```python
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    4
1    7
2   -5
3    3
dtype: int64
```

```python
print(obj.array)
print(obj.index)
```

```
<NumpyExtensionArray>
[4, 7, -5, 3]
Length: 4, dtype: int64
RangeIndex(start=0, stop=4, step=1)
```

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

```
d    4
b    7
a   -5
c    3
dtype: int64
Index(['d', 'b', 'a', 'c'], dtype='object')
```

```python
print(obj2["a"])
obj2["d"] = 6#changes 4 to 6 for d
obj2[["c", "a", "d"]]#This selects the values at the indices "c", "a",
and "d" in that order
```

```
-5
```

```
c    3
a   -5
```

```
d    6
dtype: int64
```

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

```
d    6
b    7
a   -5
c    3
dtype: int64
d    6
b    7
c    3
dtype: int64
```

```
d     403.428793
b    1096.633158
a       0.006738
c      20.085537
dtype: float64
```

```
print("b" in obj2)
"e" in obj2
```

```
True
```

```
False
```

```
sdata = {"Ohio": 35000, "Texas": 71000,
         "Oregon": 16000, "Utah": 5000}
#converting a series from it by passing the dict
obj3 = pd.Series(sdata)
obj3
```

```
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"]#Nan as no value for
'California' while Utah not included in states no excluded from the
object
```

```
obj4 = pd.Series(sdata, index=states)
obj4
```

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

```
print(pd.isna(obj4))# function checks whether each element in a pandas
Series (or DataFrame) is "Not Available" (NA) or NaN
print(pd.notna(obj4))
```

```
California     True
Ohio          False
Oregon        False
Texas         False
dtype: bool
California    False
Ohio           True
Oregon         True
Texas          True
dtype: bool
```

```
obj4.isna()
```

```
California     True
Ohio          False
Oregon        False
Texas         False
dtype: bool
```

```
print(obj3)
print(obj4)
print(obj3 + obj4)
```

```
Ohio      35000
Texas     71000
Oregon    16000
Utah       5000
dtype: int64
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
dtype: float64
California         NaN
Ohio           70000.0
Oregon         32000.0
Texas         142000.0
```

```
Utah                   NaN
dtype: float64

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

state
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
Name: population, dtype: float64

print(obj)
obj.index = ["Bob", "Steve", "Jeff", "Ryan"]
print(obj)

0    4
1    7
2   -5
3    3
dtype: int64
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
#Dataframe represents a rectangular table of data and contains an
ordered collection of columnn each of which can be a different value
type

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

frame.head()#selects first five rows
```

```
    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
```

```
frame.tail()#last five rows
```

```
    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"])
print(frame2)
print(frame2.columns)#This prints the column names of the DataFrame.
```

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

```
frame2["state"]
```

```
0      Ohio
1      Ohio
2      Ohio
3    Nevada
4    Nevada
5    Nevada
Name: state, dtype: object
```

```
frame2.year
```

```
0    2000
1    2001
2    2002
3    2001
4    2002
5    2003
Name: year, dtype: int64
```

```
frame2.loc[1]#access the row at index 1 in the frame2 DataFrame.
```

```
year     2001
state    Ohio
pop       1.7
debt      NaN
Name: 1, dtype: object
```

```
frame2.iloc[2]
```

```
year     2002
state    Ohio
pop       3.6
debt      NaN
Name: 2, dtype: object
```

```
frame2['debt']=16.5
frame2
```

```
   year   state  pop  debt
0  2000    Ohio  1.5  16.5
1  2001    Ohio  1.7  16.5
2  2002    Ohio  3.6  16.5
3  2001  Nevada  2.4  16.5
4  2002  Nevada  2.9  16.5
5  2003  Nevada  3.2  16.5
```

```
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
```

```
val = pd.Series([-1.2, -1.5, -1.7],
                index=[2,4,5])
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  -1.2
3  2001  Nevada  2.4   NaN
4  2002  Nevada  2.9  -1.5
5  2003  Nevada  3.2  -1.7
```

```
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  -1.2     True
3  2001  Nevada  2.4   NaN    False
4  2002  Nevada  2.9  -1.5    False
5  2003  Nevada  3.2  -1.7    False
```

```
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
```

```
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)
```

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

frame3.to_numpy()

frame2.to_numpy()

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

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

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

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

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

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

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

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

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

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

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

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

```
new_obj = obj.drop("c")
new_obj
obj.drop(["d", "c"])
```

```
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
```

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

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

```
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]
```

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

```
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]]
```

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

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

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

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"]]

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

data < 5

data[data < 5] = 0
data

data
data.loc["Colorado"]

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

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

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

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

data.loc[data.three >= 2]

ser = pd.Series(np.arange(3.))
ser
ser[-1]

ser

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

ser.iloc[-1]

ser[:2]
```

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

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

data

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

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

s1 + s2

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

df1 + df2

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

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

df1 + df2

df1.add(df2, fill_value=0)

1 / df1
df1.rdiv(1)
```

```python
df1.reindex(columns=df2.columns, fill_value=0)

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

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

frame - series

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

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

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

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

frame.apply(f1)

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

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

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

frame.applymap(my_format)

frame["e"].map(my_format)

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

```
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")

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

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

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

obj.sort_values(na_position="first")

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

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

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

obj.rank(method="first")

obj.rank(ascending=False)

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")

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

obj.index.is_unique

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

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

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

df.sum()

df.sum(axis="columns")

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

df.mean(axis="columns")

df.idxmax()

df.cumsum()

df.describe()

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

price = pd.read_pickle("examples/yahoo_price.pkl")
volume = pd.read_pickle("examples/yahoo_volume.pkl")

returns = price.pct_change()
returns.tail()

returns["MSFT"].corr(returns["IBM"])
returns["MSFT"].cov(returns["IBM"])

returns.corr()
returns.cov()

returns.corrwith(returns["IBM"])

returns.corrwith(volume)

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

uniques = obj.unique()
uniques

obj.value_counts()

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

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

to_match = pd.Series(["c", "a", "b", "b", "c", "a"])
unique_vals = pd.Series(["c", "b", "a"])
```

```
indices = pd.Index(unique_vals).get_indexer(to_match)
indices

data = pd.DataFrame({"Qu1": [1, 3, 4, 3, 4],
                     "Qu2": [2, 3, 1, 2, 3],
                     "Qu3": [1, 5, 2, 4, 4]})
data

data["Qu1"].value_counts().sort_index()

result = data.apply(pd.value_counts).fillna(0)
result

data = pd.DataFrame({"a": [1, 1, 1, 2, 2], "b": [0, 0, 1, 0, 0]})
data
data.value_counts()



pd.options.display.max_rows = PREVIOUS_MAX_ROWS
```