Q1. Give the output of the following commands

In [ ]:
```
import pandas as pd

from pandas import Series, DataFrame

import numpy as np
np.random.seed(12345)
```

i)

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

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

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

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

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

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

ii)

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

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

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

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

In [ ]:
```
obj2['a']
obj2['d'] = 6
obj2[['c', 'a', 'd']]
obj2
```

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

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

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

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

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

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

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

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

Out[ ]:
```
True
```

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

Out[ ]:
```
False
```

iii)

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

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

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

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

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

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

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

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

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

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

In [ ]:
```
obj3
```

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

In [ ]:
```
obj4
```

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

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

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

In [ ]:
```
obj4.name = 'population'
```

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

In [ ]:
```
obj4
```

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

In [ ]:
```
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
```

Out[ ]:

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

Out[ ]:

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

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

Out[ ]:

	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 [ ]:
```
frame2.columns
```

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

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

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

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

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

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

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

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

Out[ ]:

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

Out[ ]:

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

Out[ ]:

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

Out[ ]:

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

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

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

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

Out[ ]:

	Nevada	Ohio
2001	2.4	1.7
2002	2.9	3.6
2000	NaN	1.5

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

Out[ ]:

	2001	2002	2000
Nevada	2.4	2.9	NaN
Ohio	1.7	3.6	1.5

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

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

frame3.index.name = 'year'; frame3.columns.name = 'state'
frame3
```

Out[ ]:

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

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

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

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

Out[ ]:
```
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)
```

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

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

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

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

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

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

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

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

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

Out[ ]:
```
True
```

In [ ]:
```
frame3
```

Out[ ]:

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

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

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

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

Out[ ]:
```
True
```

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

Out[ ]:
```
False
```

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

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

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

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

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

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

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

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

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

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

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

Out[ ]:

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

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

Out[ ]:

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

Out[ ]:

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

In [ ]:
```
frame.loc[['a', 'b', 'c', 'd'], states]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
c:\Users\hp\Desktop\SEM III\DSE Data Analysis and Visualisation\Hands on\Assignme
nt1.ipynb Cell 63 line 1
----> <a href='vscode-notebook-cell:/c%3A/Users/hp/Desktop/SEM%20III/DSE%20Data%2
0Analysis%20and%20Visualisation/Hands%20on/Assignment1.ipynb#Y116sZmlsZQ%3D%3D?li
ne=0'>1</a> frame.loc[['a', 'b', 'c', 'd'], states]

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexing.py:1097, in _LocationIndexer.__getitem__(self, key)
   1095     if self._is_scalar_access(key):
   1096         return self.obj._get_value(*key, takeable=self._takeable)
-> 1097     return self._getitem_tuple(key)
   1098 else:
   1099     # we by definition only have the 0th axis
   1100     axis = self.axis or 0

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexing.py:1287, in _LocIndexer._getitem_tuple(self, tup)
   1285 # ugly hack for GH #836
   1286 if self._multi_take_opportunity(tup):
-> 1287     return self._multi_take(tup)
   1289 return self._getitem_tuple_same_dim(tup)

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexing.py:1238, in _LocIndexer._multi_take(self, tup)
   1222 """
   1223 Create the indexers for the passed tuple of keys, and
   1224 executes the take operation. This allows the take operation to be
   (...)
   1235 values: same type as the object being indexed
   1236 """
   1237 # GH 836
-> 1238 d = {
   1239     axis: self._get_listlike_indexer(key, axis)
   1240     for (key, axis) in zip(tup, self.obj._AXIS_ORDERS)
   1241 }
   1242 return self.obj._reindex_with_indexers(d, copy=True, allow_dups=True)

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexing.py:1239, in <dictcomp>(.0)
   1222 """
   1223 Create the indexers for the passed tuple of keys, and
   1224 executes the take operation. This allows the take operation to be
   (...)
   1235 values: same type as the object being indexed
   1236 """
   1237 # GH 836
   1238 d = {
-> 1239     axis: self._get_listlike_indexer(key, axis)
   1240     for (key, axis) in zip(tup, self.obj._AXIS_ORDERS)
   1241 }
   1242 return self.obj._reindex_with_indexers(d, copy=True, allow_dups=True)

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexing.py:1462, in _LocIndexer._get_listlike_indexer(self, key, axis)
   1459 ax = self.obj._get_axis(axis)
   1460 axis_name = self.obj._get_axis_name(axis)
-> 1462 keyarr, indexer = ax._get_indexer_strict(key, axis_name)
   1464 return keyarr, indexer
```

```
File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexes\base.py:5877, in Index._get_indexer_strict(self, key, axis_name)
   5874 else:
   5875     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 5877 self._raise_if_missing(keyarr, indexer, axis_name)
   5879 keyarr = self.take(indexer)
   5880 if isinstance(key, Index):
   5881     # GH 42790 - Preserve name from an Index

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexes\base.py:5941, in Index._raise_if_missing(self, key, indexer, axis_n
ame)
   5938     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   5940 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 5941 raise KeyError(f"{not_found} not in index")

KeyError: "['b'] not in index"
```

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

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

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

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

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

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

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

Out[ ]:

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

Out[ ]:

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

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

Out[ ]:

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

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

Out[ ]:

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

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

Out[ ]:

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

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

In [ ]:
```
obj
```

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

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

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

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

Out[ ]:
```
1.0
```

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

Out[ ]:
```
1.0
```

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

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

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

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

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

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

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

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

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

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

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

In [ ]:
```
obj
```

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

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

Out[ ]:

	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 [ ]:
```
data['two']
```

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

In [ ]:
```
data[['three', 'one']]
```

Out[ ]:

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

In [ ]:
```
data[:2]
```

Out[ ]:

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

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

Out[ ]:

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

In [ ]:
```
data < 5
```

Out[ ]:

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

Out[ ]:

	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

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

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

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

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

In [ ]:
```
data.iloc[2]
```

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

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

Out[ ]:

	four	one	two
Colorado	7	0	5
Utah	11	8	9

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

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

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

Out[ ]:

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

In [ ]:
```
ser = pd.Series(np.arange(3))
ser
```

Out[ ]:
```
0    0
1    1
2    2
dtype: int32
```

In [ ]:
```
ser[-1]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexes\range.py:345, in RangeIndex.get_loc(self, key)
    344 try:
--> 345     return self._range.index(new_key)
    346 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)
c:\Users\hp\Desktop\SEM III\DSE Data Analysis and Visualisation\Hands on\Assignme
nt1.ipynb Cell 98 line 1
----> <a href='vscode-notebook-cell:/c%3A/Users/hp/Desktop/SEM%20III/DSE%20Data%2
0Analysis%20and%20Visualisation/Hands%20on/Assignment1.ipynb#Y166sZmlsZQ%3D%3D?li
ne=0'>1</a> ser[-1]

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\series.py:1007, in Series.__getitem__(self, key)
   1004     return self._values[key]
   1006 elif key_is_scalar:
-> 1007     return self._get_value(key)
   1009 if is_hashable(key):
   1010     # Otherwise index.get_value will raise InvalidIndexError
   1011     try:
   1012         # For labels that don't resolve as scalars like tuples and frozen
sets

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\series.py:1116, in Series._get_value(self, label, takeable)
   1113     return self._values[label]
   1115 # Similar to Index.get_value, but we do not fall back to positional
-> 1116 loc = self.index.get_loc(label)
   1118 if is_integer(loc):
   1119     return self._values[loc]

File c:\Users\hp\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas
\core\indexes\range.py:347, in RangeIndex.get_loc(self, key)
    345         return self._range.index(new_key)
    346     except ValueError as err:
--> 347         raise KeyError(key) from err
    348 if isinstance(key, Hashable):
    349     raise KeyError(key)

KeyError: -1
```

In [ ]:
```
ser = pd.Series(np.arange(3.))

ser
```

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

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

Out[ ]:
```
2.0
```

In [ ]:
```
ser[:1]
```

Out[ ]:
```
0    0.0
dtype: float64
```

In [ ]:
```
ser.loc[:1]
```

Out[ ]:
```
0    0.0
1    1.0
dtype: float64
```

In [ ]:
```
ser.iloc[:1]
```

Out[ ]:
```
0    0.0
dtype: float64
```

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

In [ ]:
```
s2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1],
               index=['a', 'c', 'e', 'f', 'g'])
s1
```

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

In [ ]:
```
s2
```

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

In [ ]:
```
s1 + s2
```

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

In [ ]:
```
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[ ]:

	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 [ ]:
```
df2
```

Out[ ]:

	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 [ ]:
```
df1 + df2
```

Out[ ]:

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

In [ ]:
```
df1
```

Out[ ]:

	A
0	1
1	2

In [ ]:
```
df2
```

Out[ ]:

	B
0	3
1	4

In [ ]:
```
df1 - df2
```

Out[ ]:

	A	B
0	NaN	NaN
1	NaN	NaN

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

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

In [ ]:
```
df2.loc[1, 'b'] = np.nan
```

In [ ]:
```
df1
```

Out[ ]:

	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 [ ]:
```
df2
```

Out[ ]:

	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 [ ]:
```
df1 + df2
```

Out[ ]:

	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 [ ]:
```
df1.add(df2, fill_value=0)
```

Out[ ]:

	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 [ ]:
```
1 / df1
```

Out[ ]:

	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 [ ]:
```
df1.rdiv(1)
```

Out[ ]:

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

Out[ ]:

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

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

In [ ]:
```
arr[0]
```

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

In [ ]:
```
arr - arr[0]
```

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

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

Out[ ]:

	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 [ ]:
```
series
```

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

In [ ]:
```
frame - series
```

Out[ ]:

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

Out[ ]:

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

Out[ ]:

	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 [ ]:
```
series3
```

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

In [ ]:
```
frame.sub(series3, axis='index')
```

Out[ ]:

	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

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

Out[ ]:

	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 [ ]:
```
np.abs(frame)
```

Out[ ]:

	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 [ ]:
```
f = lambda x: x.max() - x.min()
frame.apply(f)
```

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

In [ ]:
```
frame.apply(f, axis='columns')
```

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

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

Out[ ]:

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

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

Out[ ]:

	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 [ ]:
```
frame['e'].map(format)
```

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

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

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

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

Out[ ]:

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

In [ ]:
```
frame.sort_index(axis=1)
```

Out[ ]:

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

In [ ]:
```
frame.sort_index(axis=1, ascending=False)
```

Out[ ]:

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

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

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

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

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

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

Out[ ]:

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

In [ ]:
```
frame.sort_values(by='b')
```

Out[ ]:

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

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

Out[ ]:

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

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

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

In [ ]:
```
obj.rank(method='first')
```

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

In [ ]:
```
obj.rank(ascending=False, method='max')

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

Out[ ]:

	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 [ ]:
```
frame.rank(axis='columns')
```

Out[ ]:

	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

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

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

In [ ]:
```
obj.index.is_unique
```

Out[ ]:
```
False
```

In [ ]:
```
obj['a']
```

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

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

Out[ ]:
```
4
```

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

Out[ ]:

	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 [ ]:
```
df.loc['b']
```

Out[ ]:

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

In [ ]:
```
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[ ]:

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

In [ ]:
```
df.sum()
```

Out[ ]:
```
one    9.25
two   -5.80
dtype: float64
```

In [ ]:
```
df.sum(axis='columns')
```

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

In [ ]:
```
df.mean(axis='columns', skipna=False)
```

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

In [ ]:
```
df.idxmax()
```

Out[ ]:
```
one    b
two    d
dtype: object
```

In [ ]:
```
df.cumsum()
```

Out[ ]:

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

In [ ]:
```
df.describe()
```

Out[ ]:

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

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

In [ ]:
```
pip install pandas-datareader
```

```
Collecting pandas-datareader
  Downloading pandas_datareader-0.10.0-py3-none-any.whl (109 kB)
     ---------------------------------------- 0.0/109.5 kB ? eta -:--:--
     ----------- ---------------------------- 30.7/109.5 kB ? eta -:--:--
     ------------------- ---------------- 61.4/109.5 kB 812.7 kB/s eta 0:00:01
     ---------------------------------- 109.5/109.5 kB 793.8 kB/s eta 0:00:00
Requirement already satisfied: lxml in c:\users\hp\appdata\local\programs\python
\python310\lib\site-packages (from pandas-datareader) (4.9.2)
Requirement already satisfied: pandas>=0.23 in c:\users\hp\appdata\local\programs
\python\python310\lib\site-packages (from pandas-datareader) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in c:\users\hp\appdata\local\prog
rams\python\python310\lib\site-packages (from pandas-datareader) (2.31.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\appdata\loca
l\programs\python\python310\lib\site-packages (from pandas>=0.23->pandas-dataread
er) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\appdata\local\programs
\python\python310\lib\site-packages (from pandas>=0.23->pandas-datareader) (2022.
2.1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\hp\appdata\local\progra
ms\python\python310\lib\site-packages (from pandas>=0.23->pandas-datareader) (202
2.4)
Requirement already satisfied: numpy>=1.21.0 in c:\users\hp\appdata\local\program
s\python\python310\lib\site-packages (from pandas>=0.23->pandas-datareader) (1.2
3.5)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hp\appdata\lo
cal\programs\python\python310\lib\site-packages (from requests>=2.19.0->pandas-da
tareader) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\appdata\local\programs
\python\python310\lib\site-packages (from requests>=2.19.0->pandas-datareader)
(3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hp\appdata\local\pr
ograms\python\python310\lib\site-packages (from requests>=2.19.0->pandas-dataread
er) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\appdata\local\pr
ograms\python\python310\lib\site-packages (from requests>=2.19.0->pandas-dataread
er) (2022.9.24)
Requirement already satisfied: six>=1.5 in c:\users\hp\appdata\local\programs\pyt
hon\python310\lib\site-packages (from python-dateutil>=2.8.2->pandas>=0.23->panda
s-datareader) (1.16.0)
Installing collected packages: pandas-datareader
Successfully installed pandas-datareader-0.10.0
Note: you may need to restart the kernel to use updated packages.
```