Getting Started with pandas

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

In [2]:
```
from pandas import Series, DataFrame
```

In [3]:
```
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
np.set_printoptions(precision=4, suppress=True)
```

Introduction to pandas Data Structures

Series

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

Out[4]:
```
0    4
1    7
2   -5
3    3
dtype: int64
```

In [5]:
```
obj.values
```

Out[5]:
```
array([ 4,  7, -5,  3], dtype=int64)
```

In [6]:
```
obj.index  # like range(4)
```

Out[6]:
```
RangeIndex(start=0, stop=4, step=1)
```

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

Out[7]:
```
d    4
b    7
a   -5
c    3
dtype: int64
```

In [8]:
```
obj2.index
```

Out[8]:
```
Index(['d', 'b', 'a', 'c'], dtype='object')
```

In [9]:
```
obj2['a']
```

Out[9]:
```
-5
```

In [10]:
```
obj2['d'] = 6
```

In [11]:
```
obj2[['c', 'a', 'd']]
```

Out[11]:
```
c    3
a   -5
d    6
dtype: int64
```

In [12]:
```
obj2[obj2 > 0]
```

Out[12]:
```
d    6
b    7
c    3
dtype: int64
```

In [13]:
```
obj2 * 2
```

Out[13]:
```
d    12
b    14
a   -10
c     6
dtype: int64
```

In [14]:
```
np.exp(obj2)
```

Out[14]:
```
d     403.428793
b    1096.633158
a       0.006738
c      20.085537
dtype: float64
```

In [15]:
```
'b' in obj2
```

Out[15]:
```
True
```

In [16]:
```
'e' in obj2
```

Out[16]:
```
False
```

In [17]:
```
sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}
obj3 = pd.Series(sdata)
obj3
```

Out[17]:
```
Ohio      35000
Texas     71000
Oregon    16000
Utah       5000
dtype: int64
```

In [18]:
```
states = ['California', 'Ohio', 'Oregon', 'Texas']
obj4 = pd.Series(sdata, index=states)
obj4
```

Out[18]:
```
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
dtype: float64
```

In [19]:
```
pd.isnull(obj4)
```

Out[19]:
```
California     True
Ohio          False
Oregon        False
Texas         False
dtype: bool
```

In [20]:
```
pd.notnull(obj4)
```

Out[20]:
```
California    False
Ohio           True
Oregon         True
Texas          True
dtype: bool
```

In [21]:
```
obj4.isnull()
```

Out[21]:
```
California     True
Ohio          False
Oregon        False
Texas         False
dtype: bool
```

In [22]:
```
obj3
```

Out[22]:
```
Ohio      35000
Texas     71000
Oregon    16000
Utah       5000
dtype: int64
```

In [23]:
```
obj4
```

Out[23]:
```
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
dtype: float64
```

In [24]:
```
obj3 + obj4
```

Out[24]:
```
California         NaN
Ohio           70000.0
Oregon         32000.0
Texas         142000.0
Utah               NaN
dtype: float64
```

In [25]:
```
obj4.name = 'population'
obj4.index.name = 'state'
obj4
```

Out[25]:
```
state
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
Name: population, dtype: float64
```

In [26]:
```
obj
```

Out[26]:
```
0    4
1    7
2   -5
3    3
dtype: int64
```

In [27]:
```
obj.index = ['Bob', 'Steve', 'Jeff', 'Ryan']
obj
```

Out[27]:
```
Bob      4
Steve    7
Jeff    -5
Ryan     3
dtype: int64
```

DataFrame

In [28]:
```
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)
```

In [29]:
```
frame
```

Out[29]:

	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

In [30]:
```
frame.head()
```

Out[30]:

	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

In [31]:
```
pd.DataFrame(data, columns=['year', 'state', 'pop'])
```

Out[31]:

	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

In [32]:
```
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                      index=['one', 'two', 'three', 'four',
                             'five', 'six'])
frame2
```

Out[32]:

	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

In [33]:
```
frame2.columns
```

Out[33]:
```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

In [34]:
```
frame2['state']
```

Out[34]:
```
one        Ohio
two        Ohio
three      Ohio
four     Nevada
five     Nevada
six      Nevada
Name: state, dtype: object
```

In [35]:
```
frame2.year
```

Out[35]:
```
one      2000
two      2001
three    2002
four     2001
five     2002
six      2003
Name: year, dtype: int64
```

In [36]:
```
frame2.loc['three']
```

Out[36]:
```
year     2002
state    Ohio
pop       3.6
debt      NaN
Name: three, dtype: object
```

In [37]:
```
frame2['debt'] = 16.5
frame2
```

Out[37]:

	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

In [38]:
```
frame2['debt'] = np.arange(6.)
frame2
```

Out[38]:

	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

In [39]:
```
val = pd.Series([-1.2, -1.5, -1.7], index=['two', 'four', 'five'])
frame2['debt'] = val
frame2
```

Out[39]:

	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

In [40]:
```
frame2['eastern'] = frame2.state == 'Ohio'
frame2
```

Out[40]:

	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

In [41]:
```
del frame2['eastern']
frame2.columns
```

Out[41]:
```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

In [42]:
```
pop = {'Nevada': {2001: 2.4, 2002: 2.9},
       'Ohio': {2000: 1.5, 2001: 1.7, 2002: 3.6}}
```

In [43]:
```
frame3 = pd.DataFrame(pop)
frame3
```

Out[43]:

	Nevada	Ohio
2001	2.4	1.7
2002	2.9	3.6
2000	NaN	1.5

In [44]:
```
frame3.T
```

Out[44]:

	2001	2002	2000
Nevada	2.4	2.9	NaN
Ohio	1.7	3.6	1.5

In [45]:
```
pd.DataFrame(pop, index=[2001, 2002, 2003])
```

Out[45]:

	Nevada	Ohio
2001	2.4	1.7
2002	2.9	3.6
2003	NaN	NaN

In [47]:
```
pda = {'Ohio': frame3['Ohio'][:-1]}
pda
```

Out[47]:
```
{'Ohio': 2001    1.7
 2002    3.6
 Name: Ohio, dtype: float64}
```

In [48]:
```
pdata = {'Ohio': frame3['Ohio'][:-1],
         'Nevada': frame3['Nevada'][:2]}
pd.DataFrame(pdata)
```

Out[48]:

	Ohio	Nevada
2001	1.7	2.4
2002	3.6	2.9

In [49]:
```
frame3.index.name = 'year'; frame3.columns.name = 'state'
frame3
```

Out[49]:

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

In [50]:
```
frame3.values
```

Out[50]:
```
array([[2.4, 1.7],
       [2.9, 3.6],
       [nan, 1.5]])
```

In [51]:
```
frame2.values
```

Out[51]:
```
array([[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]], dtype=object)
```

Index Objects

In [52]:
```
obj = pd.Series(range(3), index=['a', 'b', 'c'])
obj
```

Out[52]:
```
a    0
b    1
c    2
dtype: int64
```

In [53]:
```
index = obj.index
```

In [54]:
```
index
```

Out[54]:
```
Index(['a', 'b', 'c'], dtype='object')
```

In [55]:
```
index[1:]
```

Out[55]:
```
Index(['b', 'c'], dtype='object')
```

In [56]:
```
index[1] = 'd'  # TypeError
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16832\748847593.py in <module>
----> 1 index[1] = 'd'  # TypeError

C:\PythonDSA\anaconda3\lib\site-packages\pandas\core\indexes\base.py in __setitem__(self, key, value)
   5033     @final
   5034     def __setitem__(self, key, value):
-> 5035         raise TypeError("Index does not support mutable operations")
   5036 
   5037     def __getitem__(self, key):

TypeError: Index does not support mutable operations
```

In [57]:
```
labels = pd.Index(np.arange(3))
labels
```
Out[57]:
```
Int64Index([0, 1, 2], dtype='int64')
```

In [59]:
```
obj2
```
Out[59]:
```
d    6
b    7
a   -5
c    3
dtype: int64
```

In [61]:
```
obj2 = pd.Series([1.5,-2.5,0], index=labels)
obj2
```
Out[61]:
```
0    1.5
1   -2.5
2    0.0
dtype: float64
```

In [62]:
```
obj2.index is labels
```
Out[62]:
```
True
```

In [64]:
```
frame3
```
Out[64]:

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

In [65]:
```
frame3.columns
```
Out[65]:
```
Index(['Nevada', 'Ohio'], dtype='object', name='state')
```

In [66]:
```
'Ohio' in frame3.columns
```
Out[66]:
```
True
```

In [67]:
```
2003 in frame3.index
```
Out[67]:
```
False
```

In [68]:
```
dup_labels = pd.Index(['foo', 'foo', 'bar', 'bar'])
dup_labels
```
Out[68]:
```
Index(['foo', 'foo', 'bar', 'bar'], dtype='object')
```

Essential Functionality

Reindexing

In [69]:
```
obj = pd.Series([4.5, 7.2, -5.3, 3.6], index=['d', 'b', 'a', 'c'])
obj
```

Out[69]:
```
d    4.5
b    7.2
a   -5.3
c    3.6
dtype: float64
```

In [70]:
```
obj2 = obj.reindex(['a', 'b', 'c', 'd', 'e'])
obj2
```

Out[70]:
```
a   -5.3
b    7.2
c    3.6
d    4.5
e    NaN
dtype: float64
```

In [71]:
```
obj3 = pd.Series(['blue', 'purple', 'yellow'], index=[0, 2, 4])
obj3
```

Out[71]:
```
0      blue
2    purple
4    yellow
dtype: object
```

In [72]:
```
obj3.reindex(range(6), method='ffill')
```

Out[72]:
```
0      blue
1      blue
2    purple
3    purple
4    yellow
5    yellow
dtype: object
```

In [95]:
```
frame = pd.DataFrame(np.arange(9).reshape((3, 3)),
                     index=['a', 'c', 'd'],
                     columns=['Ohio', 'Texas', 'California'])
frame
```

Out[95]:

	Ohio	Texas	California
a	0	1	2
c	3	4	5
d	6	7	8

In [75]:
```
frame2 = frame.reindex(['a', 'b', 'c', 'd'])
frame2
```

Out[75]:

	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

In [93]:
```
states = ['Texas', 'Utah', 'California']
frame.reindex(columns=states)
```

Out[93]:

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

In [98]:
```
frame_reindexed = frame.reindex(index=['a', 'b', 'c', 'd'], columns=states, fill_va

print(frame_reindexed)
```

```
   Texas  Utah  California
a    1.0   NaN         2.0
b    NaN   NaN         NaN
c    4.0   NaN         5.0
d    7.0   NaN         8.0
```

Dropping Entries from an Axis

In [99]:
```
obj = pd.Series(np.arange(5.), index=['a', 'b', 'c', 'd', 'e'])
obj
```

Out[99]:
```
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
```

In [100…
```
new_obj = obj.drop('c')
new_obj
```

Out[100]:
```
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

In [101…
```
obj.drop(['d', 'c'])
```

Out[101]:
```
a    0.0
b    1.0
e    4.0
dtype: float64
```

In [102…
```
data = pd.DataFrame(np.arange(16).reshape((4, 4)),
                    index=['Ohio', 'Colorado', 'Utah', 'New York'],
                    columns=['one', 'two', 'three', 'four'])
data
```

Out[102]:

	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

In [103…
```
data.drop(['Colorado', 'Ohio'])
```

Out[103]:

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

In [104…
```
data.drop('two', axis=1)
```

Out[104]:

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

In [105…
```
data.drop(['two', 'four'], axis='columns')
```

Out[105]:

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

In [106…
```
obj.drop('c', inplace=True)
obj
```

Out[106]:
```
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

Indexing, Selection, and Filtering

In [107…
```
obj = pd.Series(np.arange(4.), index=['a', 'b', 'c', 'd'])
obj
```

Out[107]:
```
a    0.0
b    1.0
c    2.0
d    3.0
dtype: float64
```

In [108…
```
obj['b']
```

Out[108]:
```
1.0
```

In [109…
```
obj[1]
```

Out[109]:
```
1.0
```

In [110…
```
obj[2:4]
```

Out[110]:
```
c    2.0
d    3.0
dtype: float64
```

In [111…
```
obj[['b', 'a', 'd']]
```

Out[111]:
```
b    1.0
a    0.0
d    3.0
dtype: float64
```

In [112…
```
obj[[1, 3]]
```

Out[112]:
```
b    1.0
d    3.0
dtype: float64
```

In [113…
```
obj[obj < 2]
```

Out[113]:
```
a    0.0
b    1.0
dtype: float64
```

In [114…
```
obj['b':'c']
```

Out[114]:
```
b    1.0
c    2.0
dtype: float64
```

In [115…
```
obj['b':'c'] = 5
obj
```

Out[115]:
```
a    0.0
b    5.0
c    5.0
d    3.0
dtype: float64
```

In [116…
```
data = pd.DataFrame(np.arange(16).reshape((4, 4)),
                    index=['Ohio', 'Colorado', 'Utah', 'New York'],
                    columns=['one', 'two', 'three', 'four'])
data
```

Out[116]:

	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

In [117…
```
data['two']
```

Out[117]:
```
Ohio         1
Colorado     5
Utah         9
New York    13
Name: two, dtype: int32
```

In [118…
```
data[['three', 'one']]
```

Out[118]:

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

In [119…

```
data[:2]
```

Out[119]:

	one	two	three	four
Ohio	0	1	2	3
Colorado	4	5	6	7

In [120…

```
data[data['three'] > 5]
```

Out[120]:

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

In [121…

```
data < 5
```

Out[121]:

	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

In [122…

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

Out[122]:

	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

Selection with loc and iloc

In [123…

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

Out[123]:

```
two      5
three    6
Name: Colorado, dtype: int32
```

In [124…
```
data.iloc[2, [3, 0, 1]]
```

Out[124]:
```
four     11
one       8
two       9
Name: Utah, dtype: int32
```

In [125…
```
data.iloc[2]
```

Out[125]:
```
one       8
two       9
three    10
four     11
Name: Utah, dtype: int32
```

In [126…
```
data.iloc[[1, 2], [3, 0, 1]]
```

Out[126]:

	four	one	two
Colorado	7	0	5
Utah	11	8	9

In [127…
```
data.loc[:'Utah', 'two']
```

Out[127]:
```
Ohio        0
Colorado    5
Utah        9
Name: two, dtype: int32
```

In [128…
```
data.iloc[:, :3][data.three > 5]
```

Out[128]:

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

Integer Indexes

In [129…
```
ser = pd.Series(np.arange(3.))
ser
```

Out[129]:
```
0    0.0
1    1.0
2    2.0
dtype: float64
```

In [130…
```
ser[-1]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\PythonDSA\anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(s
elf, key, method, tolerance)
    384                 try:
--> 385                     return self._range.index(new_key)
    386                 except ValueError as err:

ValueError: -1 is not in range

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16832\3387420178.py in <module>
----> 1 ser[-1]

C:\PythonDSA\anaconda3\lib\site-packages\pandas\core\series.py in __getitem__(sel
f, key)
    956 
    957         elif key_is_scalar:
--> 958             return self._get_value(key)
    959 
    960         if is_hashable(key):

C:\PythonDSA\anaconda3\lib\site-packages\pandas\core\series.py in _get_value(self,
label, takeable)
   1067 
   1068         # Similar to Index.get_value, but we do not fall back to positiona
l
-> 1069         loc = self.index.get_loc(label)
   1070         return self.index._get_values_for_loc(self, loc, label)
   1071 

C:\PythonDSA\anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(s
elf, key, method, tolerance)
    385                     return self._range.index(new_key)
    386                 except ValueError as err:
--> 387                     raise KeyError(key) from err
    388             self._check_indexing_error(key)
    389             raise KeyError(key)

KeyError: -1
```

In [131…

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

In [132…

```
ser
```

Out[132]:

```
0    0.0
1    1.0
2    2.0
dtype: float64
```

In [135…

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

Out[135]:

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

In [136…

```
ser2[-1]
```

Out[136]:

```
2.0
```

In [137…
```
ser[:1]
```
Out[137]:
```
0    0.0
dtype: float64
```

In [138…
```
ser.loc[:1]
```
Out[138]:
```
0    0.0
1    1.0
dtype: float64
```

In [139…
```
ser.iloc[:1]
```
Out[139]:
```
0    0.0
dtype: float64
```

Arithmetic and Data Alignment

In [140…
```
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
```
Out[140]:
```
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
```

In [141…
```
s2
```
Out[141]:
```
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
dtype: float64
```

In [142…
```
s1 + s2
```
Out[142]:
```
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64
```

In [143…
```
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
```
Out[143]:

	b	c	d
Ohio	0.0	1.0	2.0
Texas	3.0	4.0	5.0
Colorado	6.0	7.0	8.0

In [144…
```
df2
```

Out[144]:

	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

In [145…

```
df1 + df2
```

Out[145]:

	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

In [146…

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

Out[146]:

	A
0	1
1	2

In [147…

```
df2
```

Out[147]:

	B
0	3
1	4

In [148…

```
df1 - df2
```

Out[148]:

	A	B
0	NaN	NaN
1	NaN	NaN

Arithmetic methods with fill values

In [149…

```
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
```

Out[149]:

	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

In [150…

```
df2
```

Out[150]:

	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

In [151…

```
df1 + df2
```

Out[151]:

	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

In [152…

```
df1.add(df2, fill_value=0)
```

Out[152]:

	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

In [153…

```
1 / df1
```

Out[153]:

	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

In [154…

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

Out[154]:

	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

In [155…

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

Out[155]:

	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

Operations between DataFrame and Series

In [156…

```
arr = np.arange(12.).reshape((3, 4))
arr
```

Out[156]:

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

In [157…

```
arr[0]
```

Out[157]:

```
array([0., 1., 2., 3.])
```

In [158…

```
arr - arr[0]
```

Out[158]:

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

In [160…

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

Out[160]:

	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

In [161…

```
series
```

Out[161]:

```
b    0.0
d    1.0
e    2.0
Name: Utah, dtype: float64
```

In [162…

```
frame - series
```

Out[162]:

	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

In [164…

```
series2 = pd.Series(range(3), index=['b', 'e', 'f'])
series2
```

Out[164]:

```
b    0
e    1
f    2
dtype: int64
```

In [165…

```
frame + series2
```

Out[165]:

	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

In [166…

```
series3 = frame['d']
frame
```

Out[166]:

	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

In [167…

```
series3
```

Out[167]:

```
Utah       1.0
Ohio       4.0
Texas      7.0
Oregon    10.0
Name: d, dtype: float64
```

In [168…

```
frame.sub(series3, axis='index')
```

Out[168]:

	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

Function Application and Mapping

In [169…
```
frame = pd.DataFrame(np.random.randn(4, 3), columns=list('bde'),
                     index=['Utah', 'Ohio', 'Texas', 'Oregon'])
frame
```

Out[169]:

	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

In [170…
```
np.abs(frame)
```

Out[170]:

	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

In [171…
```
f = lambda x: x.max() - x.min()
frame.apply(f)
```

Out[171]:
```
b    1.802165
d    1.684034
e    2.689627
dtype: float64
```

In [172…
```
frame.apply(f, axis='columns')
```

Out[172]:
```
Utah      0.998382
Ohio      2.521511
Texas     0.676115
Oregon    2.542656
dtype: float64
```

In [173…
```
def f(x):
    return pd.Series([x.min(), x.max()], index=['min', 'max'])
frame.apply(f)
```

Out[173]:

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

In [174…
```
format = lambda x: '%.2f' % x
frame.applymap(format)
```

Out[174]:

	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

In [175…
```
frame['e'].map(format)
```

Out[175]:
```
Utah      -0.52
Ohio       1.39
Texas      0.77
Oregon    -1.30
Name: e, dtype: object
```

Sorting and Ranking

In [176…
```
obj = pd.Series(range(4), index=['d', 'a', 'b', 'c'])
obj.sort_index()
```

Out[176]:
```
a    1
b    2
c    3
d    0
dtype: int64
```

In [177…
```
frame = pd.DataFrame(np.arange(8).reshape((2, 4)),
                     index=['three', 'one'],
                     columns=['d', 'a', 'b', 'c'])
frame
```

Out[177]:

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

In [178…
```
frame.sort_index()
```

Out[178]:

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

In [179…
```
frame.sort_index(axis=1)
```

Out[179]:

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

In [180…
```
frame.sort_index(axis=1, ascending=False)
```

Out[180]:

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

In [181…

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

Out[181]:

```
2   -3
3    2
0    4
1    7
dtype: int64
```

In [182…

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

Out[182]:

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

In [183…

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

Out[183]:

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

In [184…

```
frame.sort_values(by='b')
```

Out[184]:

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

In [185…

```
frame.sort_values(by=['a', 'b'])
```

Out[185]:

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

In [186…

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

Out[186]:

```
0    6.5
1    1.0
2    6.5
3    4.5
4    3.0
5    2.0
6    4.5
dtype: float64
```

In [187…

```python
obj.rank(method='first')
```

Out[187]:

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

In [188…

```python
# Assign tie values the maximum rank in the group
obj.rank(ascending=False, method='max')
```

Out[188]:

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

In [189…

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

Out[189]:

	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

In [190…

```python
frame.rank(axis='columns')
```

Out[190]:

	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

Axis Indexes with Duplicate Labels

In [191…
```
obj = pd.Series(range(5), index=['a', 'a', 'b', 'b', 'c'])
obj
```

Out[191]:
```
a    0
a    1
b    2
b    3
c    4
dtype: int64
```

In [192…
```
obj.index.is_unique
```

Out[192]:
```
False
```

In [193…
```
obj['a']
```

Out[193]:
```
a    0
a    1
dtype: int64
```

In [194…
```
obj['c']
```

Out[194]:
```
4
```

In [195…
```
df = pd.DataFrame(np.random.randn(4, 3), index=['a', 'a', 'b', 'b'])
df
```

Out[195]:

	0	1	2
a	0.274992	0.228913	1.352917
a	0.886429	-2.001637	-0.371843
b	1.669025	-0.438570	-0.539741
b	0.476985	3.248944	-1.021228

In [196…
```
df.loc['b']
```

Out[196]:

	0	1	2
b	1.669025	-0.438570	-0.539741
b	0.476985	3.248944	-1.021228

Summarizing and Computing Descriptive Statistics

In [197…
```
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
```

Out[197]:

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

In [198…
```
df.sum()
```
Out[198]:
```
one     9.25
two    -5.80
dtype: float64
```
In [199…
```
df.sum(axis='columns')
```
Out[199]:
```
a    1.40
b    2.60
c    0.00
d   -0.55
dtype: float64
```
In [200…
```
df.mean(axis='columns', skipna=False)
```
Out[200]:
```
a      NaN
b    1.300
c      NaN
d   -0.275
dtype: float64
```
In [201…
```
df.mean(axis='columns')
```
Out[201]:
```
a    1.400
b    1.300
c      NaN
d   -0.275
dtype: float64
```
In [202…
```
df.idxmax()
```
Out[202]:
```
one    b
two    d
dtype: object
```
In [203…
```
df.cumsum()
```
Out[203]:

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

In [204…
```
df.describe()
```

Out[204]:

	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

In [206…

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

Out[206]:

```
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
```

In [207…

```
obj.describe()
```

Out[207]:

```
count     16
unique     3
top        a
freq       8
dtype: object
```

Correlation and Covariance

conda install pandas-datareader

In [224…

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

In [225…

```
import pandas_datareader.data as web
all_data = {ticker: web.get_data_yahoo(ticker)
            for ticker in ['AAPL', 'IBM', 'MSFT', 'GOOG']}

price = pd.DataFrame({ticker: data['Adj Close']
                     for ticker, data in all_data.items()})
volume = pd.DataFrame({ticker: data['Volume']
                      for ticker, data in all_data.items()})
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16832\1974973852.py in <module>
      1 import pandas_datareader.data as web
----> 2 all_data = {ticker: web.get_data_yahoo(ticker)
      3             for ticker in ['AAPL', 'IBM', 'MSFT', 'GOOG']}
      4 
      5 price = pd.DataFrame({ticker: data['Adj Close']

~\AppData\Local\Temp\ipykernel_16832\1974973852.py in <dictcomp>(.0)
      1 import pandas_datareader.data as web
----> 2 all_data = {ticker: web.get_data_yahoo(ticker)
      3             for ticker in ['AAPL', 'IBM', 'MSFT', 'GOOG']}
      4 
      5 price = pd.DataFrame({ticker: data['Adj Close']

C:\PythonDSA\anaconda3\lib\site-packages\pandas_datareader\data.py in get_data_yah
oo(*args, **kwargs)
     78 
     79 def get_data_yahoo(*args, **kwargs):
---> 80     return YahooDailyReader(*args, **kwargs).read()
     81 
     82 

C:\PythonDSA\anaconda3\lib\site-packages\pandas_datareader\base.py in read(self)
    251         # If a single symbol, (e.g., 'GOOG')
    252         if isinstance(self.symbols, (string_types, int)):
--> 253             df = self._read_one_data(self.url, params=self._get_params(sel
f.symbols))
    254         # Or multiple symbols, (e.g., ['GOOG', 'AAPL', 'MSFT'])
    255         elif isinstance(self.symbols, DataFrame):

C:\PythonDSA\anaconda3\lib\site-packages\pandas_datareader\yahoo\daily.py in _read
_one_data(self, url, params)
    150         ptrn = r"root\.App\.main = (.*?);\n}\(this\)\);"
    151         try:
--> 152             j = json.loads(re.search(ptrn, resp.text, re.DOTALL).group(1))
    153             data = j["context"]["dispatcher"]["stores"]["HistoricalPriceSt
ore"]
    154         except KeyError:

AttributeError: 'NoneType' object has no attribute 'group'
```

In [226…

```
import yfinance as yf

tickers = ['AAPL', 'IBM', 'MSFT', 'GOOG']
all_data = {ticker: yf.download(ticker) for ticker in tickers}

price = pd.DataFrame({ticker: data['Adj Close'] for ticker, data in all_data.items(
volume = pd.DataFrame({ticker: data['Volume'] for ticker, data in all_data.items()}
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
```

In [227…

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

Out[227]:

Date	AAPL	IBM	MSFT	GOOG
2024-08-05	-0.048167	-0.030721	-0.032657	-0.046081
2024-08-06	-0.009748	0.019039	0.011287	-0.000623
2024-08-07	0.012498	0.000000	-0.002953	0.001308
2024-08-08	0.016633	0.031103	0.010692	0.019222
2024-08-09	0.013736	0.002671	0.008269	0.009460

In [228…

```
returns['MSFT'].corr(returns['IBM'])
```

Out[228]:

```
0.4402627500353811
```

In [229…

```
returns['MSFT'].cov(returns['IBM'])
```

Out[229]:

```
0.00015847968195175535
```

In [230…

```
returns.MSFT.corr(returns.IBM)
```

Out[230]:

```
0.4402627500353811
```

In [231…

```
returns.corr()
```

Out[231]:

	AAPL	IBM	MSFT	GOOG
AAPL	1.000000	0.366891	0.430426	0.515725
IBM	0.366891	1.000000	0.440263	0.387398
MSFT	0.430426	0.440263	1.000000	0.563487
GOOG	0.515725	0.387398	0.563487	1.000000

In [232…

```
returns.cov()
```

Out[232]:

	AAPL	IBM	MSFT	GOOG
AAPL	0.000778	0.000170	0.000246	0.000205
IBM	0.000170	0.000251	0.000158	0.000107
MSFT	0.000246	0.000158	0.000445	0.000186
GOOG	0.000205	0.000107	0.000186	0.000373

In [233…

```
returns.corrwith(returns.IBM)
```

Out[233]:

```
AAPL    0.366891
IBM     1.000000
MSFT    0.440263
GOOG    0.387398
dtype: float64
```

In [234…

```
returns.corrwith(volume)
```

```
Out[234]: AAPL    0.000634
          IBM    -0.010061
          MSFT   -0.005571
          GOOG    0.036116
          dtype: float64
```

Unique Values, Value Counts, and Membership

```
In [235…  obj = pd.Series(['c', 'a', 'd', 'a', 'a', 'b', 'b', 'c', 'c'])
```

```
In [236…  uniques = obj.unique()
          uniques
```

```
Out[236]: array(['c', 'a', 'd', 'b'], dtype=object)
```

```
In [237…  obj.value_counts()
```

```
Out[237]: c    3
          a    3
          b    2
          d    1
          dtype: int64
```

```
In [238…  pd.value_counts(obj.values, sort=False)
```

```
Out[238]: c    3
          a    3
          d    1
          b    2
          dtype: int64
```

```
In [239…  obj
```

```
Out[239]: 0    c
          1    a
          2    d
          3    a
          4    a
          5    b
          6    b
          7    c
          8    c
          dtype: object
```

```
In [240…  mask = obj.isin(['b', 'c'])
          mask
```

```
Out[240]: 0     True
          1    False
          2    False
          3    False
          4    False
          5     True
          6     True
          7     True
          8     True
          dtype: bool
```

```
In [241…  obj[mask]
```

Out[241]:
```
0    c
5    b
6    b
7    c
8    c
dtype: object
```

In [242…
```
to_match = pd.Series(['c', 'a', 'b', 'b', 'c', 'a'])
unique_vals = pd.Series(['c', 'b', 'a'])
pd.Index(unique_vals).get_indexer(to_match)
```

Out[242]:
```
array([0, 2, 1, 1, 0, 2], dtype=int64)
```

In [243…
```
data = pd.DataFrame({'Qu1': [1, 3, 4, 3, 4],
                     'Qu2': [2, 3, 1, 2, 3],
                     'Qu3': [1, 5, 2, 4, 4]})
data
```

Out[243]:

	Qu1	Qu2	Qu3
0	1	2	1
1	3	3	5
2	4	1	2
3	3	2	4
4	4	3	4

In [244…
```
result = data.apply(pd.value_counts).fillna(0)
result
```

Out[244]:

	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

Conclusion

In [246…
```
pd.options.display.max_rows = PREVIOUS_MAX_ROWS
```