pandas1

September 25, 2025

[276]:
```
import pandas as pd
import numpy as np
```

0.1

0.1.1 Series

[277]:
```
obj = pd.Series([4, 7, -5, 3])
print(obj) #
print(obj.values) #
print(obj.index) #

obj2 = pd.Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])
print(obj2)
print(obj2.index)

print(obj2['a']) #
obj2['d'] = 6 #
print(obj2[['c', 'a', 'd']]) #
```

```
0    4
1    7
2   -5
3    3
dtype: int64
[ 4  7 -5  3]
RangeIndex(start=0, stop=4, step=1)
d    4
b    7
a   -5
c    3
dtype: int64
Index(['d', 'b', 'a', 'c'], dtype='object')
-5
c    3
a   -5
d    6
dtype: int64
```

[278]:
```
print(obj2[obj2 > 0]) #
print(obj2 * 2) #
print(np.exp(obj2)) #numpy

print('b' in obj2) #

sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000} #  Series
obj3 = pd.Series(sdata)
print(obj3)
```

```
d    6
b    7
c    3
dtype: int64
d    12
b    14
a   -10
c     6
dtype: int64
d     403.428793
b    1096.633158
a       0.006738
c      20.085537
dtype: float64
True
Ohio      35000
Texas     71000
Oregon    16000
Utah       5000
dtype: int64
```

[279]:
```
states = ['California', 'Ohio', 'Oregon', 'Texas'] #   Series
obj4 = pd.Series(sdata, index=states)
print(obj4) #  NaN
print(pd.isnull(obj4)) #
print(pd.notnull(obj4)) #
print(obj3 + obj4) #

obj4.name = 'population' # name
obj4.index.name = 'state' # index name
print(obj4)
```

```
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
dtype: float64
California     True
```

```
Ohio          False
Oregon        False
Texas         False
dtype: bool
California    False
Ohio           True
Oregon         True
Texas          True
dtype: bool
California         NaN
Ohio           70000.0
Oregon         32000.0
Texas         142000.0
Utah               NaN
dtype: float64
state
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
Name: population, dtype: float64
```

[280]:
```python
obj.index = ['Bob', 'Steve', 'Jeff', 'Ryan'] #
print(obj)
```

```
Bob      4
Steve    7
Jeff    -5
Ryan     3
dtype: int64
```

0.1.2 DataFrame

[281]:
```python
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
                'year': [2000, 2001, 2002, 2001, 2002, 2003],
                'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
frame = pd.DataFrame(data) #  DataFrame
print(frame)
print(frame.head()) #
print(frame.tail(3)) # 3
```

```
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
5  Nevada  2003  3.2
```

```
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
    state  year  pop
3  Nevada  2001  2.4
4  Nevada  2002  2.9
5  Nevada  2003  3.2
```

[282]:
```
print(pd.DataFrame(data, columns=['year', 'state', 'pop'])) #
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],␣
 ↪index=['one', 'two', 'three', 'four', 'five', 'six']) #
print(frame2)

print(frame.columns) #
```

```
   year   state  pop
0  2000    Ohio  1.5
1  2001    Ohio  1.7
2  2002    Ohio  3.6
3  2001  Nevada  2.4
4  2002  Nevada  2.9
5  2003  Nevada  3.2
       year   state  pop debt
one    2000    Ohio  1.5  NaN
two    2001    Ohio  1.7  NaN
three  2002    Ohio  3.6  NaN
four   2001  Nevada  2.4  NaN
five   2002  Nevada  2.9  NaN
six    2003  Nevada  3.2  NaN
Index(['state', 'year', 'pop'], dtype='object')
```

[283]:
```
print(frame2['state']) #
print(frame2.year) #

print(frame2.loc['three']) #
```

```
one        Ohio
two        Ohio
three      Ohio
four     Nevada
five     Nevada
six      Nevada
Name: state, dtype: object
one      2000
two      2001
```

```
three    2002
four     2001
five     2002
six      2003
Name: year, dtype: int64
year     2002
state    Ohio
pop       3.6
debt      NaN
Name: three, dtype: object
```

[284]:
```
frame2['debt'] = 16.5 #
print(frame2)
frame2['debt'] = np.arange(6.)
print(frame2)
```

```
       year   state  pop  debt
one    2000    Ohio  1.5  16.5
two    2001    Ohio  1.7  16.5
three  2002    Ohio  3.6  16.5
four   2001  Nevada  2.4  16.5
five   2002  Nevada  2.9  16.5
six    2003  Nevada  3.2  16.5
       year   state  pop  debt
one    2000    Ohio  1.5   0.0
two    2001    Ohio  1.7   1.0
three  2002    Ohio  3.6   2.0
four   2001  Nevada  2.4   3.0
five   2002  Nevada  2.9   4.0
six    2003  Nevada  3.2   5.0
```

[285]:
```
val = pd.Series([-1.2, -1.5, -1.7], index=['two', 'four', 'five'])
frame2['debt'] = val #
print(frame2)

frame2['eastern'] = frame2.state == 'Ohio' #
print(frame2)
del frame2['eastern'] #
print(frame2.columns) #
```

```
       year   state  pop  debt
one    2000    Ohio  1.5   NaN
two    2001    Ohio  1.7  -1.2
three  2002    Ohio  3.6   NaN
four   2001  Nevada  2.4  -1.5
five   2002  Nevada  2.9  -1.7
six    2003  Nevada  3.2   NaN
       year   state  pop  debt  eastern
```

```
one     2000    Ohio  1.5   NaN     True
two     2001    Ohio  1.7  -1.2     True
three   2002    Ohio  3.6   NaN     True
four    2001  Nevada  2.4  -1.5    False
five    2002  Nevada  2.9  -1.7    False
six     2003  Nevada  3.2   NaN    False
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

[286]:
```
pop = {'Nevada': {2001: 2.4, 2002: 2.9},
           'Ohio': {2000: 1.5, 2001: 1.7, 2002: 3.6}}
frame3 = pd.DataFrame(pop) #   DataFrame
print(frame3)
print(frame3.T) #

pdata = {'Ohio': frame3['Ohio'][:-1], 'Nevada': frame3['Nevada'][:2]}␣
  ↪#   DataFrame
print(pd.DataFrame(pdata))

frame3.index.name = 'year' # index name
frame3.columns.name = 'state' # columns name
print(frame3)

print(frame3.values) # ndarray
print(frame3.values.dtype) #
print(frame2.values) # ndarray     object
print(frame2.values.dtype) #
```

```
      Nevada  Ohio
2001     2.4   1.7
2002     2.9   3.6
2000     NaN   1.5
        2001  2002  2000
Nevada   2.4   2.9   NaN
Ohio     1.7   3.6   1.5
      Ohio  Nevada
2001   1.7     2.4
2002   3.6     2.9
state  Nevada  Ohio
year
2001      2.4   1.7
2002      2.9   3.6
2000      NaN   1.5
[[2.4 1.7]
 [2.9 3.6]
 [nan 1.5]]
float64
[[2000 'Ohio' 1.5 nan]
 [2001 'Ohio' 1.7 -1.2]
 [2002 'Ohio' 3.6 nan]
 [2001 'Nevada' 2.4 -1.5]
 [2002 'Nevada' 2.9 -1.7]
 [2003 'Nevada' 3.2 nan]]
object
```

DataFrame

- 2D ndarray
- DataFrame
- NumPy /
- Series Series
-
- Series DataFrame Series DataFrame
-
- DataFrame DatraFrame
- NumPy MaskedArray

0.1.3

[287]:
```python
obj = pd.Series(range(3), index=['a', 'b', 'c'])
index = obj.index
print(obj)
print(index)
print(index[1:]) #
print(index[1]) #
# index[1] = 'd' #
lables = pd.Index(np.arange(3)) # Index
print(lables)
obj2 = pd.Series([1.5, -2.5, 0], index=lables)
print(obj2)
print(obj2.index is lables) #obj2   lables
```

```
a    0
b    1
c    2
dtype: int64
Index(['a', 'b', 'c'], dtype='object')
Index(['b', 'c'], dtype='object')
b
Index([0, 1, 2], dtype='int64')
0    1.5
1   -2.5
2    0.0
dtype: float64
True
```

```
[288]: #
print(frame3)
print('Ohio' in frame3.columns) #
print(2003 in frame3.index) #
dup_lables = pd.Index(['foo', 'foo', 'bar', 'bar']) #
print(dup_lables)
```

```
state  Nevada  Ohio
year
2001      2.4   1.7
2002      2.9   3.6
2000      NaN   1.5
True
False
Index(['foo', 'foo', 'bar', 'bar'], dtype='object')
```

```
[289]: del obj, obj2, obj3, obj4, frame, frame2, frame3, data, sdata, states, pop,␣
        ↪pdata, val, index, lables, dup_lables
```

0.2

0.2.1

Series reindex()

```
[290]: obj = pd.Series([4.5, 7.2, -5.3, 3.6], index=['d', 'b', 'a', 'c'])
print(obj)
obj2 = obj.reindex(['a', 'b', 'c', 'd', 'e']) #     NaN
print(obj2)
obj3 = obj.reindex(['a', 'b', 'c', 'd', 'e'], fill_value=0) #      0
print(obj3)
obj4 = pd.Series(['blue', 'purple', 'yellow'], index=[0, 2, 4])
print(obj4)
obj4 = obj4.reindex(range(6), method='ffill') #
print(obj4)
```

```
d    4.5
b    7.2
a   -5.3
c    3.6
dtype: float64
a   -5.3
b    7.2
c    3.6
d    4.5
e    NaN
dtype: float64
a   -5.3
```

```
b    7.2
c    3.6
d    4.5
e    0.0
dtype: float64
0      blue
2    purple
4    yellow
dtype: object
0      blue
1      blue
2    purple
3    purple
4    yellow
5    yellow
dtype: object
```

DataFrame

```
[291]: frame = pd.DataFrame(np.arange(9).reshape((3, 3)), index=['a', 'c', 'd'],␣
        ↪columns=['Ohio', 'Texas', 'California'])
       print(frame)
       frame2 = frame.reindex(['a', 'b', 'c', 'd']) #      NaN
       print(frame2)
       states = ['Texas', 'Utah', 'California']
       frame3 = frame.reindex(columns=states) #      NaN
       print(frame3)
```

```
   Ohio  Texas  California
a     0      1           2
c     3      4           5
d     6      7           8
   Ohio  Texas  California
a   0.0    1.0         2.0
b   NaN    NaN         NaN
c   3.0    4.0         5.0
d   6.0    7.0         8.0
   Texas  Utah  California
a      1   NaN           2
c      4   NaN           5
d      7   NaN           8
```

reindex - index python - method 'ffill' 'bfill' -
fill_value - limit - tolerance -
level MultiIndex - copy True False

0.2.2

[292]:
```
obj = pd.Series(np.arange(5.), index=['a', 'b', 'c', 'd', 'e'])
print(obj)
new_obj = obj.drop('c') #
print(new_obj)
obj.drop(['d', 'c']) #

data = pd.DataFrame(np.arange(16).reshape((4, 4)),
                                        index=['Ohio', 'Colorado', 'Utah', 'New␣
 ↪York'],
                                        columns=['one', 'two', 'three', 'four'])
print(data)
data2 = data.drop(['Colorado', 'Ohio']) #
print(data2)
print(data.drop('two', axis=1)) #
print(data.drop(['two', 'four'], axis='columns')) #
```

```
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
          one  two  three  four
Ohio        0    1      2     3
Colorado    4    5      6     7
Utah        8    9     10    11
New York   12   13     14    15
          one  two  three  four
Utah        8    9     10    11
New York   12   13     14    15
          one  three  four
Ohio        0      2     3
Colorado    4      6     7
Utah        8     10    11
New York   12     14    15
          one  three
Ohio        0      2
Colorado    4      6
Utah        8     10
New York   12     14
```

[293]:
```
obj.drop('c', inplace=True) # obj
print(obj)
```

```
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

0.2.3

[294]:
```
obj = pd.Series(np.arange(4.), index=['a', 'b', 'c', 'd'])
print(obj)
print(obj['b']) #
print(obj[1]) #
print(obj[2:4]) #
print(obj[['b', 'a', 'd']]) #
print(obj[[1, 3]]) #
print(obj[obj < 2]) #
obj[obj < 2] = 0 #
print(obj)
```

```
a    0.0
b    1.0
c    2.0
d    3.0
dtype: float64
1.0
1.0
c    2.0
d    3.0
dtype: float64
b    1.0
a    0.0
d    3.0
dtype: float64
b    1.0
d    3.0
dtype: float64
a    0.0
b    1.0
dtype: float64
a    0.0
b    0.0
c    2.0
d    3.0
dtype: float64
```

```
/tmp/ipykernel_6887/2710548481.py:4: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  print(obj[1]) #
/tmp/ipykernel_6887/2710548481.py:7: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  print(obj[[1, 3]]) #
```

[295]:

```
data = pd.DataFrame(np.arange(16).reshape((4, 4)),
                                                index=['Ohio', 'Colorado', 'Utah', 'New␣
 ↪York'],
                                                columns=['one', 'two', 'three', 'four'])
print(data)
print(data['two']) #
print(data[['three', 'one']]) #
print(data[:2]) #
```

```
          one  two  three  four
Ohio        0    1      2     3
Colorado    4    5      6     7
Utah        8    9     10    11
New York   12   13     14    15
Ohio         1
Colorado     5
Utah         9
New York    13
Name: two, dtype: int64
          three  one
Ohio          2    0
Colorado      6    4
Utah         10    8
New York     14   12
          one  two  three  four
Ohio        0    1      2     3
Colorado    4    5      6     7
```

[296]:

```
print(data[data['three'] > 5]) #
print(data < 5) #   DataFrame
data[data < 5] = 0 #   DataFrame
print(data)
```

```
          one  two  three  four
Colorado    4    5      6     7
Utah        8    9     10    11
New York   12   13     14    15
```

```
            one    two  three   four
Ohio       True   True   True   True
Colorado   True  False  False  False
Utah      False  False  False  False
New York  False  False  False  False
          one  two  three  four
Ohio        0    0      0     0
Colorado    0    5      6     7
Utah        8    9     10    11
New York   12   13     14    15
```

loc iloc

```
[ ]: print(data.loc['Colorado', ['two', 'three']]) #
     print(data.iloc[2, [3, 0, 1]]) #
     print(data.iloc[2]) #
```

```
two      5
three    6
Name: Colorado, dtype: int64
four    11
one      8
two      9
Name: Utah, dtype: int64
one       8
two       9
three    10
four     11
Name: Utah, dtype: int64
```

0.2.4

```
[298]: df1 = pd.DataFrame(np.arange(12.).reshape((3, 4)), columns=list('abcd'))
       df2 = pd.DataFrame(np.arange(20.).reshape((4, 5)), columns=list('abcde'))
       df2.loc[1, 'b'] = np.nan
       print(df1)
       print(df2)
```

```
     a    b     c     d
0  0.0  1.0   2.0   3.0
1  4.0  5.0   6.0   7.0
2  8.0  9.0  10.0  11.0
      a     b     c     d     e
0   0.0   1.0   2.0   3.0   4.0
1   5.0   NaN   7.0   8.0   9.0
```

```
2  10.0  11.0  12.0  13.0  14.0
3  15.0  16.0  17.0  18.0  19.0
```

[299]:
```
print(df1 + df2)
```

```
      a     b     c     d   e
0   0.0   2.0   4.0   6.0 NaN
1   9.0   NaN  13.0  15.0 NaN
2  18.0  20.0  22.0  24.0 NaN
3   NaN   NaN   NaN   NaN NaN
```

add

[300]:
```
print(df1.add(df2, fill_value=0))
```

```
      a     b     c     d     e
0   0.0   2.0   4.0   6.0   4.0
1   9.0   5.0  13.0  15.0   9.0
2  18.0  20.0  22.0  24.0  14.0
3  15.0  16.0  17.0  18.0  19.0
```

[301]:
```
print(1/df1)
print(df1.rdiv(1))
```

```
       a         b         c         d
0    inf  1.000000  0.500000  0.333333
1  0.250  0.200000  0.166667  0.142857
2  0.125  0.111111  0.100000  0.090909
       a         b         c         d
0    inf  1.000000  0.500000  0.333333
1  0.250  0.200000  0.166667  0.142857
2  0.125  0.111111  0.100000  0.090909
```

reindex

[302]:
```
print(df1.reindex(columns=df2.columns, fill_value=0))
```

```
     a    b     c     d  e
0  0.0  1.0   2.0   3.0  0
1  4.0  5.0   6.0   7.0  0
2  8.0  9.0  10.0  11.0  0
```

- add, radd - sub, rsub - div, rdiv - floordiv, rfloordiv - mul, rmul - pow, rpow

[303]:
```
arr = np.arange(12.).reshape((3, 4))
print(arr)
print(arr[0])
print(arr - arr[0]) #
```

```
[[ 0.  1.  2.  3.]
 [ 4.  5.  6.  7.]
 [ 8.  9. 10. 11.]]
[0. 1. 2. 3.]
[[0. 0. 0. 0.]
 [4. 4. 4. 4.]
 [8. 8. 8. 8.]]
```

DataFrame Series

[304]:
```
frame = pd.DataFrame(np.arange(12.).reshape((4, 3)), columns=list('bde'),␣
  ↪index=['Utah', 'Ohio', 'Texa', 'Oregon'])
series = frame.iloc[0]
print(frame)
print(series)
print(frame-series) # Series  DataFrame
```

```
          b     d     e
Utah    0.0   1.0   2.0
Ohio    3.0   4.0   5.0
Texa    6.0   7.0   8.0
Oregon  9.0  10.0  11.0
b    0.0
d    1.0
e    2.0
Name: Utah, dtype: float64
          b    d    e
Utah    0.0  0.0  0.0
Ohio    3.0  3.0  3.0
Texa    6.0  6.0  6.0
Oregon  9.0  9.0  9.0
```

[305]:
```
series2 = frame['d']
print(frame)
print(series2)
print(frame.sub(series2, axis='index'))
```

```
          b     d     e
Utah    0.0   1.0   2.0
Ohio    3.0   4.0   5.0
Texa    6.0   7.0   8.0
Oregon  9.0  10.0  11.0
Utah       1.0
Ohio       4.0
Texa       7.0
Oregon    10.0
Name: d, dtype: float64
```

```
          b    d    e
Utah   -1.0  0.0  1.0
Ohio   -1.0  0.0  1.0
Texa   -1.0  0.0  1.0
Oregon -1.0  0.0  1.0
```

0.2.5

[306]:
```
frame = pd.DataFrame(np.random.randn(4, 3), columns=list('bde'), index=['Utah',␣
 ↪'Ohio', 'Texa', 'Oregon'])
print(frame)
print(np.abs(frame)) #numpy
f = lambda x: x.max() - x.min() #
print(frame.apply(f)) #
print(frame.apply(f, axis='columns')) #
```

```
               b         d         e
Utah   -0.171523 -0.750481  1.880434
Ohio   -1.053568 -0.826458 -1.497577
Texa    0.988448  0.703451  0.974423
Oregon -0.242716 -0.754870  1.500686
               b         d         e
Utah    0.171523  0.750481  1.880434
Ohio    1.053568  0.826458  1.497577
Texa    0.988448  0.703451  0.974423
Oregon  0.242716  0.754870  1.500686
b    2.042016
d    1.529909
e    3.378011
dtype: float64
Utah      2.630915
Ohio      0.671119
Texa      0.284998
Oregon    2.255555
dtype: float64
```

0.2.6

[307]:
```
object = pd.Series(range(4), index=['d', 'a', 'b', 'c'])
print(object)
print(object.sort_index()) #

frame = pd.DataFrame(np.arange(8).reshape((2, 4)), index=['three', 'one'],␣
 ↪columns=['d', 'a', 'b', 'c'])
print(frame)
print(frame.sort_index()) #
print(frame.sort_index(axis=1)) #
print(frame.sort_index(axis=1, ascending=False)) #
```

```
d    0
a    1
b    2
c    3
dtype: int64
a    1
b    2
c    3
d    0
dtype: int64
       d  a  b  c
three  0  1  2  3
one    4  5  6  7
       d  a  b  c
one    4  5  6  7
three  0  1  2  3
       a  b  c  d
three  1  2  3  0
one    5  6  7  4
       d  c  b  a
three  0  3  2  1
one    4  7  6  5
```

[308]:
```
obj = pd.Series([4, 7, -3, 2])
print(obj)
print(obj.sort_values()) #

obj = pd.Series([4, np.nan, 7, np.nan, -3, 2])
print(obj)
print(obj.sort_values()) #   NaN
```

```
0    4
1    7
2   -3
3    2
dtype: int64
2   -3
3    2
0    4
1    7
dtype: int64
0    4.0
1    NaN
2    7.0
3    NaN
4   -3.0
5    2.0
dtype: float64
```

```
4   -3.0
5    2.0
0    4.0
2    7.0
1    NaN
3    NaN
dtype: float64
```

[309]:
```python
frame = pd.DataFrame({'b': [4, 7, -3, 2], 'a': [0, 1, 0, 1]})
print(frame)
print(frame.sort_values(by='b')) #
print(frame.sort_values(by=['a', 'b'])) #     a  b

obj = pd.Series([7, -5, 7, 4, 2, 0, 4])
print(obj)
print(obj.rank()) #
print(obj.rank(method='first')) #
print(obj.rank(ascending=False, method='max')) #
```

```
   b  a
0  4  0
1  7  1
2 -3  0
3  2  1
   b  a
2 -3  0
3  2  1
0  4  0
1  7  1
   b  a
2 -3  0
0  4  0
3  2  1
1  7  1
0    7
1   -5
2    7
3    4
4    2
5    0
6    4
dtype: int64
0    6.5
1    1.0
2    6.5
3    4.5
4    3.0
5    2.0
```

```
6    4.5
dtype: float64
0    6.0
1    1.0
2    7.0
3    4.0
4    3.0
5    2.0
6    5.0
dtype: float64
0    2.0
1    7.0
2    2.0
3    4.0
4    5.0
5    6.0
6    4.0
dtype: float64
```

[310]:
```
frame = pd.DataFrame({'b': [4.3, 7, -3, 2], 'a': [0, 1, 0, 1], 'c': [-2, 5, 8,␣
  ↪-2.5]})
print(frame)
print(frame.rank(axis='columns')) #
print(frame.rank(axis='index', method='max')) #
```

```
     b  a    c
0  4.3  0 -2.0
1  7.0  1  5.0
2 -3.0  0  8.0
3  2.0  1 -2.5
     b    a    c
0  3.0  2.0  1.0
1  3.0  1.0  2.0
2  1.0  2.0  3.0
3  3.0  2.0  1.0
     b    a    c
0  3.0  2.0  2.0
1  4.0  4.0  3.0
2  1.0  2.0  4.0
3  2.0  4.0  1.0
```

0.2.7

[311]:
```
obj = pd.Series(range(5), index=['a', 'a', 'b', 'b', 'c'])
print(obj)
print(obj.index.is_unique) #
print(obj['a']) #    Series
print(obj['c']) #
```

```
a    0
a    1
b    2
b    3
c    4
dtype: int64
False
a    0
a    1
dtype: int64
4
```

[312]:
```
df = pd.DataFrame(np.random.randn(4, 3), index=['a', 'a', 'b', 'b'])
print(df)
print(df.loc['b']) #    DataFrame
print(df.loc['a']) #    DataFrame
```

```
          0         1         2
a  1.574966 -0.564201 -0.731516
a  0.622392 -0.439082 -0.118496
b -0.152188  0.583649 -1.964059
b  0.277674  0.487701 -1.008114
          0         1         2
b -0.152188  0.583649 -1.964059
b  0.277674  0.487701 -1.008114
          0         1         2
a  1.574966 -0.564201 -0.731516
a  0.622392 -0.439082 -0.118496
```

0.3

[313]:
```
df = pd.DataFrame([[1.4, np.nan], [7.1, -4.5], [np.nan, np.nan], [0.75, -1.3]],
                                  index=['a', 'b', 'c', 'd'],
                                  columns=['one', 'two'])
print(df)
print(df.sum()) #     axis=0
print(df.sum(axis=1)) #
print(df.mean(axis=1, skipna=False)) #    skipna=False NaN   NaN
```

```
    one  two
a  1.40  NaN
b  7.10 -4.5
```

```
c   NaN  NaN
d  0.75 -1.3
one    9.25
two   -5.80
dtype: float64
a    1.40
b    2.60
c    0.00
d   -0.55
dtype: float64
a      NaN
b    1.300
c      NaN
d   -0.275
dtype: float64
```

[314]:
```
print(df.idxmax()) #
print(df.cumsum()) #
```

```
one    b
two    d
dtype: object
    one  two
a  1.40  NaN
b  8.50 -4.5
c   NaN  NaN
d  9.25 -5.8
```

[315]:
```
print(df.describe()) #
print(df.T.describe()) #
```

```
            one       two
count  3.000000  2.000000
mean   3.083333 -2.900000
std    3.493685  2.262742
min    0.750000 -4.500000
25%    1.075000 -3.700000
50%    1.400000 -2.900000
75%    4.250000 -2.100000
max    7.100000 -1.300000
         a         b    c         d
count  1.0  2.000000  0.0  2.000000
mean   1.4  1.300000  NaN -0.275000
std    NaN  8.202439  NaN  1.449569
min    1.4 -4.500000  NaN -1.300000
25%    1.4 -1.600000  NaN -0.787500
50%    1.4  1.300000  NaN -0.275000
75%    1.4  4.200000  NaN  0.237500
```

```
max     1.4  7.100000  NaN  0.750000
```

[316]:
```
obj = pd.Series(['a', 'a', 'b', 'c'] * 4)
print(obj)
print(obj.describe()) #
```

```
0     a
1     a
2     b
3     c
4     a
5     a
6     b
7     c
8     a
9     a
10    b
11    c
12    a
13    a
14    b
15    c
dtype: object
count     16
unique     3
top        a
freq       8
dtype: object
```

0.3.1

[317]:
```
frame = pd.DataFrame(np.random.randn(4, 3), columns=list('bde'), index=['Utah',␣
 ↪'Ohio', 'Texa', 'Oregon'])
print(frame)
print(frame.corr()) #
print(frame.cov()) #
```

```
               b         d         e
Utah    0.569030  1.224514  1.992728
Ohio   -0.865025 -0.103354  1.195028
Texa   -0.058713 -0.786837  0.709671
Oregon -1.099544 -1.762042  0.203053
          b         d         e
b  1.000000  0.781481  0.758563
d  0.781481  1.000000  0.998783
e  0.758563  0.998783  1.000000
          b         d         e
b  0.585265  0.750254  0.442014
```

```
d  0.750254  1.574809  0.954671
e  0.442014  0.954671  0.580145
```

0.3.2

[318]:
```python
obj = pd.Series(['c', 'a', 'd', 'a', 'a', 'b', 'b', 'c', 'c', 'c'])
uniques = obj.unique() #
print(uniques)
print(obj.value_counts()) #
print(pd.value_counts(obj.values, sort=False)) #  sort=False
```

```
['c' 'a' 'd' 'b']
c    4
a    3
b    2
d    1
Name: count, dtype: int64
c    4
a    3
d    1
b    2
Name: count, dtype: int64
```

```
/tmp/ipykernel_6887/2678439928.py:5: FutureWarning: pandas.value_counts is
deprecated and will be removed in a future version. Use
pd.Series(obj).value_counts() instead.
  print(pd.value_counts(obj.values, sort=False)) #  sort=False
```

[319]:
```python
mask = obj.isin(['b', 'c']) #
print(mask)
print(obj[mask]) #
```

```
0     True
1    False
2    False
3    False
4    False
5     True
6     True
7     True
8     True
9     True
dtype: bool
0    c
5    b
6    b
7    c
8    c
```

```
9    c
dtype: object
```

```
[320]: to_match = pd.Series(['c', 'a', 'b', 'b', 'c', 'a'])
unique_vals = pd.Series(['c', 'b', 'a'])
print(pd.Index(unique_vals).get_indexer(to_match))␣
 ↪# to_match   unique_vals     -1

data = pd.DataFrame({'Qu1': [1, 3, 4, 3, 4],
                                         'Qu2': [2, 3, 1, 2, 3],
                                         'Qu3': [1, 5, 2, 4, 4]})
print(data)
result = data.apply(pd.value_counts).fillna(0) #         0
print(result)
```

```
[0 2 1 1 0 2]
   Qu1  Qu2  Qu3
0    1    2    1
1    3    3    5
2    4    1    2
3    3    2    4
4    4    3    4
   Qu1  Qu2  Qu3
1  1.0  1.0  1.0
2  0.0  2.0  1.0
3  2.0  2.0  0.0
4  2.0  0.0  2.0
5  0.0  0.0  1.0
```

```
/tmp/ipykernel_6887/3531663070.py:9: FutureWarning: pandas.value_counts is
deprecated and will be removed in a future version. Use
pd.Series(obj).value_counts() instead.
  result = data.apply(pd.value_counts).fillna(0) #         0
```