Pandas toolkit Part 2

Syed Afroz Ali

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
s = pd.Series(np.random.randint(0, 7, size=10))
s
```

Out[2]:
```
0    0
1    6
2    5
3    6
4    5
5    5
6    2
7    2
8    4
9    2
dtype: int32
```

In [3]:
```
s.value_counts()
```

Out[3]:
```
5    3
2    3
6    2
0    1
4    1
dtype: int64
```

In [4]:
```
s = pd.Series(["A", "B", "C", "Aaba", "Baca", np.nan, "CABA", "dog", "cat"])
s.str.lower()
```

Out[4]:
```
0       a
1       b
2       c
3    aaba
4    baca
5     NaN
6    caba
7     dog
8     cat
dtype: object
```

In [5]:
```
df = pd.DataFrame(np.random.randn(10, 4))
df
```

Out[5]:

	0	1	2	3
0	0.423742	0.285896	-1.319679	0.106474
1	-0.917753	-0.612802	0.510887	-0.192883
2	-0.398895	1.744283	0.392814	1.252180
3	1.275378	-2.030125	1.377954	0.047816
4	0.541705	0.955300	-0.510238	0.690620
5	-0.968414	-0.495862	1.229128	0.968220
6	-1.478773	0.418985	1.010736	-1.494321
7	0.743932	-0.563562	0.986714	0.625697
8	-0.407228	-1.016760	0.617824	-0.370512
9	-0.129449	0.430960	0.192333	0.270365

In [6]:
```
pieces = [df[:3], df[3:7], df[7:]]
pd.concat(pieces)
```

Out[6]:

	0	1	2	3
0	0.423742	0.285896	-1.319679	0.106474
1	-0.917753	-0.612802	0.510887	-0.192883
2	-0.398895	1.744283	0.392814	1.252180
3	1.275378	-2.030125	1.377954	0.047816
4	0.541705	0.955300	-0.510238	0.690620
5	-0.968414	-0.495862	1.229128	0.968220
6	-1.478773	0.418985	1.010736	-1.494321
7	0.743932	-0.563562	0.986714	0.625697
8	-0.407228	-1.016760	0.617824	-0.370512
9	-0.129449	0.430960	0.192333	0.270365

In [9]:
```
left = pd.DataFrame({"key": ["foo", "foo"], "lval": [1, 2]})
right = pd.DataFrame({"key": ["foo", "foo"], "rval": [4, 5]})
left
```

Out[9]:

	key	lval
0	foo	1
1	foo	2

In [10]:
```
pd.merge(left, right, on="key")
```

Out[10]:

	key	lval	rval
0	foo	1	4
1	foo	1	5
2	foo	2	4
3	foo	2	5

In [11]:
```
left = pd.DataFrame({"key": ["foo", "bar"], "lval": [1, 2]})
right = pd.DataFrame({"key": ["foo", "bar"], "rval": [4, 5]})
left
```

Out[11]:

	key	lval
0	foo	1
1	bar	2

In [12]:
```
pd.merge(left, right, on="key")
```

Out[12]:

	key	lval	rval
0	foo	1	4
1	bar	2	5

In [13]:
```
df = pd.DataFrame({

"A": ["foo", "bar", "foo", "bar", "foo", "bar", "foo", "foo"],
"B": ["one", "one", "two", "three", "two", "two", "one", "three"],
"C": np.random.randn(8),
"D": np.random.randn(8)

     })
df
```

Out[13]:

	A	B	C	D
0	foo	one	-0.663656	-0.382136
1	bar	one	-1.314220	-1.048106
2	foo	two	0.391064	-0.693546
3	bar	three	-0.778172	0.695501
4	foo	two	1.171063	-0.340544
5	bar	two	-0.688497	0.840714
6	foo	one	-0.762570	-0.991745
7	foo	three	-1.449696	0.397626

In [14]:
```
df.groupby("A").sum()
```

Out[14]:

A	C	D
bar	-2.780889	0.488108
foo	-1.313795	-2.010344

In [15]:
```
df.groupby(["A", "B"]).sum()
```

Out[15]:

A	B	C	D
bar	one	-1.314220	-1.048106
	three	-0.778172	0.695501
	two	-0.688497	0.840714
foo	one	-1.426226	-1.373881
	three	-1.449696	0.397626
	two	1.562128	-1.034089

In [16]:
```
tuples = list(zip(*[
["bar", "bar", "baz", "baz", "foo", "foo", "qux", "qux"],
["one", "two", "one", "two", "one", "two", "one", "two"]
]))

index = pd.MultiIndex.from_tuples(tuples, names=["first", "second"])
df = pd.DataFrame(np.random.randn(8, 2), index=index, columns=["A", "B"])
df2 = df[:4]
df2
```

Out[16]:

first	second	A	B
bar	one	-2.314275	-0.404936
	two	0.370079	-0.128071
baz	one	1.472769	-1.160009
	two	-1.060644	-1.309345

In [17]:
```
stacked = df2.stack()
stacked
```

Out[17]:
```
first  second
bar    one     A   -2.314275
               B   -0.404936
       two     A    0.370079
               B   -0.128071
baz    one     A    1.472769
               B   -1.160009
       two     A   -1.060644
               B   -1.309345
dtype: float64
```

In [18]:
```
stacked.unstack()
```

Out[18]:

		A	B
first	**second**		
bar	**one**	-2.314275	-0.404936
	two	0.370079	-0.128071
baz	**one**	1.472769	-1.160009
	two	-1.060644	-1.309345

In [19]:
```
stacked.unstack(1)
```

Out[19]:

	second	**one**	**two**
first			
bar	**A**	-2.314275	0.370079
	B	-0.404936	-0.128071
baz	**A**	1.472769	-1.060644
	B	-1.160009	-1.309345

In [20]:
```
df = pd.DataFrame({

"A": ["one", "one", "two", "three"] * 3,
"B": ["A", "B", "C"] * 4,
"C": ["foo", "foo", "foo", "bar", "bar", "bar"] * 2,
"D": np.random.randn(12),
"E": np.random.randn(12)

})
df
```

Out[20]:

	A	B	C	D	E
0	one	A	foo	-1.156964	1.545813
1	one	B	foo	-1.975856	-0.172538
2	two	C	foo	-1.677984	-1.310321
3	three	A	bar	-1.159864	-0.761936
4	one	B	bar	0.408105	1.079825
5	one	C	bar	-0.717914	-1.394031
6	two	A	foo	0.620242	-0.099728
7	three	B	foo	0.315321	-0.546775
8	one	C	foo	-0.243819	1.283591
9	one	A	bar	0.718317	-0.877277
10	two	B	bar	0.747791	-0.748307
11	three	C	bar	-0.147173	0.632441

In [21]:
```
pd.pivot_table(df, values="D", index=["A", "B"], columns=["C"])
```

Out[21]:

	C	bar	foo
A	**B**		
one	**A**	0.718317	-1.156964
	B	0.408105	-1.975856
	C	-0.717914	-0.243819
three	**A**	-1.159864	NaN
	B	NaN	0.315321
	C	-0.147173	NaN
two	**A**	NaN	0.620242
	B	0.747791	NaN
	C	NaN	-1.677984

In [22]:
```
rng = pd.date_range("1/1/2012", periods=100, freq="S")
ts = pd.Series(np.random.randint(0, 500, len(rng)), index=rng)
ts.resample("5Min").sum()
```

Out[22]:
```
2012-01-01    26075
Freq: 5T, dtype: int32
```

In [23]:
```
rng = pd.date_range("3/6/2012 00:00", periods=5, freq="D")
ts = pd.Series(np.random.randn(len(rng)), rng)
ts
```

Out[23]:
```
2012-03-06    0.023926
2012-03-07   -0.602996
2012-03-08    0.686197
2012-03-09    0.535357
2012-03-10   -1.408127
Freq: D, dtype: float64
```

In [24]:
```
ts_utc = ts.tz_localize("UTC")
ts_utc
```

Out[24]:
```
2012-03-06 00:00:00+00:00    0.023926
2012-03-07 00:00:00+00:00   -0.602996
2012-03-08 00:00:00+00:00    0.686197
2012-03-09 00:00:00+00:00    0.535357
2012-03-10 00:00:00+00:00   -1.408127
Freq: D, dtype: float64
```

In [25]:
```
rng = pd.date_range("1/1/2012", periods=5, freq="M")
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
```

Out[25]:
```
2012-01-31   -1.062234
2012-02-29    0.942182
2012-03-31   -0.908925
2012-04-30    0.171292
2012-05-31   -2.773022
Freq: M, dtype: float64
```

In [26]:
```
ps = ts.to_period()
ps
```

Out[26]:
```
2012-01   -1.062234
2012-02    0.942182
2012-03   -0.908925
2012-04    0.171292
2012-05   -2.773022
Freq: M, dtype: float64
```

In [27]:
```
ps.to_timestamp()
```

Out[27]:
```
2012-01-01   -1.062234
2012-02-01    0.942182
2012-03-01   -0.908925
2012-04-01    0.171292
2012-05-01   -2.773022
Freq: MS, dtype: float64
```

In [28]:
```
prng = pd.period_range("1990Q1", "2000Q4", freq="Q-NOV")
ts = pd.Series(np.random.randn(len(prng)), prng)
ts.index = (prng.asfreq("M", "e") + 1).asfreq("H", "s") + 9
ts.head()
```

Out[28]:
```
1990-03-01 09:00   -0.745830
1990-06-01 09:00   -0.117445
1990-09-01 09:00   -0.189264
1990-12-01 09:00    0.541704
1991-03-01 09:00   -0.280971
Freq: H, dtype: float64
```

In [29]:
```
df = pd.DataFrame({"id": [1, 2, 3, 4, 5, 6], "raw_grade": ["a", "b", "b", "a",
df.head()
```

Out[29]:

	id	raw_grade
0	1	a
1	2	b
2	3	b
3	4	a
4	5	a

In [30]:
```
df["grade"] = df["raw_grade"].astype("category")
In [125]: df["grade"]
```

Out[30]:
```
0    a
1    b
2    b
3    a
4    a
5    e
Name: grade, dtype: category
Categories (3, object): ['a', 'b', 'e']
```

In [31]:
```
df["grade"].cat.categories = ["very good", "good", "very bad"]
```

In [32]:
```
df["grade"] = df["grade"].cat.set_categories(["very bad", "bad", "medium", "goo

df["grade"]
```

Out[32]:
```
0    very good
1         good
2         good
3    very good
4    very good
5     very bad
Name: grade, dtype: category
Categories (5, object): ['very bad', 'bad', 'medium', 'good', 'very good']
```

In [33]:
```
df.sort_values(by="grade")
```

Out[33]:

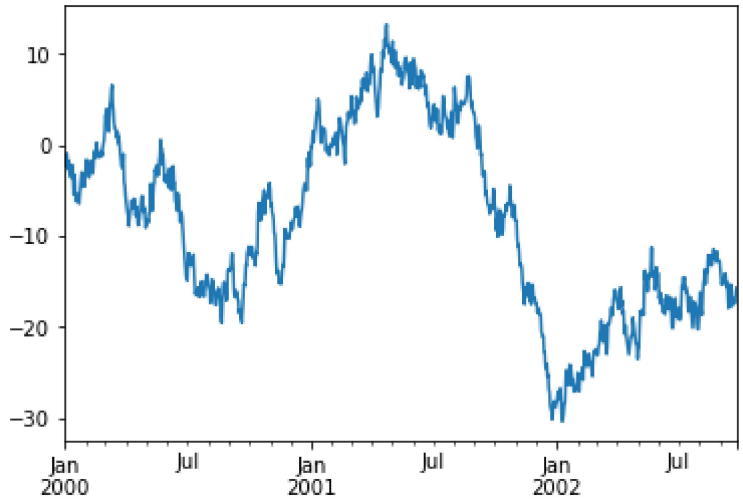
	id	raw_grade	grade
5	6	e	very bad
1	2	b	good
2	3	b	good
0	1	a	very good
3	4	a	very good
4	5	a	very good

In [34]:
```
df.groupby("grade").size()
```

Out[34]:
```
grade
very bad     1
bad          0
medium       0
good         2
very good    3
dtype: int64
```

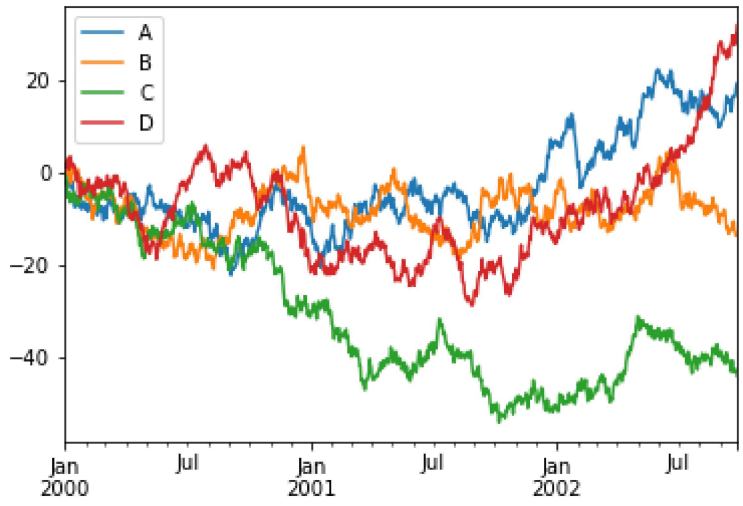
In [35]:
```
import matplotlib.pyplot as plt
plt.close("all")
```

In [36]:
```
ts = pd.Series(np.random.randn(1000), index=pd.date_range("1/1/2000",periods=10
ts = ts.cumsum()
ts.plot();
```



In [37]:
```
df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=["A", "B",
df = df.cumsum()
plt.figure();
df.plot();
plt.legend(loc='best');
```

```
<Figure size 432x288 with 0 Axes>
```



In [40]:
```
df.to_csv("foo.csv")
pd.read_csv("foo.csv")
```

Out[40]:

	Unnamed: 0	A	B	C	D
0	2000-01-01	0.070171	-0.384287	-0.333774	1.929496
1	2000-01-02	0.366578	-0.826180	0.025036	1.947304
2	2000-01-03	0.031841	-2.156061	-0.295405	3.846140
3	2000-01-04	-1.453527	-1.720770	0.681525	4.016645
4	2000-01-05	-0.506616	-1.748306	-0.953566	4.466979
...	...	...	...	...	...
995	2002-09-22	-8.952960	12.174642	-44.968475	21.234458
996	2002-09-23	-9.990733	13.157426	-43.955102	22.196782
997	2002-09-24	-10.370856	13.446153	-44.753827	23.111634
998	2002-09-25	-10.254712	13.043342	-44.844481	22.830099
999	2002-09-26	-9.240218	13.054899	-44.254345	22.267607

1000 rows × 5 columns

In [41]:
```
df.to_hdf("foo.h5", "df")
pd.read_hdf("foo.h5", "df")
```

Out[41]:

	A	B	C	D
2000-01-01	0.070171	-0.384287	-0.333774	1.929496
2000-01-02	0.366578	-0.826180	0.025036	1.947304
2000-01-03	0.031841	-2.156061	-0.295405	3.846140
2000-01-04	-1.453527	-1.720770	0.681525	4.016645
2000-01-05	-0.506616	-1.748306	-0.953566	4.466979
...	...	...	...	...
2002-09-22	-8.952960	12.174642	-44.968475	21.234458
2002-09-23	-9.990733	13.157426	-43.955102	22.196782
2002-09-24	-10.370856	13.446153	-44.753827	23.111634
2002-09-25	-10.254712	13.043342	-44.844481	22.830099
2002-09-26	-9.240218	13.054899	-44.254345	22.267607

1000 rows × 4 columns

In [42]:
```
df.to_excel("foo.xlsx", sheet_name="Sheet1")
pd.read_excel("foo.xlsx", "Sheet1", index_col=None, na_values=["NA"])
```

Out[42]:

	Unnamed: 0	A	B	C	D
0	2000-01-01	0.070171	-0.384287	-0.333774	1.929496
1	2000-01-02	0.366578	-0.826180	0.025036	1.947304
2	2000-01-03	0.031841	-2.156061	-0.295405	3.846140
3	2000-01-04	-1.453527	-1.720770	0.681525	4.016645
4	2000-01-05	-0.506616	-1.748306	-0.953566	4.466979
...	...	...	...	...	...
995	2002-09-22	-8.952960	12.174642	-44.968475	21.234458
996	2002-09-23	-9.990733	13.157426	-43.955102	22.196782
997	2002-09-24	-10.370856	13.446153	-44.753827	23.111634
998	2002-09-25	-10.254712	13.043342	-44.844481	22.830099
999	2002-09-26	-9.240218	13.054899	-44.254345	22.267607

1000 rows × 5 columns

In [43]:
```
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
s
```

Out[43]:
```
a   -0.168186
b   -2.300513
c   -1.303243
d    1.538910
e   -1.602989
dtype: float64
```

In [44]:
```
d = {"b": 1, "a": 0, "c": 2}
pd.Series(d)
```

Out[44]:
```
b    1
a    0
c    2
dtype: int64
```

In [45]:
```
d = {"a": 0.0, "b": 1.0, "c": 2.0}
pd.Series(d)
```

Out[45]:
```
a    0.0
b    1.0
c    2.0
dtype: float64
```

In [46]:
```
pd.Series(d, index=["b", "c", "d", "a"])
```

Out[46]:
```
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

In [47]:
```
pd.Series(5.0, index=["a", "b", "c", "d", "e"])
```

Out[47]:
```
a    5.0
b    5.0
c    5.0
d    5.0
e    5.0
dtype: float64
```

In [48]:
```
s[0]
```

Out[48]:
```
-0.1681860503643496
```

In [49]:
```
s[:3]
```

Out[49]:
```
a   -0.168186
b   -2.300513
c   -1.303243
dtype: float64
```

In [50]:
```
s[s > s.median()]
```

Out[50]:
```
a   -0.168186
d    1.538910
dtype: float64
```

In [51]:
```
s[[4, 3, 1]]
```

Out[51]:
```
e   -1.602989
d    1.538910
b   -2.300513
dtype: float64
```

In [52]:
```
np.exp(s)
```

Out[52]:
```
a    0.845197
b    0.100207
c    0.271649
d    4.659509
e    0.201294
dtype: float64
```

In [53]:
```
s.array
```

Out[53]:
```
<PandasArray>
[-0.1681860503643496, -2.3005130449484565, -1.3032428199311967,
  1.5389100807195653, -1.6029891733694224]
Length: 5, dtype: float64
```

In [54]:
```
s.to_numpy()
```

Out[54]:
```
array([-0.16818605, -2.30051304, -1.30324282,  1.53891008, -1.60298917])
```

In [55]:
```
s["a"]
```

Out[55]:
```
-0.1681860503643496
```

In [56]:
```
s["e"] = 12.0
s
```

Out[56]:
```
a    -0.168186
b    -2.300513
c    -1.303243
d     1.538910
e    12.000000
dtype: float64
```

In [57]:
```
np.exp(s)
```

Out[57]:
```
a         0.845197
b         0.100207
c         0.271649
d         4.659509
e    162754.791419
dtype: float64
```

In [58]:
```
s[1:] + s[:-1]
```

Out[58]:
```
a         NaN
b   -4.601026
c   -2.606486
d    3.077820
e         NaN
dtype: float64
```

In [59]:
```
s = pd.Series(np.random.randn(5), name="something")
s
```

Out[59]:
```
0   -0.522842
1    0.741873
2   -1.460176
3   -0.526032
4   -0.180085
Name: something, dtype: float64
```

In [60]:
```
s2 = s.rename("different")
s2.name
```

Out[60]:
```
'different'
```

In [61]:
```
d = {
"one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]),
"two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),
}

df = pd.DataFrame(d)
df
```

Out[61]:

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

In [62]:
```
pd.DataFrame(d, index=["d", "b", "a"])
```

Out[62]:

	one	two
d	NaN	4.0
b	2.0	2.0
a	1.0	1.0

In [63]:
```
pd.DataFrame(d, index=["d", "b", "a"], columns=["two", "three"])
```

Out[63]:

	two	three
d	4.0	NaN
b	2.0	NaN
a	1.0	NaN

In [64]:
```
df.index
```

Out[64]:
```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

In [65]:
```
df.columns
```

Out[65]:
```
Index(['one', 'two'], dtype='object')
```

In [66]:
```
d = {"one": [1.0, 2.0, 3.0, 4.0], "two": [4.0, 3.0, 2.0, 1.0]}
pd.DataFrame(d)
```

Out[66]:

	one	two
0	1.0	4.0
1	2.0	3.0
2	3.0	2.0
3	4.0	1.0

In [67]:
```
pd.DataFrame(d, index=["a", "b", "c", "d"])
```

Out[67]:

	one	two
a	1.0	4.0
b	2.0	3.0
c	3.0	2.0
d	4.0	1.0

In [68]:
```
data = np.zeros((2,), dtype=[("A", "i4"), ("B", "f4"), ("C", "a10")])
data[:] = [(1, 2.0, "Hello"), (2, 3.0, "World")]
pd.DataFrame(data)
```

Out[68]:

	A	B	C
0	1	2.0	b'Hello'
1	2	3.0	b'World'

In [69]:
```
pd.DataFrame(data, index=["first", "second"])
```

Out[69]:

	A	B	C
first	1	2.0	b'Hello'
second	2	3.0	b'World'

In [70]:
```
pd.DataFrame(data, columns=["C", "A", "B"])
```

Out[70]:

	C	A	B
0	b'Hello'	1	2.0
1	b'World'	2	3.0

In [71]:
```
data2 = [{"a": 1, "b": 2}, {"a": 5, "b": 10, "c": 20}]
pd.DataFrame(data2)
```

Out[71]:

	a	b	c
0	1	2	NaN
1	5	10	20.0

In [72]:
```
pd.DataFrame(data2, index=["first", "second"])
```

Out[72]:

	a	b	c
first	1	2	NaN
second	5	10	20.0

In [73]:
```
pd.DataFrame(data2, columns=["a", "b"])
```

Out[73]:

	a	b
0	1	2
1	5	10

In [74]:
```
pd.DataFrame({

("a", "b"): {("A", "B"): 1, ("A", "C"): 2},
("a", "a"): {("A", "C"): 3, ("A", "B"): 4},
("a", "c"): {("A", "B"): 5, ("A", "C"): 6},
("b", "a"): {("A", "C"): 7, ("A", "B"): 8},
("b", "b"): {("A", "D"): 9, ("A", "B"): 10}

})
```

Out[74]:

		a		b	
		b	a	c	a	b
	B	1.0	4.0	5.0	8.0	10.0
A	**C**	2.0	3.0	6.0	7.0	NaN
	D	NaN	NaN	NaN	NaN	9.0

In [75]:
```
from collections import namedtuple
Point = namedtuple("Point", "x y")
pd.DataFrame([Point(0, 0), Point(0, 3), (2, 3)])
```

Out[75]:

	x	y
0	0	0
1	0	3
2	2	3

In [76]:
```
Point3D = namedtuple("Point3D", "x y z")
```

In [77]:
```
pd.DataFrame([Point3D(0, 0, 0), Point3D(0, 3, 5), Point(2, 3)])
```

Out[77]:

	x	y	z
0	0	0	0.0
1	0	3	5.0
2	2	3	NaN

In [78]:
```
from dataclasses import make_dataclass
Point = make_dataclass("Point", [("x", int), ("y", int)])
pd.DataFrame([Point(0, 0), Point(0, 3), Point(2, 3)])
```

Out[78]:

	x	y
0	0	0
1	0	3
2	2	3

In [79]:
```
pd.DataFrame.from_dict(dict([("A", [1, 2, 3]), ("B", [4, 5, 6])]))
```

Out[79]:

	A	B
0	1	4
1	2	5
2	3	6

In [80]:
```
pd.DataFrame.from_dict(
dict([("A", [1, 2, 3]), ("B", [4, 5, 6])]),
orient="index",
columns=["one", "two", "three"],
)
```

Out[80]:

	one	two	three
A	1	2	3
B	4	5	6

In [81]:
```
pd.DataFrame.from_records(data, index="C")
```

Out[81]:

	A	B
C		
b'Hello'	1	2.0
b'World'	2	3.0

In [82]:
```
df["three"] = df["one"] * df["two"]
df["flag"] = df["one"] > 2
df
```

Out[82]:

	one	two	three	flag
a	1.0	1.0	1.0	False
b	2.0	2.0	4.0	False
c	3.0	3.0	9.0	True
d	NaN	4.0	NaN	False

In [83]:
```
del df["two"]
three = df.pop("three")
df
```

Out[83]:

	one	flag
a	1.0	False
b	2.0	False
c	3.0	True
d	NaN	False

In [84]:
```
df["foo"] = "bar"
df
```

Out[84]:

	one	flag	foo
a	1.0	False	bar
b	2.0	False	bar
c	3.0	True	bar
d	NaN	False	bar

In [85]:
```
df["one_trunc"] = df["one"][:2]
df
```

Out[85]:

	one	flag	foo	one_trunc
a	1.0	False	bar	1.0
b	2.0	False	bar	2.0
c	3.0	True	bar	NaN
d	NaN	False	bar	NaN

In [86]:
```
df.insert(1, "bar", df["one"])
df
```

Out[86]:

	one	bar	flag	foo	one_trunc
a	1.0	1.0	False	bar	1.0
b	2.0	2.0	False	bar	2.0
c	3.0	3.0	True	bar	NaN
d	NaN	NaN	False	bar	NaN

In [87]:
```
iris = pd.read_csv("Iris.csv")
iris.head()
```

Out[87]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [88]:
```
iris.assign(sepal_ratio=iris["SepalWidthCm"] / iris["SepalLengthCm"]).head()
```

Out[88]:

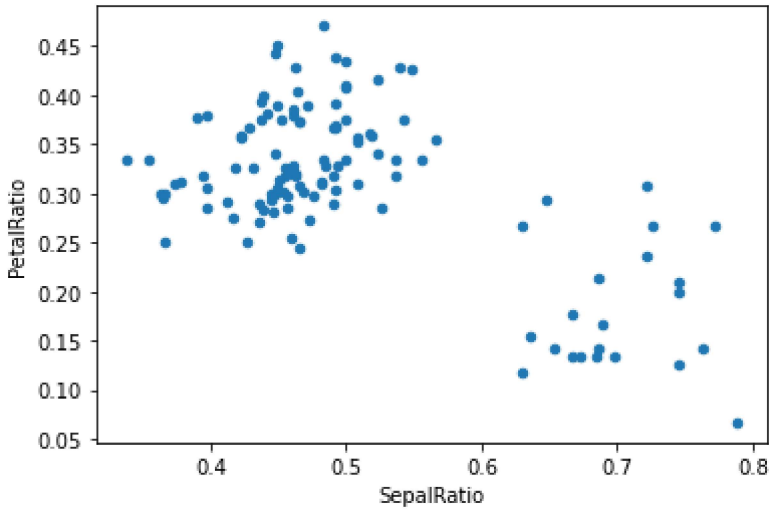
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	sepal_ratio
0	1	5.1	3.5	1.4	0.2	Iris-setosa	0.686275
1	2	4.9	3.0	1.4	0.2	Iris-setosa	0.612245
2	3	4.7	3.2	1.3	0.2	Iris-setosa	0.680851
3	4	4.6	3.1	1.5	0.2	Iris-setosa	0.673913
4	5	5.0	3.6	1.4	0.2	Iris-setosa	0.720000

In [89]:
```
iris.assign(sepal_ratio=lambda x: (x["SepalWidthCm"] / x["SepalLengthCm"])).hea
```

Out[89]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	sepal_ratio
0	1	5.1	3.5	1.4	0.2	Iris-setosa	0.686275
1	2	4.9	3.0	1.4	0.2	Iris-setosa	0.612245
2	3	4.7	3.2	1.3	0.2	Iris-setosa	0.680851
3	4	4.6	3.1	1.5	0.2	Iris-setosa	0.673913
4	5	5.0	3.6	1.4	0.2	Iris-setosa	0.720000

In [90]:
```
(iris.query("SepalLengthCm > 5").assign(
SepalRatio=lambda x: x.SepalWidthCm / x.SepalLengthCm,
PetalRatio=lambda x: x.PetalWidthCm / x.PetalLengthCm,
).plot(kind="scatter", x="SepalRatio", y="PetalRatio"));
```



In [91]:
```
dfa = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
dfa.assign(C=lambda x: x["A"] + x["B"], D=lambda x: x["A"] + x["C"])
```

Out[91]:

	A	B	C	D
0	1	4	5	6
1	2	5	7	9
2	3	6	9	12

In [92]:
```
df = pd.DataFrame(np.random.randn(10, 4), columns=["A", "B", "C", "D"])
df2 = pd.DataFrame(np.random.randn(7, 3), columns=["A", "B", "C"])
df + df2
```

Out[92]:

	A	B	C	D
0	2.420866	1.658919	-0.004957	NaN
1	0.230904	-1.496995	0.531114	NaN
2	0.387801	-2.293522	-0.940368	NaN
3	0.198035	-1.598423	0.608666	NaN
4	-0.750591	1.431920	-1.540342	NaN
5	0.144717	-2.595985	0.175171	NaN
6	0.396199	0.681542	3.810757	NaN
7	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN

In [93]:
```
df1 = pd.DataFrame({"a": [1, 0, 1], "b": [0, 1, 1]}, dtype=bool)
df2 = pd.DataFrame({"a": [0, 1, 1], "b": [1, 1, 0]}, dtype=bool)
df1 & df2
```

Out[93]:

	a	b
0	False	False
1	False	True
2	True	False

In [94]:
```
df1 | df2
```

Out[94]:

	a	b
0	True	True
1	True	True
2	True	True

In [95]:
```
df1 ^ df2
```

Out[95]:

	a	b
0	True	True
1	True	False
2	False	True

In [96]:
```
-df1
```

Out[96]:

	a	b
0	False	True
1	True	False
2	False	False

In [97]:
```
np.exp(df)
```

Out[97]:

	A	B	C	D
0	4.850797	7.643395	1.172958	1.574652
1	4.009842	0.933421	7.619213	0.661011
2	1.349996	0.193248	0.811424	1.976038
3	0.941819	0.520655	0.461768	2.772299
4	0.894556	1.922714	0.224244	3.327236
5	1.056101	0.233735	0.556385	0.303122
6	2.077379	0.660008	7.266333	3.149107
7	0.800964	1.414727	0.712628	0.400347
8	1.998324	2.184248	3.783204	0.802994
9	0.723489	8.619131	0.100072	2.146063

In [98]:
```
ser = pd.Series([1, 2, 3, 4])
np.exp(ser)
```

Out[98]:
```
0     2.718282
1     7.389056
2    20.085537
3    54.598150
dtype: float64
```

In [99]:
```
ser1 = pd.Series([1, 2, 3], index=["a", "b", "c"])
ser2 = pd.Series([1, 3, 5], index=["b", "a", "c"])
ser1
```

Out[99]:
```
a    1
b    2
c    3
dtype: int64
```

In [100]:
```
np.remainder(ser1, ser2)
```

Out[100]:
```
a    1
b    0
c    3
dtype: int64
```

In [101]:
```
ser3 = pd.Series([2, 4, 6], index=["b", "c", "d"])
np.remainder(ser1, ser3)
```

Out[101]:
```
a    NaN
b    0.0
c    3.0
d    NaN
dtype: float64
```

In [102]:
```
ser = pd.Series([1, 2, 3])
idx = pd.Index([4, 5, 6])
np.maximum(ser, idx)
```

Out[102]:
```
0    4
1    5
2    6
dtype: int64
```

In [104]:
```
pd.DataFrame(np.random.randn(3, 12))
```

Out[104]:

	0	1	2	3	4	5	6	7	8
0	-0.501245	0.571128	-0.508366	-0.326784	0.636010	1.006448	-0.910515	-0.502634	1.306846
1	-0.555579	0.528680	0.244514	-0.921669	-0.942707	-1.253826	-1.811112	-0.900820	0.616784
2	-0.576731	-0.872153	2.207928	-0.505467	-0.130966	1.685892	-1.459214	-0.262680	-0.229437

In [105]:
```
pd.set_option("display.width", 40) # default is 80
pd.DataFrame(np.random.randn(3, 12))
```

Out[105]:

	0	1	2	3	4	5	6	7	8
0	0.022773	-1.401868	-0.970989	2.171251	0.414099	0.151816	-1.110283	-0.449167	1.714537
1	-0.789318	-0.265809	-0.678194	-2.483981	0.364979	-0.704912	-0.847870	-1.562679	1.817926
2	-0.841586	-1.769016	1.341991	-0.463400	1.139975	0.237341	-0.223064	-0.146064	0.253593

In [106]:
```
datafile = {
"filename": ["filename_01", "filename_02"],
"path": [
"media/user_name/storage/folder_01/filename_01",
"media/user_name/storage/folder_02/filename_02",]
}

pd.set_option("display.max_colwidth", 30)
pd.DataFrame(datafile)
```

Out[106]:

	filename	path
0	filename_01	media/user_name/storage/fo...
1	filename_02	media/user_name/storage/fo...

In [107]:
```
pd.set_option("display.max_colwidth", 100)
pd.DataFrame(datafile)
```

Out[107]:

	filename	path
0	filename_01	media/user_name/storage/folder_01/filename_01
1	filename_02	media/user_name/storage/folder_02/filename_02

In [ ]:
```
df = pd.DataFrame({"foo1": np.random.randn(5), "foo2": np.random.randn(5)})
df
```

In [108]:
```
index = pd.date_range("1/1/2000", periods=8)
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
df = pd.DataFrame(np.random.randn(8, 3), index=index, columns=["A", "B", "C"])
```

In [109]:
```
long_series = pd.Series(np.random.randn(1000))
long_series.head()
```

Out[109]:
```
0   -0.026014
1   -1.002232
2    0.435552
3   -0.516468
4    0.919732
dtype: float64
```

In [110]:
```
df[:2]
```

Out[110]:

	A	B	C
2000-01-01	-0.896867	0.519293	0.574513
2000-01-02	1.499996	0.210594	0.004202

In [111]:
```
df.columns = [x.lower() for x in df.columns]
df
```

Out[111]:

	a	b	c
2000-01-01	-0.896867	0.519293	0.574513
2000-01-02	1.499996	0.210594	0.004202
2000-01-03	0.670616	0.012021	-1.118078
2000-01-04	-0.708142	-0.351169	-0.596160
2000-01-05	0.571710	-1.264462	-0.999771
2000-01-06	-0.355902	-0.458909	1.478698
2000-01-07	0.242235	0.194339	-0.864089
2000-01-08	0.073826	0.314112	1.816110

In [112]:
```
s.array
```

Out[112]:
```
<PandasArray>
[-0.9915161162074533,
   0.816408648335188,
  1.1267915421666856,
 0.48624698933925486,
 0.26060288152211175]
Length: 5, dtype: float64
```

In [113]:
```
s.index.array
```

Out[113]:
```
<PandasArray>
['a', 'b', 'c', 'd', 'e']
Length: 5, dtype: object
```

In [114]:
```
s.to_numpy()
```

Out[114]:
```
array([-0.99151612,  0.81640865,  1.12679154,  0.48624699,  0.26060288])
```

In [115]:
```
np.asarray(s)
```

Out[115]:
```
array([-0.99151612,  0.81640865,  1.12679154,  0.48624699,  0.26060288])
```

In [116]:
```
ser = pd.Series(pd.date_range("2000", periods=2, tz="CET"))
ser.to_numpy(dtype=object)
```

Out[116]:
```
array([Timestamp('2000-01-01 00:00:00+0100', tz='CET'),
       Timestamp('2000-01-02 00:00:00+0100', tz='CET')], dtype=object)
```

In [117]:
```
pd.set_option("compute.use_bottleneck", False)
pd.set_option("compute.use_numexpr", False)
```

In [118]:
```
df = pd.DataFrame({
"one": pd.Series(np.random.randn(3), index=["a", "b", "c"]),
"two": pd.Series(np.random.randn(4), index=["a", "b", "c", "d"]),
"three": pd.Series(np.random.randn(3), index=["b", "c", "d"]),
})

df
```

Out[118]:

	one	two	three
a	-1.171896	-1.181811	NaN
b	0.758395	-0.897135	-1.107687
c	-0.844188	0.018352	2.354688
d	NaN	1.613328	-0.269916

In [119]:
```
row = df.iloc[1]
column = df["two"]
df.sub(row, axis="columns")
```

Out[119]:

	one	two	three
a	-1.930291	-0.284676	NaN
b	0.000000	0.000000	0.000000
c	-1.602583	0.915488	3.462375
d	NaN	2.510464	0.837772

In [120]:
```
df.sub(row, axis=1)
```

Out[120]:

	one	two	three
a	-1.930291	-0.284676	NaN
b	0.000000	0.000000	0.000000
c	-1.602583	0.915488	3.462375
d	NaN	2.510464	0.837772

In [121]:
```
df.sub(column, axis="index")
```

Out[121]:

	one	two	three
a	0.009915	0.0	NaN
b	1.655531	0.0	-0.210552
c	-0.862540	0.0	2.336335
d	NaN	0.0	-1.883244

In [122]:
```
df.sub(column, axis=0)
```

Out[122]:

	one	two	three
a	0.009915	0.0	NaN
b	1.655531	0.0	-0.210552
c	-0.862540	0.0	2.336335
d	NaN	0.0	-1.883244

In [123]:
```
dfmi = df.copy()
In [27]: dfmi.index = pd.MultiIndex.from_tuples(
[(1, "a"), (1, "b"), (1, "c"), (2, "a")], names=["first", "second"]
)

dfmi.sub(column, axis=0, level="second")
```

Out[123]:

		one	two	three
first	**second**			
	a	0.009915	0.000000	NaN
1	**b**	1.655531	0.000000	-0.210552
	c	-0.862540	0.000000	2.336335
2	**a**	NaN	2.795139	0.911896

In [124]:
```
pd.Series(np.arange(10))
```

Out[124]:
```
0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
8    8
9    9
dtype: int32
```

In [125]:
```
div, rem = divmod(df,3)
div
```

Out[125]:

	one	two	three
a	-1.0	-1.0	NaN
b	0.0	-1.0	-1.0
c	-1.0	0.0	0.0
d	NaN	0.0	-1.0

In [126]:
```
idx = pd.Index(np.arange(10))
idx
```

Out[126]:
```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8,
            9],
           dtype='int64')
```

In [127]:
```
div, rem = divmod(idx, 3)
div
```

Out[127]:
```
Int64Index([0, 0, 0, 1, 1, 1, 2, 2, 2,
            3],
           dtype='int64')
```

In [128]:
```
df.gt(df)
```

Out[128]:

	one	two	three
a	False	False	False
b	False	False	False
c	False	False	False
d	False	False	False

In [129]:
```
(df > 0).all()
```

Out[129]:
```
one      False
two      False
three    False
dtype: bool
```

In [130]:
```
(df > 0).any()
```

Out[130]:
```
one      True
two      True
three    True
dtype: bool
```

In [131]:
```
(df > 0).any().any()
```

Out[131]:
```
True
```

In [132]:
```
pd.DataFrame(columns=list("ABC")).empty
```

Out[132]:
```
True
```

In [133]:
```
(df + df == df * 2).all()
```

Out[133]:
```
one      False
two       True
three    False
dtype: bool
```

In [134]:
```
(df + df).equals(df * 2)
```

Out[134]:
```
True
```

In [135]:
```
df1 = pd.DataFrame({"col": ["foo", 0, np.nan]})
df2 = pd.DataFrame({"col": [np.nan, 0, "foo"]}, index=[2, 1, 0])
df1.equals(df2)
```

Out[135]:
```
False
```

In [136]:
```
pd.Series(["foo", "bar", "baz"]) == "foo"
```

Out[136]:
```
0     True
1    False
2    False
dtype: bool
```

In [137]:
```
pd.Index(["foo", "bar", "baz"]) == "foo"
```

Out[137]:
```
array([ True, False, False])
```

In [138]:
```
pd.Series(["foo", "bar", "baz"]) == pd.Index(["foo", "bar", "qux"])
```

Out[138]:
```
0     True
1     True
2    False
dtype: bool
```

In [139]:
```
pd.Series(["foo", "bar", "baz"]) == np.array(["foo", "bar", "qux"])
```

Out[139]:
```
0     True
1     True
2    False
dtype: bool
```

In [140]:
```
np.array([1, 2, 3]) == np.array([2])
```

Out[140]:
```
array([False,  True, False])
```

In [141]:
```
df1 = pd.DataFrame(
{"A": [1.0, np.nan, 3.0, 5.0, np.nan], "B": [np.nan, 2.0, 3.0, np.nan, 6.0]}
)

df2 = pd.DataFrame({
"A": [5.0, 2.0, 4.0, np.nan, 3.0, 7.0],
"B": [np.nan, np.nan, 3.0, 4.0, 6.0, 8.0],
})

df1
```

Out[141]:

	A	B
0	1.0	NaN
1	NaN	2.0
2	3.0	3.0
3	5.0	NaN
4	NaN	6.0

In [142]:
```
df1.combine_first(df2)
```

Out[142]:

	A	B
0	1.0	NaN
1	2.0	2.0
2	3.0	3.0
3	5.0	4.0
4	3.0	6.0
5	7.0	8.0

In [143]:
```
def combiner(x, y):
    return np.where(pd.isna(x), y, x)
df1.combine(df2, combiner)
```

Out[143]:

	A	B
0	1.0	NaN
1	2.0	2.0
2	3.0	3.0
3	5.0	4.0
4	3.0	6.0
5	7.0	8.0

In [144]:
```
df.sum(0, skipna=False)
```

Out[144]:
```
one           NaN
two     -0.447266
three         NaN
dtype: float64
```

In [145]:
```
df.sum(axis=1, skipna=True)
```

Out[145]:
```
a   -2.353707
b   -1.246427
c    1.528852
d    1.343413
dtype: float64
```

In [146]:
```
ts_stand = (df - df.mean()) / df.std()
ts_stand.std()
```

Out[146]:
```
one      1.0
two      1.0
three    1.0
dtype: float64
```

In [147]:
```
xs_stand = df.sub(df.mean(1), axis=0).div(df.std(1), axis=0)
xs_stand.std(1)
```

Out[147]:
```
a    1.0
b    1.0
c    1.0
d    1.0
dtype: float64
```

In [148]:
```
np.mean(df["one"])
```

Out[148]:
```
-0.41922927314676367
```

In [149]:
```
np.mean(df["one"].to_numpy())
```

Out[149]:
```
nan
```

In [150]:
```
series = pd.Series(np.random.randn(500))
series[20:500] = np.nan
series[10:20] = 5
series.nunique()
```

Out[150]:
```
11
```

In [151]:
```
series = pd.Series(np.random.randn(1000))
series[::2] = np.nan
series.describe()
```

Out[151]:
```
count    500.000000
mean      -0.007995
std        0.963051
min       -4.036894
25%       -0.617967
50%        0.044174
75%        0.671260
max        2.694971
dtype: float64
```

In [152]:
```
frame = pd.DataFrame(np.random.randn(1000, 5), columns=["a", "b", "c", "d", "e"
frame.iloc[::2] = np.nan
frame.describe()
```

Out[152]:

	a	b	c	d	e
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	-0.033536	-0.000459	0.063157	0.059826	0.043546
std	0.973013	0.936975	1.046047	1.024661	0.978513
min	-3.491812	-2.591516	-2.787851	-2.796227	-3.234868
25%	-0.637533	-0.607886	-0.712782	-0.616478	-0.564365
50%	-0.022158	0.024663	0.037883	0.044762	-0.024644
75%	0.666086	0.639350	0.758282	0.721884	0.685598
max	2.566799	3.027822	2.766063	3.472135	3.077083

In [153]:
```
series.describe(percentiles=[0.05, 0.25, 0.75, 0.95])
```

Out[153]:
```
count    500.000000
mean      -0.007995
std        0.963051
min       -4.036894
5%        -1.707012
25%       -0.617967
50%        0.044174
75%        0.671260
95%        1.444216
max        2.694971
dtype: float64
```

In [154]:
```
s = pd.Series(["a", "a", "b", "b", "a", "a", np.nan, "c", "d", "a"])
s.describe()
```

Out[154]:
```
count     9
unique    4
top       a
freq      5
dtype: object
```

Syed Afroz Ali