Practical : 4

Aim : Explore Pandas Data Structures.

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

data=pd.Series([4,5,6,7])
print(data)
```

```
0    4
1    5
2    6
3    7
dtype: int64
```

In [2]:
```
data1=pd.Series([7,4,56,6],index=['a','b','g','f'])
print(data1)
```

```
a     7
b     4
g    56
f     6
dtype: int64
```

In [3]:
```
(data1.index)
```

Out[3]:
```
Index(['a', 'b', 'g', 'f'], dtype='object')
```

In [4]:
```
data1[['a','b','g']]
```

Out[4]:
```
a     7
b     4
g    56
dtype: int64
```

In [5]:
```
data1[data1>0]
```

Out[5]:
```
a     7
b     4
g    56
f     6
dtype: int64
```

In [6]:
```
np.exp(data1)
```

Out[6]:
```
a    1.096633e+03
b    5.459815e+01
g    2.091659e+24
f    4.034288e+02
dtype: float64
```

In [7]:
```
'b' in data1
```

Out[7]:
```
True
```

In [8]:
```
dii1 = {'ohio':35000,'texas':71000,'oregon':16000,'utah':5000}
data2= pd.Series(dii1)
print(data2)
```

```
ohio      35000
texas     71000
oregon    16000
utah       5000
dtype: int64
```

In [9]:
```
state=['california','ohio','oregon','texas']
data3=pd.Series(dii1,index=state)
print(data3)
```

```
california        NaN
ohio          35000.0
oregon        16000.0
texas         71000.0
dtype: float64
```

In [10]:
```
pd.isnull(data3)
```

Out[10]:
```
california     True
ohio          False
oregon        False
texas         False
dtype: bool
```

In [11]:
```
pd.notnull(data3)
```

Out[11]:
```
california    False
ohio           True
oregon         True
texas          True
dtype: bool
```

In [12]:
```
data2
```

Out[12]:
```
ohio      35000
texas     71000
oregon    16000
utah       5000
dtype: int64
```

In [13]:
```
data3
```

Out[13]:
```
california        NaN
ohio          35000.0
oregon        16000.0
texas         71000.0
dtype: float64
```

In [14]:
```
data3+data2
```

Out[14]:
```
california         NaN
ohio           70000.0
oregon         32000.0
texas         142000.0
utah               NaN
dtype: float64
```

In [15]:
```
data3.name='population'
data3.index.name='state'
data3
```

Out[15]:
```
state
california        NaN
ohio          35000.0
oregon        16000.0
texas         71000.0
Name: population, dtype: float64
```

In [16]:
```
data
```

Out[16]:
```
0    4
1    5
2    6
3    7
dtype: int64
```

In [17]:
```
data.index=['bob','steve','jeff','ryan']
data
```

Out[17]:
```
bob      4
steve    5
jeff     6
ryan     7
dtype: int64
```

In [18]:
```
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]
}
```

In [19]:
```
df = pd.DataFrame(data)
```

In [20]:
```
df
```

Out[20]:

	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 [21]:
```
df.head()
```

Out[21]:

	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 [22]:
```
pd.DataFrame(data, columns = ['year', 'state', 'pop'])
```

Out[22]:

	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 [23]:
```
frame2 = pd.DataFrame(data, columns = ['year', 'state', 'pop', 'debt'],
                      index = ['one', 'two', 'three', 'four', 'five', 'six'])
```

In [24]:
```
frame2
```

Out[24]:

	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 [25]:
```
frame2['state']
```

Out[25]:
```
one        Ohio
two        Ohio
three      Ohio
four     Nevada
five     Nevada
six      Nevada
Name: state, dtype: object
```

In [26]:
```
frame2.year
```

Out[26]:
```
one      2000
two      2001
three    2002
four     2001
five     2002
six      2003
Name: year, dtype: int64
```

In [27]:
```
frame2.loc['three']
```

Out[27]:
```
year     2002
state    Ohio
pop       3.6
debt      NaN
Name: three, dtype: object
```

In [28]:
```
frame2['debt'] = 16.5
```

In [29]:
```
frame2
```

Out[29]:

	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 [30]:
```
frame2['debt'] = np.arange(6.)
```

In [31]:
```
frame2
```

Out[31]:

	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 [32]:
```
val = pd.Series([-1.2, -1.5, -1.7], index = ['two', 'four', 'five'])
```

In [33]:
```
frame2['debt'] = val
```

In [34]:
```
frame2
```

Out[34]:

	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 [35]:
```
frame2['eastern'] = frame2.state == 'Ohio'
```

In [36]:
```
frame2
```

Out[36]:

	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 [37]:
```
del frame2['eastern']
```

In [38]:
```
frame2.columns
```

Out[38]:
```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

In [39]:
```
pop = {
    'Nevada' : {
        2001 :2.4,
        2002 : 2.9
    },

    'Ohio' : {
        2000 : 1.5,
        2001 : 1.7,
        2002 : 3.6
    }
}
```

In [40]:
```
frame3 = pd.DataFrame(pop)
```

In [41]:
```
frame3
```

Out[41]:

	Nevada	Ohio
2001	2.4	1.7
2002	2.9	3.6
2000	NaN	1.5

In [42]:
```
frame3.T
```

Out[42]:

	2001	2002	2000
Nevada	2.4	2.9	NaN
Ohio	1.7	3.6	1.5

In [43]:
```
pdata = {
    'Ohio' : frame3['Ohio'][:-1],
```

```
    'Nevada' : frame3['Nevada'][:2]
}
```

In [44]:
```
pd.DataFrame(pdata)
```

Out[44]:

	Ohio	Nevada
2001	1.7	2.4
2002	3.6	2.9

In [45]:
```
frame3.index.name = 'year'; frame3.columns.name = 'state'
```

In [46]:
```
frame3
```

Out[46]:

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

In [47]:
```
frame3.values
```

Out[47]:
```
array([[2.4, 1.7],
       [2.9, 3.6],
       [nan, 1.5]])
```

In [48]:
```
frame2.values
```

Out[48]:
```
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 Object

In [49]:
```
obj = pd.Series(range(3), index = ['a', 'b', 'c'])
```

In [50]:
```
index = obj.index
```

In [51]:
```
index
```

Out[51]:
```
Index(['a', 'b', 'c'], dtype='object')
```

In [52]:
```
index[1:]
```

Out[52]:
```
Index(['b', 'c'], dtype='object')
```

In [53]:
```
labels = pd.Index(np.arange(3))
```

In [54]:
```
labels
```

Out[54]:
```
Index([0, 1, 2], dtype='int32')
```

In [55]:
```
obj2 = pd.Series([1.5, -2.5, 0], index = labels)
```

In [56]:
```
obj2
```

Out[56]:
```
0    1.5
1   -2.5
2    0.0
dtype: float64
```

In [57]:
```
obj2.index is labels
```

Out[57]:
```
True
```

In [58]:
```
frame3
```

Out[58]:

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

In [59]:
```
frame3.columns
```

Out[59]:
```
Index(['Nevada', 'Ohio'], dtype='object', name='state')
```

In [60]:
```
'Ohio' in frame3.columns
```

Out[60]:
```
True
```

In [61]:
```
2003 in frame3.index
```

Out[61]:
```
False
```

In [62]:
```
dup_labels = pd.Index(['foo', 'foo', 'bar', 'bar'])
```

In [63]:
```
dup_labels
```

Out[63]:
```
Index(['foo', 'foo', 'bar', 'bar'], dtype='object')
```

esential functionality

In [64]:
```
obj = pd.Series([4.5, 7.2, -5.3, 3.6], index = ['d', 'b', 'a', 'c'])
```

In [65]:
```
obj
```

Out[65]:
```
d    4.5
b    7.2
a   -5.3
c    3.6
dtype: float64
```

In [66]:
```
obj2 = obj.reindex(['a', 'b', 'c', 'd', 'e'])
```

In [67]:
```
obj2
```

Out[67]:
```
a   -5.3
b    7.2
c    3.6
d    4.5
e    NaN
dtype: float64
```

In [68]:
```
obj3 = pd.Series(['blue', 'purple', 'yellow'], index = [0, 2, 4])
```

In [69]:
```
obj3
```

Out[69]:
```
0      blue
2    purple
4    yellow
dtype: object
```

In [70]:
```
obj3.reindex(range(6), method = 'ffill')
```

Out[70]:
```
0      blue
1      blue
2    purple
3    purple
4    yellow
5    yellow
dtype: object
```

In [71]:
```
frame = pd.DataFrame(np.arange(9).reshape((3,3)),
                     index = ['a', 'c', 'd'],
                     columns = ['Ohio', 'Texas', 'California'])
```

In [72]:
```
frame
```

Out[72]:

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

In [73]:
```
frame2 = frame.reindex(['a', 'b', 'c', 'd'])
```

In [74]:
```
frame2
```

Out[74]:

	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 [75]:
```
states = ['Texas', 'Utah', 'California']
```

In [76]:
```
frame.reindex(columns = states)
```

Out[76]:

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

Droping entries from axis

In [77]:
```
obj = pd.Series(np.arange(5.), index = ['a', 'b', 'c', 'd', 'e'])
```

In [78]:
```
obj
```

Out[78]:
```
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
```

In [79]:
```
new_obj = obj.drop('c')
```

In [80]:
```
new_obj
```

Out[80]:
```
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

In [81]:
```
obj.drop(['d', 'c'])
```

Out[81]:
```
a    0.0
b    1.0
e    4.0
dtype: float64
```

In [82]:
```
data = pd.DataFrame(np.arange(16).reshape((4,4)),
                    index = ['Ohio', 'Colorado', 'Utah', 'New York'],
                    columns = ['one', 'two', 'three', 'four'])
```

In [83]:
```
data
```

Out[83]:

	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 [84]:
```
data.drop(['Colorado', 'Ohio'])
```

Out[84]:

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

In [85]:
```
data.drop('two', axis = 1)
```

Out[85]:

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

In [86]:
```
data.drop(['two', 'four'], axis = 'columns')
```

Out[86]:

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

In [87]:
```
obj.drop('c', inplace = True)
```

In [88]:
```
obj
```

Out[88]:
```
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

Indexing, Selection and Filtering

In [89]:
```
obj = pd.Series(np.arange(4.), index = ['a', 'b', 'c', 'd'])
```

In [90]:
```
obj
```

Out[90]:
```
a    0.0
b    1.0
c    2.0
d    3.0
dtype: float64
```

In [91]:
```
obj['b']
```

Out[91]:
```
1.0
```

In [92]:
```
obj[1]
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_12896\2469632899.py:1: FutureWar
ning: Series.__getitem__ treating keys as positions is deprecated. In a futu
re version, integer keys will always be treated as labels (consistent with D
ataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  obj[1]
```

Out[92]:
```
1.0
```

In [93]:
```
obj[2:4]
```

Out[93]:
```
c    2.0
d    3.0
dtype: float64
```

In [94]:
```
obj[['b', 'a', 'd']]
```

Out[94]:
```
b    1.0
a    0.0
d    3.0
dtype: float64
```

In [95]:
```
obj[[1, 3]]
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_12896\2982346117.py:1: FutureWar
ning: Series.__getitem__ treating keys as positions is deprecated. In a futu
re version, integer keys will always be treated as labels (consistent with D
ataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  obj[[1, 3]]
```

Out[95]:
```
b    1.0
d    3.0
dtype: float64
```

In [96]:
```
obj[obj < 2]
```

Out[96]:
```
a    0.0
b    1.0
dtype: float64
```

In [97]:
```
obj['b': 'c']
```

Out[97]:
```
b    1.0
c    2.0
dtype: float64
```

In [98]:
```
obj['b': 'c'] = 5
```

In [99]:
```
obj
```

Out[99]:
```
a    0.0
b    5.0
c    5.0
d    3.0
dtype: float64
```

In [100…
```
data = pd.DataFrame(np.arange(16).reshape((4,4)),
                    index = ['Ohio', 'Colorado', 'Utah', 'New York'],
                    columns = ['one', 'two', 'three', 'four'])
```

In [101…
```
data
```

Out[101…

	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 [102…
```
data['two']
```

Out[102…
```
Ohio         1
Colorado     5
Utah         9
New York    13
Name: two, dtype: int32
```

In [103…
```
data[['three', 'one']]
```

Out[103…

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

In [104…
```
data[:2]
```

Out[104…

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

In [105…
```
data[data['three'] > 5]
```

Out[105…

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

In [106…
```
data < 5
```

Out[106…

	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 [107…
```
data[data < 5] = 0
```

In [108…
```
data
```

Out[108…

	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 [109…
```
data.loc['Colorado', ['two', 'three']]
```

Out[109…
```
two      5
three    6
Name: Colorado, dtype: int32
```

In [110…
```
data.iloc[2,[3,0,1]]
```

Out[110…
```
four    11
one      8
two      9
Name: Utah, dtype: int32
```

In [111…
```
data.iloc[2]
```

Out[111…
```
one       8
two       9
three    10
four     11
Name: Utah, dtype: int32
```

In [112…
```
data.iloc[[1,2], [3,0,1]]
```

Out[112…

	four	one	two
Colorado	7	0	5
Utah	11	8	9

In [113…
```
data.iloc[:, :3][data.three > 5]
```

Out[113…

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

Integer Index

In [114…
```
ser = pd.Series(np.arange(3.))
```

In [115…
```
ser
```

Out[115…
```
0    0.0
1    1.0
2    2.0
dtype: float64
```

In [116…
```
ser2 = pd.Series(np.arange(3.), index = ['a', 'b', 'c'])
```

In [117…
```
ser2[-1]
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_12896\811950851.py:1: FutureWarn
ing: Series.__getitem__ treating keys as positions is deprecated. In a futur
e version, integer keys will always be treated as labels (consistent with Da
taFrame behavior). To access a value by position, use `ser.iloc[pos]`
  ser2[-1]
```

Out[117… 2.0

In [118…
```
ser[:1]
```

Out[118…
```
0    0.0
dtype: float64
```

In [119…
```
ser.loc[:1]
```

Out[119…
```
0    0.0
1    1.0
dtype: float64
```

In [120…
```
ser.iloc[:1]
```

Out[120…
```
0    0.0
dtype: float64
```

In [121…
```
s1 = pd.Series([7.3, -2.5, 3.4, 1.5], index = ['a', 'c', 'd', 'e'])
```

In [122…
```
s2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1], index = ['a', 'c', 'e', 'f', 'g'])
```

In [123…
```
s1
```

Out[123…
```
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
```

In [124…
```
s2
```

Out[124…
```
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
dtype: float64
```

In [125…
```
s1 + s2
```

Out[125…
```
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64
```

In [126…
```
df1 = pd.DataFrame(np.arange(9.).reshape((3,3)), columns = list('bcd'),
                   index = ['Ohio', 'Texas', 'Colorado'])
```

In [127…
```
df2 = pd.DataFrame(np.arange(12.).reshape((4,3)), columns = list('bde'),
                   index = ['Utah', 'Ohio', 'Texas', 'Oregon'])
```

In [128…
```
df1
```

Out[128…

	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 [129…
```
df2
```

Out[129…

	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 [130…
```
df1 + df2
```

Out[130…

	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 [131…
```
df1 = pd.DataFrame({'A' : [1,2]})
```

In [132…
```
df2 = pd.DataFrame({'B' : [3,4]})
```

In [133…
```
df1
```

Out[133…

	A
0	1
1	2

In [ ]:

Out[134…

	B
0	3
1	4

In [135…
```
df1 - df2
```

Out[135…

	A	B
0	NaN	NaN
1	NaN	NaN

Arithmatic methods with fill values

In [136…
```
df1 = pd.DataFrame(np.arange(12.).reshape((3,4)), columns = list('abcd'))
```

In [137…
```
df2 = pd.DataFrame(np.arange(20.).reshape((4,5)), columns = list('abcde'))
```

In [138…
```
df1
```

Out[138…

	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 [139…
```
df2
```

Out[139…

	a	b	c	d	e
0	0.0	1.0	2.0	3.0	4.0
1	5.0	6.0	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 [140…
```
df2.loc[1, 'b'] = np.nan
```

In [141…
```
df2
```

Out[141…

	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 [142…
```
df1 + df2
```

Out[142…

	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 [143…
```
df1.add(df2, fill_value = 0)
```

Out[143…

	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 [144…
```
1/df1
```

Out[144…

	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 [145…
```
df1.rdiv(1)
```

Out[145…

	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 [146…
```
df1.reindex(columns = df2.columns, fill_value = 0)
```

Out[146…

	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

In [147…
```
arr = np.arange(12.).reshape((3,4))
```

In [148…
```
arr[0]
```

Out[148…
```
array([0., 1., 2., 3.])
```

In [149…
```
arr - arr[0]
```

Out[149…
```
array([[0., 0., 0., 0.],
       [4., 4., 4., 4.],
       [8., 8., 8., 8.]])
```

In [150…
```
frame = pd.DataFrame(np.arange(12.).reshape((4,3)), columns = list('bde'),
                     index = ['Utah', 'Ohio', 'Texas', 'Oregon'])
```

In [151…
```
series = frame.iloc[0]
```

In [152…
```
frame
```

Out[152…

	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 [153…
```
series
```

Out[153…
```
b    0.0
d    1.0
e    2.0
Name: Utah, dtype: float64
```

In [154…
```
frame - series
```

Out[154…

	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 [155…
```
series2 = pd.Series(range(3), index = ['b', 'e', 'f'])
```

In [156…
```
frame +series2
```

Out[156…

	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 [157…
```
series3 = frame['d']
```

In [158…
```
frame
```

Out[158…

	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 [159…
```
series3
```

Out[159…
```
Utah       1.0
Ohio       4.0
Texas      7.0
Oregon    10.0
Name: d, dtype: float64
```

In [160…
```
frame.sub(series3, axis = 'index')
```

Out[160…

	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

Functio Application and Mapping

In [161…
```
frame = pd.DataFrame(np.random.randn(4,3), columns = list('bde'),
                     index = ['Utah', 'Ohio', 'Texas', 'Oregon'])
```

In [162…
```
frame
```

Out[162…

	b	d	e
Utah	0.886382	-1.302339	0.666226
Ohio	-1.682805	-1.386821	-0.720509
Texas	-0.129567	-0.278961	0.525297
Oregon	1.207296	-0.991846	-0.649313

In [163…
```
np.abs(frame)
```

Out[163…

	b	d	e
Utah	0.886382	1.302339	0.666226
Ohio	1.682805	1.386821	0.720509
Texas	0.129567	0.278961	0.525297
Oregon	1.207296	0.991846	0.649313

In [164…
```
f = lambda x: x.max() - x.min()
```

In [165…
```
frame.apply(f)
```

Out[165…
```
b    2.890101
d    1.107860
e    1.386734
dtype: float64
```

In [166…
```
def f(x) :
  return pd.Series([x.min(), x.max()], index = ['min', 'max'])
```

In [167…
```
frame.apply(f)
```

Out[167…

	b	d	e
min	-1.682805	-1.386821	-0.720509
max	1.207296	-0.278961	0.666226

In [168…
```
format = lambda x: '%.2f' % x
```

In [169…
```
frame.applymap(format)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_12896\304758519.py:1: FutureWarn
ing: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  frame.applymap(format)
```

Out[169…

	b	d	e
Utah	0.89	-1.30	0.67
Ohio	-1.68	-1.39	-0.72
Texas	-0.13	-0.28	0.53
Oregon	1.21	-0.99	-0.65

In [170…
```
frame['e'].map(format)
```

Out[170…
```
Utah       0.67
Ohio      -0.72
Texas      0.53
Oregon    -0.65
Name: e, dtype: object
```

Sorting and Ranking

In [171…
```
obj = pd.Series(range(4), index = ['d', 'a', 'b', 'c'])
```

In [172…
```
obj.sort_index()
```

Out[172…
```
a    1
b    2
c    3
d    0
dtype: int64
```

In [173…
```
frame = pd.DataFrame(np.arange(8).reshape((2,4)),
                     index = ['three', 'one'],
                     columns = ['d', 'a', 'b', 'c'])
```

In [174…
```
frame.sort_index()
```

Out[174…

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

In [175…
```
frame.sort_index(axis = 1)
```

Out[175…

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

In [176…
```
frame.sort_index(axis = 1, ascending=False)
```

Out[176…

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

In [177…
```
obj = pd.Series([4, 7, -3, 2])
```

In [178…
```
obj.sort_values()
```

Out[178…
```
2   -3
3    2
0    4
1    7
dtype: int64
```

In [179…
```
obj = pd.Series([4, np.nan, 7, np.nan, -3, 2])
```

In [180…
```
obj.sort_values()
```

Out[180…
```
4   -3.0
5    2.0
0    4.0
2    7.0
1    NaN
3    NaN
dtype: float64
```

In [181…
```
frame = pd.DataFrame({
    'b' : [4, 7, -3, 2],
    'a' : [0, 1, 0, 1]
})
```

In [182…
```
frame
```

Out[182…

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

In [183…
```
frame.sort_values(by = 'b')
```

Out[183…

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

In [184…
```
frame.sort_values(by = ['a', 'b'])
```

Out[184…

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

In [185…
```
obj = pd.Series([7, -5, 7, 4, 2, 0, 4])
```

In [186…
```
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…
```
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…
```
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…
```
frame = pd.DataFrame({
    'b' : [4.3, 7, -3, 2],
    'a' : [0, 1, 0, 1],
    'c' : [-2, 5, 8, -2.5]
})
```

In [190…
```
frame
```

Out[190…

	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

Axis Index with Duplicate Labels

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

In [192…
```
obj
```

Out[192…
```
a    0
a    1
b    2
b    3
c    4
dtype: int64
```

In [193…
```
obj.index.is_unique
```

Out[193…
```
False
```

In [194…
```
obj['a']
```

Out[194…
```
a    0
a    1
dtype: int64
```

In [195…
```
obj['c']
```

Out[195…
```
4
```

In [196…
```
df = pd.DataFrame(np.random.randn(4,3), index = ['a', 'a', 'b', 'b'])
```

In [197…
```
df
```

Out[197…

	0	1	2
a	-0.325623	0.076378	0.527401
a	-1.816576	-0.113608	-0.767549
b	0.795250	-0.153524	0.077303
b	0.421872	1.173547	-1.300046

In [198…
```
df.loc['b']
```

Out[198…

	0	1	2
b	0.795250	-0.153524	0.077303
b	0.421872	1.173547	-1.300046

Summarizing and Computing Descriptive Statistic

In [199…
```
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'])
```

In [200…
```
df
```

Out[200…

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

In [201…
```
df.sum()
```

Out[201…
```
one    9.25
two   -5.80
dtype: float64
```

In [202…
```
df.sum(axis = 'columns')
```

Out[202…
```
a    1.40
b    2.60
c    0.00
d   -0.55
dtype: float64
```

In [203…
```
df.mean(axis = 'columns', skipna = False)
```

Out[203…
```
a      NaN
b    1.300
c      NaN
d   -0.275
dtype: float64
```

In [204…
```
df.idxmax()
```

Out[204…
```
one    b
two    d
dtype: object
```

In [205…
```
df.cumsum()
```

Out[205…

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

In [206…
```
df.describe()
```

Out[206…

	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 [207…
```
obj = pd.Series(['a', 'a', 'b', 'c'] * 4)
obj.describe()
```

Out[207…
```
count     16
unique     3
top        a
freq       8
dtype: object
```