,4,0,0,1,1
Estoniaă(EST),11,9,9,15,33,9,4,2,1,7,20,13,11,16,40
Ethiopiaă(ETH),12,21,7,17,45,2,0,0,0,0,14,21,7,17,45
Finlandăă(FIN),24,101,84,117,302,22,42,62,57,161,46,143,146,174,463
Franceă(FRA) [O] [P] [Z],27,202,223,246,671,22,31,31,47,109,49,233,254,293,780
Gabonăă(GAB),9,0,1,0,1,0,0,0,0,0,9,0,1,0,1
Georgiaă(GEO),5,6,5,14,25,6,0,0,0,0,11,6,5,14,25
Germanyă(GER) [GER] [Z],15,174,182,217,573,11,78,78,53,209,26,252,260,270,782
United Team of Germanyă(EUA) [EUA],3,28,54,36,118,3,8,6,5,19,6,36,60,41,137
```

East Germanyă(GDR) [GDR],5,153,129,127,409,6,39,36,35,110,11,192,165,162,519
West Germanyă(FRG) [FRG],5,56,67,81,204,6,11,15,13,39,11,67,82,94,243
Ghanaă(GHA) [GHA],13,0,1,3,4,1,0,0,0,0,14,0,1,3,4
Great Britainăă(GBR) [GBR] [Z],27,236,272,272,780,22,10,4,12,26,49,246,276,284,806
Greeceă(GRE) [Z],27,30,42,39,111,18,0,0,0,0,45,30,42,39,111
Grenadaă(GRN),8,1,0,0,1,0,0,0,0,0,8,1,0,0,1
Guatemalaă(GUA),13,0,1,0,1,1,0,0,0,0,14,0,1,0,1
Guyanaă(GUY) [GUY],16,0,0,1,1,0,0,0,0,0,16,0,0,1,1
Haitiă(HAI) [J],14,0,1,1,2,0,0,0,0,0,14,0,1,1,2
Hong Kongă(HKG) [HKG],15,1,1,1,3,4,0,0,0,0,19,1,1,1,3
Hungaryă(HUN),25,167,144,165,476,22,0,2,4,6,47,167,146,169,482
Icelandă(ISL),19,0,2,2,4,17,0,0,0,0,36,0,2,2,4
Indiaă(IND) [F],23,9,6,11,26,9,0,0,0,0,32,9,6,11,26
Indonesiaă(INA),14,6,10,11,27,0,0,0,0,0,14,6,10,11,27
Iranăă(IRI) [K],15,15,20,25,60,10,0,0,0,0,25,15,20,25,60
Iraqă(IRQ),13,0,0,1,1,0,0,0,0,0,13,0,0,1,1
Irelandă(IRL),20,9,8,12,29,6,0,0,0,0,26,9,8,12,29
Israelă(ISR),15,1,1,5,7,6,0,0,0,0,21,1,1,5,7
Italyă(ITA) [M] [S],26,198,166,185,549,22,37,34,43,114,48,235,200,228,663
Jamaicaă(JAM) [JAM],16,17,30,20,67,7,0,0,0,0,23,17,30,20,67
Japanăă(JPN),21,130,126,142,398,20,10,17,18,45,41,140,143,160,443
Kazakhstanăă(KAZ),5,16,17,19,52,6,1,3,3,7,11,17,20,22,59
Kenyaă(KEN),13,25,32,29,86,3,0,0,0,0,16,25,32,29,86
North Koreaă(PRK),9,14,12,21,47,8,0,1,1,2,17,14,13,22,49
South Koreaă(KOR),16,81,82,80,243,17,26,17,10,53,33,107,99,90,296
Kuwaităă(KUW),12,0,0,2,2,0,0,0,0,0,12,0,0,2,2
Kyrgyzstanăă(KGZ),5,0,1,2,3,6,0,0,0,0,11,0,1,2,3
Latviaă(LAT),10,3,11,5,19,10,0,4,3,7,20,3,15,8,26
Lebanonăă(LIB),16,0,2,2,4,16,0,0,0,0,32,0,2,2,4
Liechtensteinăă(LIE),16,0,0,0,0,18,2,2,5,9,34,2,2,5,9
Lithuaniaă(LTU),8,6,5,10,21,8,0,0,0,0,16,6,5,10,21
Luxembourgăă(LUX) [O],22,1,1,0,2,8,0,2,0,2,30,1,3,0,4
Macedoniaă(MKD),5,0,0,1,1,5,0,0,0,0,10,0,0,1,1
Malaysiaă(MAS) [MAS],12,0,3,3,6,0,0,0,0,0,12,0,3,3,6
Mauritiusăă(MRI),8,0,0,1,1,0,0,0,0,0,8,0,0,1,1
Mexicoă(MEX),22,13,21,28,62,8,0,0,0,0,30,13,21,28,62
Moldovaă(MDA),5,0,2,5,7,6,0,0,0,0,11,0,2,5,7
Mongoliaă(MGL),12,2,9,13,24,13,0,0,0,0,25,2,9,13,24
Montenegroă(MNE),2,0,1,0,1,2,0,0,0,0,4,0,1,0,1
Moroccoă(MAR),13,6,5,11,22,6,0,0,0,0,19,6,5,11,22
Mozambiqueă(MOZ),9,1,0,1,2,0,0,0,0,0,9,1,0,1,2
Namibiaă(NAM),6,0,4,0,4,0,0,0,0,0,6,0,4,0,4
Netherlandsăă(NED) [Z],25,77,85,104,266,20,37,38,35,110,45,114,123,139,376
Netherlands Antillesăă(AHO) [AHO] [I],13,0,1,0,1,2,0,0,0,0,15,0,1,0,1
New Zealandăă(NZL) [NZL],22,42,18,39,99,15,0,1,0,1,37,42,19,39,100
Nigeră(NIG),11,0,0,1,1,0,0,0,0,0,11,0,0,1,1
Nigeriaă(NGR),15,3,8,12,23,0,0,0,0,0,15,3,8,12,23
Norwayă(NOR) [Q],24,56,49,43,148,22,118,111,100,329,46,174,160,143,477

Pakistană(PAK),16,3,3,4,10,2,0,0,0,0,18,3,3,4,10
Panamaă(PAN),16,1,0,2,3,0,0,0,0,0,16,1,0,2,3
Paraguayă(PAR),11,0,1,0,1,1,0,0,0,0,12,0,1,0,1
Peruă(PER) [L],17,1,3,0,4,2,0,0,0,0,19,1,3,0,4
Philippinesă(PHI),20,0,2,7,9,4,0,0,0,0,24,0,2,7,9
Polandă(POL),20,64,82,125,271,22,6,7,7,20,42,70,89,132,291
Portugală(POR),23,4,8,11,23,7,0,0,0,0,30,4,8,11,23
Puerto Ricoă(PUR),17,0,2,6,8,6,0,0,0,0,23,0,2,6,8
Qatară(QAT),8,0,0,4,4,0,0,0,0,0,8,0,0,4,4
Romaniaă(ROU),20,88,94,119,301,20,0,0,1,1,40,88,94,120,302
Russiaă(RUS) [RUS],5,132,121,142,395,6,49,40,35,124,11,181,161,177,519
Russian Empireă(RU1) [RU1],3,1,4,3,8,0,0,0,0,0,3,1,4,3,8
Soviet Unionă(URS) [URS],9,395,319,296,1010,9,78,57,59,194,18,473,376,355,1204
Unified Teamă(EUN) [EUN],1,45,38,29,112,1,9,6,8,23,2,54,44,37,135
Saudi Arabiaă(KSA),10,0,1,2,3,0,0,0,0,0,10,0,1,2,3
Senegală(SEN),13,0,1,0,1,5,0,0,0,0,18,0,1,0,1
Serbiaă(SRB) [SRB],3,1,2,4,7,2,0,0,0,0,5,1,2,4,7
Serbia and Montenegroă(SCG) [SCG],3,2,4,3,9,3,0,0,0,0,6,2,4,3,9
Singaporeă(SIN),15,0,2,2,4,0,0,0,0,0,15,0,2,2,4
Slovakiaă(SVK) [SVK],5,7,9,8,24,6,2,2,1,5,11,9,11,9,29
Sloveniaă(SLO),6,4,6,9,19,7,2,4,9,15,13,6,10,18,34
South Africaă(RSA),18,23,26,27,76,6,0,0,0,0,24,23,26,27,76
Spainăă(ESP) [Z],22,37,59,35,131,19,1,0,1,2,41,38,59,36,133
Sri Lankaă(SRI) [SRI],16,0,2,0,2,0,0,0,0,0,16,0,2,0,2
Sudană(SUD),11,0,1,0,1,0,0,0,0,0,11,0,1,0,1
Surinameă(SUR) [E],11,1,0,1,2,0,0,0,0,0,11,1,0,1,2
Swedenă(SWE) [Z],26,143,164,176,483,22,50,40,54,144,48,193,204,230,627
Switzerlandă(SUI),27,47,73,65,185,22,50,40,48,138,49,97,113,113,323
Syriaă(SYR),12,1,1,1,3,0,0,0,0,0,12,1,1,1,3
Chinese Taipeiă(TPE) [TPE] [TPE2],13,2,7,12,21,11,0,0,0,0,24,2,7,12,21
Tajikistană(TJK),5,0,1,2,3,4,0,0,0,0,9,0,1,2,3
Tanzaniaă(TAN) [TAN],12,0,2,0,2,0,0,0,0,0,12,0,2,0,2
Thailandă(THA),15,7,6,11,24,3,0,0,0,0,18,7,6,11,24
Togoă(TOG),9,0,0,1,1,1,0,0,0,0,10,0,0,1,1
Tongaă(TGA),8,0,1,0,1,1,0,0,0,0,9,0,1,0,1
Trinidad and Tobagoă(TRI) [TRI],16,2,5,11,18,3,0,0,0,0,19,2,5,11,18
Tunisiaă(TUN),13,3,3,4,10,0,0,0,0,0,13,3,3,4,10
Turkeyă(TUR),21,39,25,24,88,16,0,0,0,0,37,39,25,24,88
Ugandaă(UGA),14,2,3,2,7,0,0,0,0,0,14,2,3,2,7
Ukraineă(UKR),5,33,27,55,115,6,2,1,4,7,11,35,28,59,122
United Arab Emiratesă(UAE),8,1,0,0,1,0,0,0,0,0,8,1,0,0,1
United Statesă(USA) [P] [Q] [R] [Z],26,976,757,666,2399,22,96,102,84,282,48,1072,859,750,2681
Uruguayă(URU),20,2,2,6,10,1,0,0,0,0,21,2,2,6,10
Uzbekistană(UZB),5,5,5,10,20,6,1,0,0,1,11,6,5,10,21
Venezuelaă(VEN),17,2,2,8,12,4,0,0,0,0,21,2,2,8,12
Vietnamă(VIE),14,0,2,0,2,0,0,0,0,0,14,0,2,0,2
Virgin Islandsă(ISV),11,0,1,0,1,7,0,0,0,0,18,0,1,0,1
Yugoslaviaă(YUG) [YUG],16,26,29,28,83,14,0,3,1,4,30,26,32,29,87

```
Independent Olympic Participantsă(IOP) [IOP],1,0,1,2,3,0,0,0,0,0,1,0,1,2,3
Zambiaă(ZAM) [ZAM],12,0,1,1,2,0,0,0,0,0,12,0,1,1,2
Zimbabweă(ZIM) [ZIM],12,3,4,1,8,1,0,0,0,0,13,3,4,1,8
Mixed teamă(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
```

- We can read data into a DataFrame by calling the **read_csv** function of the module. When we look at the DataFrame we see that the first cell has an **NaN** in it since it's an empty value, and the rows have been automatically indexed for us.
- **pandas.read_csv** *(filepath_or_buffer: Union[str, pathlib.Path, IO[~AnyStr]], sep=',', delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, skipfooter=0, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, cache_dates=True, iterator=False, chunksize=None, compression='infer', thousands=None, decimal: str = '.', lineterminator=None, quotechar='"', quoting=0, doublequote=True, escapechar=None, comment=None, encoding=None, dialect=None, error_bad_lines=True, warn_bad_lines=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None)*

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

Out[ ]:

	0	1	2	3	4	5	6	7	8
0	NaN	Summer	01 !	02 !	03 !	Total	Winter	01 !	02 !
1	Afghanistană(AFG)	13	0	0	2	2	0	0	0
2	Algeriaă(ALG)	12	5	2	8	15	3	0	0
3	Argentinaă(ARG)	23	18	24	28	70	18	0	0
4	Armeniaă(ARM)	5	1	2	9	12	6	0	0

	9	10	11	12	13	14	15
0	03 !	Total	Games	01 !	02 !	03 !	Combined total
1	0	0	13	0	0	2	2
2	0	0	15	5	2	8	15
3	0	0	41	18	24	28	70
4	0	0	11	1	2	9	12

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

Out[ ]:

	Summer	01 !	02 !	03 !	Total	Winter	01 !.1	\
Afghanistană(AFG)	13	0	0	2	2	0	0	
Algeriaă(ALG)	12	5	2	8	15	3	0	
Argentinaă(ARG)	23	18	24	28	70	18	0	
Armeniaă(ARM)	5	1	2	9	12	6	0	
Australasiaă(ANZ) [ANZ]	2	3	4	5	12	0	0	

```
                           02 !.1  03 !.1  Total.1   Games  01 !.2  02 !.2  \
Afghanistană(AFG)               0       0        0      13       0       0
Algeriaă(ALG)                   0       0        0      15       5       2
Argentinaă(ARG)                 0       0        0      41      18      24
Armeniaă(ARM)                   0       0        0      11       1       2
Australasiaă(ANZ) [ANZ]         0       0        0       2       3       4

                           03 !.2  Combined total
Afghanistană(AFG)               2               2
Algeriaă(ALG)                   8              15
Argentinaă(ARG)                28              70
Armeniaă(ARM)                   9              12
Australasiaă(ANZ) [ANZ]         5              12
```

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

```
Out[ ]: 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')
```

- **DataFrame.rename** *(self, mapper=None, index=None, columns=None, axis=None, copy=True, inplace=False, level=None, errors='ignore')* Alter axes labels.

Function / dict values must be unique (1-to-1). Labels not contained in a dict / Series will be left as-is. Extra labels listed don't throw an error.

`DataFrame.rename` supports two calling conventions:- - (index=index_mapper, columns=columns_mapper, ...)

- (mapper, axis={'index', 'columns'}, ...)

```
In [ ]: for col in df.columns:
            if col[:2]=='01':
                df.rename(columns={col:'Gold' + col[4:]}, inplace=True)     # Dict type values a
            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[ ]:                           # Summer  Gold  Silver  Bronze  Total  # Winter  \
Afghanistană(AFG)                     13     0       0       2      2         0
Algeriaă(ALG)                         12     5       2       8     15         3
Argentinaă(ARG)                       23    18      24      28     70        18
Armeniaă(ARM)                          5     1       2       9     12         6
```

```
Australasiaă(ANZ) [ANZ]         2     3       4       5     12         0

                           Gold.1  Silver.1  Bronze.1  Total.1  # Games  Gold.2  \
Afghanistană(AFG)               0         0         0        0       13       0
Algeriaă(ALG)                   0         0         0        0       15       5
Argentinaă(ARG)                 0         0         0        0       41      18
Armeniaă(ARM)                   0         0         0        0       11       1
Australasiaă(ANZ) [ANZ]         0         0         0        0        2       3

                           Silver.2  Bronze.2  Combined total
Afghanistană(AFG)                 0         2               2
Algeriaă(ALG)                     2         8              15
Argentinaă(ARG)                  24        28              70
Armeniaă(ARM)                     2         9              12
Australasiaă(ANZ) [ANZ]           4         5              12
```

5 Querying a DataFrame

- Boolean Operations can be done in DataFrame which results into True or False value to data.

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

```
Out[ ]: Afghanistană(AFG)                          False
        Algeriaă(ALG)                              True
        Argentinaă(ARG)                            True
        Armeniaă(ARM)                              True
        Australasiaă(ANZ) [ANZ]                    True
        Australiaă(AUS) [AUS] [Z]                  True
        Austriaă(AUT)                              True
        Azerbaijană(AZE)                           True
        Bahamasă(BAH)                              True
        Bahrainăă(BRN)                            False
        Barbadosă(BAR) [BAR]                      False
        Belarusă(BLR)                              True
        Belgiumă(BEL)                              True
        Bermudaă(BER)                             False
        Bohemiaă(BOH) [BOH] [Z]                   False
        Botswanaă(BOT)                            False
        Brazilă(BRA)                               True
        British West Indiesă(BWI) [BWI]           False
        Bulgariaă(BUL) [H]                         True
        Burundiă(BDI)                              True
        Cameroonă(CMR)                             True
        Canadaă(CAN)                               True
        Chileă(CHI) [I]                            True
        Chinaă(CHN) [CHN]                          True
        Colombiaă(COL)                             True
        Costa Ricaă(CRC)                           True
```

```
Ivory Coastă(CIV) [CIV]                          False
Croatiaă(CRO)                                     True
Cubaă(CUB) [Z]                                    True
Cyprusă(CYP)                                     False
                                                   ...
Sri Lankaă(SRI) [SRI]                            False
Sudană(SUD)                                      False
Surinameă(SUR) [E]                                True
Swedenă(SWE) [Z]                                  True
Switzerlandă(SUI)                                 True
Syriaă(SYR)                                       True
Chinese Taipeiă(TPE) [TPE] [TPE2]                 True
Tajikistană(TJK)                                 False
Tanzaniaă(TAN) [TAN]                             False
Thailandă(THA)                                    True
Togoă(TOG)                                       False
Tongaă(TGA)                                      False
Trinidad and Tobagoă(TRI) [TRI]                   True
Tunisiaă(TUN)                                     True
Turkeyă(TUR)                                      True
Ugandaă(UGA)                                      True
Ukraineă(UKR)                                     True
United Arab Emiratesă(UAE)                        True
United Statesă(USA) [P] [Q] [R] [Z]               True
Uruguayă(URU)                                     True
Uzbekistană(UZB)                                  True
Venezuelaă(VEN)                                   True
Vietnamă(VIE)                                    False
Virgin Islandsă(ISV)                             False
Yugoslaviaă(YUG) [YUG]                            True
Independent Olympic Participantsă(IOP) [IOP]     False
Zambiaă(ZAM) [ZAM]                               False
Zimbabweă(ZIM) [ZIM]                              True
Mixed teamă(ZZX) [ZZX]                            True
Totals                                            True
Name: Gold, dtype: bool
```

- **DataFrame.where** *(self, cond, other=nan, inplace=False, axis=None, level=None, errors='raise', try_cast=False)* **cond**: bool Series/DataFrame, array-like, or callable Where cond is True, keep the original value. Where False, replace with corresponding value from other. I

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

```
Out[ ]:                        # Summer  Gold  Silver  Bronze  Total  # Winter  \
        Afghanistană(AFG)           NaN   NaN     NaN     NaN    NaN       NaN
        Algeriaă(ALG)              12.0   5.0     2.0     8.0   15.0       3.0
        Argentinaă(ARG)            23.0  18.0    24.0    28.0   70.0      18.0
```

```
Armeniaă(ARM)                5.0   1.0     2.0     9.0   12.0       6.0
Australasiaă(ANZ) [ANZ]      2.0   3.0     4.0     5.0   12.0       0.0

                          Gold.1  Silver.1  Bronze.1  Total.1  # Games  Gold.2  \
Afghanistană(AFG)            NaN       NaN       NaN      NaN      NaN     NaN
Algeriaă(ALG)                0.0       0.0       0.0      0.0     15.0     5.0
Argentinaă(ARG)              0.0       0.0       0.0      0.0     41.0    18.0
Armeniaă(ARM)                0.0       0.0       0.0      0.0     11.0     1.0
Australasiaă(ANZ) [ANZ]      0.0       0.0       0.0      0.0      2.0     3.0

                          Silver.2  Bronze.2  Combined total
Afghanistană(AFG)              NaN       NaN             NaN
Algeriaă(ALG)                  2.0       8.0            15.0
Argentinaă(ARG)               24.0      28.0            70.0
Armeniaă(ARM)                  2.0       9.0            12.0
Australasiaă(ANZ) [ANZ]        4.0       5.0            12.0
```

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

```
Out[ ]: 100
```

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

```
Out[ ]: 147
```

```
In [ ]: only_gold = only_gold.dropna()            #  Here only values with NaN will drop from da
        only_gold.head()
```

```
Out[ ]:                           # Summer   Gold  Silver  Bronze  Total  # Winter  \
        Algeriaă(ALG)                 12.0    5.0     2.0     8.0   15.0       3.0
        Argentinaă(ARG)               23.0   18.0    24.0    28.0   70.0      18.0
        Armeniaă(ARM)                  5.0    1.0     2.0     9.0   12.0       6.0
        Australasiaă(ANZ) [ANZ]        2.0    3.0     4.0     5.0   12.0       0.0
        Australiaă(AUS) [AUS] [Z]     25.0  139.0   152.0   177.0  468.0      18.0

                                  Gold.1  Silver.1  Bronze.1  Total.1  # Games  \
        Algeriaă(ALG)                0.0       0.0       0.0      0.0     15.0
        Argentinaă(ARG)              0.0       0.0       0.0      0.0     41.0
        Armeniaă(ARM)                0.0       0.0       0.0      0.0     11.0
        Australasiaă(ANZ) [ANZ]      0.0       0.0       0.0      0.0      2.0
        Australiaă(AUS) [AUS] [Z]    5.0       3.0       4.0     12.0     43.0

                                  Gold.2  Silver.2  Bronze.2  Combined total
        Algeriaă(ALG)                5.0       2.0       8.0            15.0
        Argentinaă(ARG)             18.0      24.0      28.0            70.0
        Armeniaă(ARM)                1.0       2.0       9.0            12.0
        Australasiaă(ANZ) [ANZ]      3.0       4.0       5.0            12.0
        Australiaă(AUS) [AUS] [Z]  144.0     155.0     181.0           480.0
```

```
In [ ]: only_gold = df[df['Gold'] > 0]        # Here directly NaN drop and integer value will all
        only_gold.head()
```

Out[ ]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	\
Algeriaă(ALG)	12	5	2	8	15	3	
Argentinaă(ARG)	23	18	24	28	70	18	
Armeniaă(ARM)	5	1	2	9	12	6	
Australasiaă(ANZ) [ANZ]	2	3	4	5	12	0	
Australiaă(AUS) [AUS] [Z]	25	139	152	177	468	18	

	Gold.1	Silver.1	Bronze.1	Total.1	# Games	\
Algeriaă(ALG)	0	0	0	0	15	
Argentinaă(ARG)	0	0	0	0	41	
Armeniaă(ARM)	0	0	0	0	11	
Australasiaă(ANZ) [ANZ]	0	0	0	0	2	
Australiaă(AUS) [AUS] [Z]	5	3	4	12	43	

	Gold.2	Silver.2	Bronze.2	Combined total
Algeriaă(ALG)	5	2	8	15
Argentinaă(ARG)	18	24	28	70
Armeniaă(ARM)	1	2	9	12
Australasiaă(ANZ) [ANZ]	3	4	5	12
Australiaă(AUS) [AUS] [Z]	144	155	181	480

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

Out[ ]: 101

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

Out[ ]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	\
Liechtensteină(LIE)	16	0	0	0	0	18	2	

	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	\
Liechtensteină(LIE)	2	5	9	34	2	2	

	Bronze.2	Combined total
Liechtensteină(LIE)	5	9

6 Indexing Dataframes

```
In [ ]: df.head()                # Return the first n rows of df.head(self,n)
```

Out[ ]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	\
Afghanistană(AFG)	13	0	0	2	2	0	
Algeriaă(ALG)	12	5	2	8	15	3	
Argentinaă(ARG)	23	18	24	28	70	18	
Armeniaă(ARM)	5	1	2	9	12	6	

```
Australasiaă(ANZ) [ANZ]         2     3       4       5     12         0

                           Gold.1  Silver.1  Bronze.1  Total.1  # Games  Gold.2  \
Afghanistană(AFG)               0         0         0        0       13       0
Algeriaă(ALG)                   0         0         0        0       15       5
Argentinaă(ARG)                 0         0         0        0       41      18
Armeniaă(ARM)                   0         0         0        0       11       1
Australasiaă(ANZ) [ANZ]         0         0         0        0        2       3

                           Silver.2  Bronze.2  Combined total
Afghanistană(AFG)                 0         2               2
Algeriaă(ALG)                     2         8              15
Argentinaă(ARG)                  24        28              70
Armeniaă(ARM)                     2         9              12
Australasiaă(ANZ) [ANZ]           4         5              12
```

In [ ]:
```
df['country'] = df.index                    # current index will be alotted to new 'cou
df = df.set_index('Gold')
df.head()
```

Out[ ]:
```
      # Summer  Silver  Bronze  Total  # Winter  Gold.1  Silver.1  Bronze.1  \
Gold
0           13       0       2      2         0       0         0         0
5           12       2       8     15         3       0         0         0
18          23      24      28     70        18       0         0         0
1            5       2       9     12         6       0         0         0
3            2       4       5     12         0       0         0         0

      Total.1  # Games  Gold.2  Silver.2  Bronze.2  Combined total  \
Gold
0           0       13       0         0         2               2
5           0       15       5         2         8              15
18          0       41      18        24        28              70
1           0       11       1         2         9              12
3           0        2       3         4         5              12

                       country
Gold
0           Afghanistană(AFG)
5               Algeriaă(ALG)
18            Argentinaă(ARG)
1               Armeniaă(ARM)
3     Australasiaă(ANZ) [ANZ]
```

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

Out[ ]:
```
   Gold  # Summer  Silver  Bronze  Total  # Winter  Gold.1  Silver.1  \
0     0        13       0       2      2         0       0         0
```

```
1     5        12       2       8     15         3       0         0
2    18        23      24      28     70        18       0         0
3     1         5       2       9     12         6       0         0
4     3         2       4       5     12         0       0         0

   Bronze.1  Total.1  # Games  Gold.2  Silver.2  Bronze.2  Combined total  \
0         0        0       13       0         0         2               2
1         0        0       15       5         2         8              15
2         0        0       41      18        24        28              70
3         0        0       11       1         2         9              12
4         0        0        2       3         4         5              12

                    country
0         Afghanistană(AFG)
1             Algeriaă(ALG)
2           Argentinaă(ARG)
3             Armeniaă(ARM)
4  Australasiaă(ANZ) [ANZ]
```

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

Out[ ]:
```
   SUMLEV  REGION  DIVISION  STATE  COUNTY   STNAME         CTYNAME  \
0      40       3         6      1       0  Alabama         Alabama
1      50       3         6      1       1  Alabama  Autauga County
2      50       3         6      1       3  Alabama  Baldwin County
3      50       3         6      1       5  Alabama  Barbour County
4      50       3         6      1       7  Alabama     Bibb County

   CENSUS2010POP  ESTIMATESBASE2010  POPESTIMATE2010     ...       \
0        4779736            4780127          4785161     ...
1          54571              54571            54660     ...
2         182265             182265           183193     ...
3          27457              27457            27341     ...
4          22915              22919            22861     ...

   RDOMESTICMIG2011  RDOMESTICMIG2012  RDOMESTICMIG2013  RDOMESTICMIG2014  \
0          0.002295         -0.193196          0.381066          0.582002
1          7.242091         -2.915927         -3.012349          2.265971
2         14.832960         17.647293         21.845705         19.243287
3         -4.728132         -2.500690         -7.056824         -3.904217
4         -5.527043         -5.068871         -6.201001         -0.177537

   RDOMESTICMIG2015  RNETMIG2011  RNETMIG2012  RNETMIG2013  RNETMIG2014  \
0         -0.467369     1.030015     0.826644     1.383282     1.724718
1         -2.530799     7.606016    -2.626146    -2.722002     2.592270
2         17.197872    15.844176    18.559627    22.727626    20.317142
3        -10.543299    -4.874741    -2.758113    -7.167664    -3.978583
```

```
4          0.177258    -5.088389    -4.363636    -5.403729     0.754533

   RNETMIG2015
0     0.712594
1    -2.187333
2    18.293499
3   -10.543299
4     1.107861

[5 rows x 100 columns]
```

```
In [ ]: df['SUMLEV'].unique()          # Return unique values
```

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

```
In [ ]: df=df[df['SUMLEV'] == 50]           # boolean values used to query the dataframe
        df.head()
```

```
Out[ ]:    SUMLEV  REGION  DIVISION  STATE  COUNTY   STNAME         CTYNAME  \
        1      50       3         6      1       1  Alabama  Autauga County
        2      50       3         6      1       3  Alabama  Baldwin County
        3      50       3         6      1       5  Alabama  Barbour County
        4      50       3         6      1       7  Alabama     Bibb County
        5      50       3         6      1       9  Alabama   Blount County

           CENSUS2010POP  ESTIMATESBASE2010  POPESTIMATE2010     ...       \
        1          54571              54571            54660     ...
        2         182265             182265           183193     ...
        3          27457              27457            27341     ...
        4          22915              22919            22861     ...
        5          57322              57322            57373     ...

           RDOMESTICMIG2011  RDOMESTICMIG2012  RDOMESTICMIG2013  RDOMESTICMIG2014  \
        1          7.242091         -2.915927         -3.012349          2.265971
        2         14.832960         17.647293         21.845705         19.243287
        3         -4.728132         -2.500690         -7.056824         -3.904217
        4         -5.527043         -5.068871         -6.201001         -0.177537
        5          1.807375         -1.177622         -1.748766         -2.062535

           RDOMESTICMIG2015  RNETMIG2011  RNETMIG2012  RNETMIG2013  RNETMIG2014  \
        1         -2.530799     7.606016    -2.626146    -2.722002     2.592270
        2         17.197872    15.844176    18.559627    22.727626    20.317142
        3        -10.543299    -4.874741    -2.758113    -7.167664    -3.978583
        4          0.177258    -5.088389    -4.363636    -5.403729     0.754533
        5         -1.369970     1.859511    -0.848580    -1.402476    -1.577232

           RNETMIG2015
        1    -2.187333
        2    18.293499
```

```
3   -10.543299
4     1.107861
5    -0.884411

[5 rows x 100 columns]
```

```
In [ ]: 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[ ]:

	STNAME	CTYNAME	BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPESTIMATE2010	POPESTIMATE2011	POPESTIMATE2012	POPESTIMATE2013	POPESTIMATE2014	POPESTIMATE2015
1	Alabama	Autauga County	151	636	615	574	623	600	54660	55253	55175	55038	55290	55347
2	Alabama	Baldwin County	517	2187	2092	2160	2186	2240	183193	186659	190396	195126	199713	203709
3	Alabama	Barbour County	70	335	300	283	260	269	27341	27226	27159	26973	26815	26489
4	Alabama	Bibb County	44	266	245	259	247	253	22861	22733	22642	22512	22549	22583
5	Alabama	Blount County	183	744	710	646	618	603	57373	57711	57776	57734	57658	57673

- **DataFrame.set_index** *(self, keys, drop=True, append=False, inplace=False, verify_integrity=False)*
  Set the DataFrame index using existing columns.

Set the DataFrame index (row labels) using one or more existing columns or arrays (of the correct length). The index can replace the existing index or expand on it.

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

Out[ ]:

STNAME	CTYNAME	BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2015	POPESTIMATE2010	POPESTIMATE2011	POPESTIMATE2012	POPESTIMATE2013	POPESTIMATE2014	POPESTIMATE2015
Alabama	Autauga County	151	636	615	574	623	600	54660	55253	55175	55038	55290	55347
	Baldwin County	517	2187	2092	2160	2186	2240	183193	186659	190396	195126	199713	203709
	Barbour County	70	335	300	283	260	269	27341	27226	27159	26973	26815	26489
	Bibb County	44	266	245	259	247	253	22861	22733	22642	22512	22549	22583
	Blount County	183	744	710	646	618	603	57373	57711	57776	57734	57658	57673

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

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

Out[ ]:

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

		BIRTHS2014	BIRTHS2015	POPESTIMATE2010	\
STNAME	CTYNAME				
Michigan	Washtenaw County	3683	3709	345563	
	Wayne County	23607	23586	1815199	

		POPESTIMATE2011	POPESTIMATE2012	POPESTIMATE2013	\
STNAME	CTYNAME				
Michigan	Washtenaw County	349048	351213	354289	
	Wayne County	1801273	1792514	1775713	

		POPESTIMATE2014	POPESTIMATE2015
STNAME	CTYNAME		
Michigan	Washtenaw County	357029	358880
	Wayne County	1766008	1759335

7 Missing values

```
In [ ]: df = pd.read_csv('log.csv')
        df
```

Out[ ]:

	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
```

- **DataFrame.fillna** *(self, value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)* Union[ForwardRef('DataFrame'), NoneType] Fill NA/NaN values using the specified method. **value** : scalar, dict, Series, or DataFrame Value to use to fill holes (e.g. 0), alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). Values not in the dict/Series/DataFrame will not be filled. This value cannot be a list.

method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None Method to use for filling holes in reindexed Series pad / ffill: propagate last valid observation forward to next valid backfill / bfill: use next valid observation to fill gap.

```
In [ ]: df.fillna?
```

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

```
Out[ ]:              user          video  playback position paused  volume
        time
        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 [ ]:
```
df = df.reset_index()
df = df.set_index(['time', 'user'])
df
```

Out[ ]:

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

Out[ ]:

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