```
import numpy as np
import pandas as pd
```

```
from pandas import Series, DataFrame
```

```
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc("figure", figsize=(10, 6))
PREVIOUS_MAX_ROWS = pd.options.display.max_rows
pd.options.display.max_rows = 20
pd.options.display.max_columns = 20
pd.options.display.max_colwidth = 80
np.set_printoptions(precision=4, suppress=True)
```

```
obj = pd.Series([4, 7, -5, 3])
obj
```

	0
0	4
1	7
2	-5
3	3

dtype: int64

```
obj.array
obj.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

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

```
Index(['d', 'b', 'a', 'c'], dtype='object')
```

```
obj2["a"]
obj2["d"] = 6
obj2[["c", "a", "d"]]
```

	0
c	3
a	-5
d	6

dtype: int64

```
obj2[obj2 > 0]
obj2 * 2
import numpy as np
np.exp(obj2)
```

	0
d	403.428793
b	1096.633158
a	0.006738
c	20.085537

dtype: float64

```
"b" in obj2
"e" in obj2
```

```
False
```

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

	0
Ohio	35000
Texas	71000
Oregon	16000
Utah	5000

dtype: int64

```
obj3.to_dict()
```

```
{'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}
```

```
states = ["California", "Ohio", "Oregon", "Texas"]
obj4 = pd.Series(sdata, index=states)
obj4
```

	0
California	NaN
Ohio	35000.0
Oregon	16000.0
Texas	71000.0

dtype: float64

```
pd.isna(obj4)
pd.notna(obj4)
```

	0
California	False
Ohio	True
Oregon	True
Texas	True

dtype: bool

```
obj4.isna()
```

	0
California	True
Ohio	False
Oregon	False
Texas	False

dtype: bool

```
obj3
obj4
obj3 + obj4
```

```
                  0
California      NaN
      Ohio  70000.0
    Oregon  32000.0
     Texas 142000.0
      Utah      NaN

dtype: float64
```

```
obj4.name = "population"
obj4.index.name = "state"
obj4
```

```
             population
      state
California          NaN
      Ohio      35000.0
    Oregon      16000.0
     Texas      71000.0

dtype: float64
```

```
obj
obj.index = ["Bob", "Steve", "Jeff", "Ryan"]
obj
```

```
         0
  Bob    4
Steve    7
 Jeff   -5
 Ryan    3

dtype: int64
```

```
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)
```

```
frame
```

```
     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
```

Next steps: Generate code with `frame` | View recommended plots | New interactive sheet

```
frame.head()
```

	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

Next steps: Generate code with `frame` View recommended plots New interactive sheet

```
frame.tail()
```

	state	year	pop
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

```
pd.DataFrame(data, columns=["year", "state", "pop"])
```

	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

```
frame2 = pd.DataFrame(data, columns=["year", "state", "pop", "debt"])
frame2
frame2.columns
```

```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

```
frame2["state"]
frame2.year
```

	year
0	2000
1	2001
2	2002
3	2001
4	2002
5	2003

dtype: int64

```
frame2.loc[1]
frame2.iloc[2]
```

	2
year	2002
state	Ohio
pop	3.6
debt	NaN

dtype: object

```
frame2["debt"] = 16.5
frame2
frame2["debt"] = np.arange(6.)
frame2
```

	year	state	pop	debt
0	2000	Ohio	1.5	0.0
1	2001	Ohio	1.7	1.0
2	2002	Ohio	3.6	2.0
3	2001	Nevada	2.4	3.0
4	2002	Nevada	2.9	4.0
5	2003	Nevada	3.2	5.0

Next steps: Generate code with frame2 View recommended plots New interactive sheet

```
val = pd.Series([-1.2, -1.5, -1.7], index=["two", "four", "five"])
frame2["debt"] = val
frame2
```

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

Next steps: Generate code with frame2 View recommended plots New interactive sheet

```
frame2["eastern"] = frame2["state"] == "Ohio"
frame2
```

	year	state	pop	debt	eastern
0	2000	Ohio	1.5	NaN	True
1	2001	Ohio	1.7	NaN	True
2	2002	Ohio	3.6	NaN	True
3	2001	Nevada	2.4	NaN	False
4	2002	Nevada	2.9	NaN	False
5	2003	Nevada	3.2	NaN	False

Next steps: Generate code with frame2 View recommended plots New interactive sheet

```
del frame2["eastern"]
frame2.columns
```

```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

```
populations = {"Ohio": {2000: 1.5, 2001: 1.7, 2002: 3.6},
               "Nevada": {2001: 2.4, 2002: 2.9}}
```

```
frame3 = pd.DataFrame(populations)
frame3
```

	Ohio	Nevada
2000	1.5	NaN
2001	1.7	2.4
2002	3.6	2.9

Next steps: Generate code with frame3 | View recommended plots | New interactive sheet

```
frame3.T
```

	2000	2001	2002
Ohio	1.5	1.7	3.6
Nevada	NaN	2.4	2.9

```
pd.DataFrame(populations, index=[2001, 2002, 2003])
```

	Ohio	Nevada
2001	1.7	2.4
2002	3.6	2.9
2003	NaN	NaN

```
pdata = {"Ohio": frame3["Ohio"][:-1],
         "Nevada": frame3["Nevada"][:2]}
pd.DataFrame(pdata)
```

	Ohio	Nevada
2000	1.5	NaN
2001	1.7	2.4

```
frame3.index.name = "year"
frame3.columns.name = "state"
frame3
```

state	Ohio	Nevada
year		
2000	1.5	NaN
2001	1.7	2.4
2002	3.6	2.9

Next steps: Generate code with frame3 | View recommended plots | New interactive sheet

```
frame3.to_numpy()
```

```
array([[1.5, nan],
       [1.7, 2.4],
       [3.6, 2.9]])
```

```
frame2.to_numpy()
```

```
array([[2000, 'Ohio', 1.5, nan],
       [2001, 'Ohio', 1.7, nan],
       [2002, 'Ohio', 3.6, nan],
       [2001, 'Nevada', 2.4, nan],
       [2002, 'Nevada', 2.9, nan],
       [2003, 'Nevada', 3.2, nan]], dtype=object)
```

```
obj = pd.Series(np.arange(3), index=["a", "b", "c"])
index = obj.index
index
index[1:]
```

```
Index(['b', 'c'], dtype='object')
```

```
labels = pd.Index(np.arange(3))
labels
obj2 = pd.Series([1.5, -2.5, 0], index=labels)
obj2
obj2.index is labels
```

```
True
```

```
frame3
frame3.columns
"Ohio" in frame3.columns
2003 in frame3.index
```

```
False
```

```
pd.Index(["foo", "foo", "bar", "bar"])
```

```
Index(['foo', 'foo', 'bar', 'bar'], dtype='object')
```

```
obj = pd.Series([4.5, 7.2, -5.3, 3.6], index=["d", "b", "a", "c"])
obj
```

	0
d	4.5
b	7.2
a	-5.3
c	3.6

dtype: float64

```
obj2 = obj.reindex(["a", "b", "c", "d", "e"])
obj2
```

	0
a	-5.3
b	7.2
c	3.6
d	4.5
e	NaN

dtype: float64

```
obj3 = pd.Series(["blue", "purple", "yellow"], index=[0, 2, 4])
obj3
obj3.reindex(np.arange(6), method="ffill")
```

	0
0	blue
1	blue
2	purple
3	purple
4	yellow
5	yellow

dtype: object

```
frame = pd.DataFrame(np.arange(9).reshape((3, 3)),
                     index=["a", "c", "d"],
                     columns=["Ohio", "Texas", "California"])
frame
frame2 = frame.reindex(index=["a", "b", "c", "d"])
frame2
```

	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

Next steps: Generate code with frame2 View recommended plots New interactive sheet

```
states = ["Texas", "Utah", "California"]
frame.reindex(columns=states)
```

	Texas	Utah	California
a	1	NaN	2
c	4	NaN	5
d	7	NaN	8

```
frame.reindex(states, axis="columns")
```

	Texas	Utah	California
a	1	NaN	2
c	4	NaN	5
d	7	NaN	8

```
frame.loc[["a", "d", "c"], ["California", "Texas"]]
```

	California	Texas
a	2	1
d	8	7
c	5	4

```
obj = pd.Series(np.arange(5.), index=["a", "b", "c", "d", "e"])
obj
new_obj = obj.drop("c")
new_obj
obj.drop(["d", "c"])
```

	0
a	0.0
b	1.0
e	4.0

dtype: float64

```
data = pd.DataFrame(np.arange(16).reshape((4, 4)),
                    index=["Ohio", "Colorado", "Utah", "New York"],
                    columns=["one", "two", "three", "four"])
data
```

	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

Next steps: Generate code with data View recommended plots New interactive sheet

```
data.drop(index=["Colorado", "Ohio"])
```

	one	two	three	four
Utah	8	9	10	11
New York	12	13	14	15

```
data.drop(columns=["two"])
```

	one	three	four
Ohio	0	2	3
Colorado	4	6	7
Utah	8	10	11
New York	12	14	15

```
data.drop("two", axis=1)
data.drop(["two", "four"], axis="columns")
```

	one	three
Ohio	0	2
Colorado	4	6
Utah	8	10
New York	12	14

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

```
/tmp/ipython-input-54-1027675808.py:4: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version,
  obj[1]
/tmp/ipython-input-54-1027675808.py:7: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version,
  obj[[1, 3]]
```

	0
a	0.0
b	1.0

dtype: float64

```
obj.loc[["b", "a", "d"]]
```

	0
b	1.0
a	0.0
d	3.0

dtype: float64

```
obj1 = pd.Series([1, 2, 3], index=[2, 0, 1])
obj2 = pd.Series([1, 2, 3], index=["a", "b", "c"])
obj1
obj2
obj1[[0, 1, 2]]
obj2[[0, 1, 2]]
```

```
/tmp/ipython-input-56-2088379638.py:6: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version,
  obj2[[0, 1, 2]]
```

	0
a	1
b	2
c	3

dtype: int64

```
obj1.iloc[[0, 1, 2]]
obj2.iloc[[0, 1, 2]]
```

	0
a	1
b	2
c	3

dtype: int64

```
obj2.loc["b":"c"]
```

	0
b	2
c	3

dtype: int64

```
obj2.loc["b":"c"] = 5
obj2
```

```
   0
a  1
b  5
c  5

dtype: int64
```

```
data = pd.DataFrame(np.arange(16).reshape((4, 4)),
                    index=["Ohio", "Colorado", "Utah", "New York"],
                    columns=["one", "two", "three", "four"])
data
data["two"]
data[["three", "one"]]
```

```
          three  one
Ohio          2    0
Colorado      6    4
Utah         10    8
New York     14   12
```

```
data[:2]
data[data["three"] > 5]
```

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

```
data < 5
```

```
            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
```

```
data[data < 5] = 0
data
```

```
          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
```

Next steps: Generate code with data   View recommended plots   New interactive sheet

```
data
data.loc["Colorado"]
```

	Colorado
one	0
two	5
three	6
four	7

dtype: int64

```
data.loc[["Colorado", "New York"]]
```

	one	two	three	four
Colorado	0	5	6	7
New York	12	13	14	15

```
data.loc["Colorado", ["two", "three"]]
```

	Colorado
two	5
three	6

dtype: int64

```
data.iloc[2]
data.iloc[[2, 1]]
data.iloc[2, [3, 0, 1]]
data.iloc[[1, 2], [3, 0, 1]]
```

	four	one	two
Colorado	7	0	5
Utah	11	8	9

```
data.loc[:"Utah", "two"]
data.iloc[:, :3][data.three > 5]
```

	one	two	three
Colorado	0	5	6
Utah	8	9	10
New York	12	13	14

```
data.loc[data.three >= 2]
```

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

```
ser = pd.Series(np.arange(3.))
ser
```

	0
0	0.0
1	1.0
2	2.0

dtype: float64

```
ser
```

	0
0	0.0
1	1.0
2	2.0

dtype: float64

```
ser2 = pd.Series(np.arange(3.), index=["a", "b", "c"])
ser2[-1]
```

```
/tmp/ipython-input-75-821879068.py:2: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, i
  ser2[-1]
np.float64(2.0)
```

```
ser.iloc[-1]
```

```
np.float64(2.0)
```

```
ser[:2]
```

	0
0	0.0
1	1.0

dtype: float64

```
data.loc[:, "one"] = 1
data
data.iloc[2] = 5
data
data.loc[data["four"] > 5] = 3
data
```

	one	two	three	four
Ohio	1	0	0	0
Colorado	3	3	3	3
Utah	5	5	5	5
New York	3	3	3	3

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data.loc[data.three == 5]["three"] = 6
```

```
/tmp/ipython-input-79-867481848.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  data.loc[data.three == 5]["three"] = 6
```

```
data
```

	one	two	three	four
Ohio	1	0	0	0
Colorado	3	3	3	3
Utah	5	5	5	5
New York	3	3	3	3

Next steps: Generate code with data View recommended plots New interactive sheet

```
data.loc[data.three == 5, "three"] = 6
data
```

	one	two	three	four
Ohio	1	0	0	0
Colorado	3	3	3	3
Utah	5	5	6	5
New York	3	3	3	3

Next steps: Generate code with data View recommended plots New interactive sheet

```
s1 = pd.Series([7.3, -2.5, 3.4, 1.5], index=["a", "c", "d", "e"])
s2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1],
               index=["a", "c", "e", "f", "g"])
s1
s2
```

	0
a	-2.1
c	3.6
e	-1.5
f	4.0
g	3.1

dtype: float64

```
s1 + s2
```

	0
a	5.2
c	1.1
d	NaN
e	0.0
f	NaN
g	NaN

dtype: float64

```
df1 = pd.DataFrame(np.arange(9.).reshape((3, 3)), columns=list("bcd"),
                   index=["Ohio", "Texas", "Colorado"])
df2 = pd.DataFrame(np.arange(12.).reshape((4, 3)), columns=list("bde"),
                   index=["Utah", "Ohio", "Texas", "Oregon"])
df1
df2
```

	b	d	e
Utah	0.0	1.0	2.0
Ohio	3.0	4.0	5.0
Texas	6.0	7.0	8.0
Oregon	9.0	10.0	11.0

Next steps: Generate code with df2 View recommended plots New interactive sheet

```
df1 + df2
```

	b	c	d	e
Colorado	NaN	NaN	NaN	NaN
Ohio	3.0	NaN	6.0	NaN
Oregon	NaN	NaN	NaN	NaN
Texas	9.0	NaN	12.0	NaN
Utah	NaN	NaN	NaN	NaN

```
df1 = pd.DataFrame({"A": [1, 2]})
df2 = pd.DataFrame({"B": [3, 4]})
df1
df2
df1 + df2
```

	A	B
0	NaN	NaN
1	NaN	NaN

```
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
df1
df2
```

	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

Next steps: Generate code with df2 View recommended plots New interactive sheet

```
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

```
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

```
1 / df1
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

```
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

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

```
array([[0., 0., 0., 0.],
       [4., 4., 4., 4.],
       [8., 8., 8., 8.]])
```

```
frame = pd.DataFrame(np.arange(12.).reshape((4, 3)),
                     columns=list("bde"),
                     index=["Utah", "Ohio", "Texas", "Oregon"])
series = frame.iloc[0]
frame
series
```

	Utah
b	0.0
d	1.0
e	2.0

dtype: float64

```
frame - series
```

	b	d	e
Utah	0.0	0.0	0.0
Ohio	3.0	3.0	3.0
Texas	6.0	6.0	6.0
Oregon	9.0	9.0	9.0

```
series2 = pd.Series(np.arange(3), index=["b", "e", "f"])
series2
frame + series2
```

	b	d	e	f
Utah	0.0	NaN	3.0	NaN
Ohio	3.0	NaN	6.0	NaN
Texas	6.0	NaN	9.0	NaN
Oregon	9.0	NaN	12.0	NaN

```
series3 = frame["d"]
frame
series3
frame.sub(series3, axis="index")
```

	b	d	e
Utah	-1.0	0.0	1.0
Ohio	-1.0	0.0	1.0
Texas	-1.0	0.0	1.0
Oregon	-1.0	0.0	1.0

```
frame = pd.DataFrame(np.random.standard_normal((4, 3)),
                     columns=list("bde"),
                     index=["Utah", "Ohio", "Texas", "Oregon"])
frame
np.abs(frame)
```

	b	d	e
Utah	0.204708	0.478943	0.519439
Ohio	0.555730	1.965781	1.393406
Texas	0.092908	0.281746	0.769023
Oregon	1.246435	1.007189	1.296221

```
def f1(x):
    return x.max() - x.min()

frame.apply(f1)
```

	0
b	1.802165
d	1.684034
e	2.689627

dtype: float64

```
frame.apply(f1, axis="columns")
```

	0
Utah	0.998382
Ohio	2.521511
Texas	0.676115
Oregon	2.542656

dtype: float64

```
def f2(x):
    return pd.Series([x.min(), x.max()], index=["min", "max"])
frame.apply(f2)
```

	b	d	e
min	-0.555730	0.281746	-1.296221
max	1.246435	1.965781	1.393406

```
def my_format(x):
    return f"{x:.2f}"

frame.applymap(my_format)
```

```
/tmp/ipython-input-101-2705683181.py:4: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  frame.applymap(my_format)
```

	b	d	e
Utah	-0.20	0.48	-0.52
Ohio	-0.56	1.97	1.39
Texas	0.09	0.28	0.77
Oregon	1.25	1.01	-1.30

```
frame["e"].map(my_format)
```

	e
Utah	-0.52
Ohio	1.39
Texas	0.77
Oregon	-1.30

dtype: object

```
obj = pd.Series(np.arange(4), index=["d", "a", "b", "c"])
obj
obj.sort_index()
```

	0
a	1
b	2
c	3
d	0

dtype: int64

```
frame = pd.DataFrame(np.arange(8).reshape((2, 4)),
                     index=["three", "one"],
                     columns=["d", "a", "b", "c"])
frame
frame.sort_index()
frame.sort_index(axis="columns")
```

	a	b	c	d
three	1	2	3	0
one	5	6	7	4

```
frame.sort_index(axis="columns", ascending=False)
```

	d	c	b	a
three	0	3	2	1
one	4	7	6	5

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

```
    0
2  -3
3   2
0   4
1   7

dtype: int64
```

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

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

dtype: float64
```

```
obj.sort_values(na_position="first")
```

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

dtype: float64
```

```
frame = pd.DataFrame({"b": [4, 7, -3, 2], "a": [0, 1, 0, 1]})
frame
frame.sort_values("b")
```

```
    b  a
2  -3  0
3   2  1
0   4  0
1   7  1
```

```
frame.sort_values(["a", "b"])
```

```
    b  a
2  -3  0
0   4  0
3   2  1
1   7  1
```

```
obj = pd.Series([7, -5, 7, 4, 2, 0, 4])
obj.rank()
```

```
    0
0 6.5
1 1.0
2 6.5
3 4.5
4 3.0
5 2.0
6 4.5

dtype: float64
```

```
obj.rank(method="first")
```

```
    0
0 6.0
1 1.0
2 7.0
3 4.0
4 3.0
5 2.0
6 5.0

dtype: float64
```

```
obj.rank(ascending=False)
```

```
    0
0 1.5
1 7.0
2 1.5
3 3.5
4 5.0
5 6.0
6 3.5

dtype: float64
```

```
frame = pd.DataFrame({"b": [4.3, 7, -3, 2], "a": [0, 1, 0, 1],
                      "c": [-2, 5, 8, -2.5]})
frame
frame.rank(axis="columns")
```

```
    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
```

```
obj = pd.Series(np.arange(5), index=["a", "a", "b", "b", "c"])
obj
```

	0
a	0
a	1
b	2
b	3
c	4

dtype: int64

```
obj.index.is_unique
```

```
False
```

```
obj["a"]
obj["c"]
```

```
np.int64(4)
```

```
df = pd.DataFrame(np.random.standard_normal((5, 3)),
                  index=["a", "a", "b", "b", "c"])
df
df.loc["b"]
df.loc["c"]
```

	c
0	-0.577087
1	0.124121
2	0.302614

dtype: float64

```
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"])
df
```

	one	two
a	1.40	NaN
b	7.10	-4.5
c	NaN	NaN
d	0.75	-1.3

Next steps: Generate code with df   View recommended plots   New interactive sheet

```
df.sum()
```

	0
one	9.25
two	-5.80

dtype: float64

```
df.sum(axis="columns")
```

	0
a	1.40
b	2.60
c	0.00
d	-0.55

dtype: float64

```
df.sum(axis="index", skipna=False)
df.sum(axis="columns", skipna=False)
```

	0
a	NaN
b	2.60
c	NaN
d	-0.55

dtype: float64

```
df.mean(axis="columns")
```

	0
a	1.400
b	1.300
c	NaN
d	-0.275

dtype: float64

```
df.idxmax()
```

	0
one	b
two	d

dtype: object

```
df.cumsum()
```

	one	two
a	1.40	NaN
b	8.50	-4.5
c	NaN	NaN
d	9.25	-5.8

```
df.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

```
obj = pd.Series(["a", "a", "b", "c"] * 4)
obj.describe()
```

	0
count	16
unique	3
top	a
freq	8

dtype: object

```
obj = pd.Series(["c", "a", "d", "a", "a", "b", "b", "c", "c"])
```

```
uniques = obj.unique()
uniques
```

```
array(['c', 'a', 'd', 'b'], dtype=object)
```

```
obj.value_counts()
```

	count
c	3
a	3
b	2
d	1

dtype: int64

```
pd.value_counts(obj.to_numpy(), sort=False)
```

```
/tmp/ipython-input-132-164454357.py:1: FutureWarning: pandas.value_counts is deprecated and will be removed in a future version. Use pd.
  pd.value_counts(obj.to_numpy(), sort=False)
```

	count
c	3
a	3
d	1
b	2

dtype: int64

```
obj
mask = obj.isin(["b", "c"])
mask
obj[mask]
```

```
0
```