In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
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[3]:

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

In [4]:

```
df.sum() # df.sum(axis=0)  # df.sum(axis='rows')
```

Out[4]:

```
one    9.25
two   -5.80
dtype: float64
```

In [10]:

```
df.sum(axis=1) #df.sum(axis='columns')
```

Out[10]:

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

In [13]:

```
df
```

Out[13]:

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

In [11]:

```
df.mean()  # NaN values are excluded
```

Out[11]:

```
one    3.083333
two   -2.900000
dtype: float64
```

In [12]:

```
df.mean(axis=1)# NaN values are excluded
```

Out[12]:

```
a    1.400
b    1.300
c      NaN
d   -0.275
dtype: float64
```

In [14]:

```
df.mean(axis=1, skipna=False)#
```

Out[14]:

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

In [15]:

```
df
```

Out[15]:

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

In [16]:

```
df.idxmin()
```

Out[16]:

```
one    d
two    b
dtype: object
```

In [18]:

```
df
```

Out[18]:

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

In [20]:

```
df.cumsum(axis=0) #
```

Out[20]:

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

In [21]:

```
df.cumsum(axis=0, skipna=False) #
```

Out[21]:

	one	two
a	1.4	NaN
b	8.5	NaN
c	NaN	NaN
d	NaN	NaN

In [22]:

```
df.cumsum(axis=1) #
```

Out[22]:

	one	two
a	1.40	NaN
b	7.10	2.60
c	NaN	NaN
d	0.75	-0.55

In [23]:

```
df.describe()
```

Out[23]:

	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 [24]:

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

Out[24]:

```
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 [25]:

```
obj.describe() #unique,freq
```

Out[25]:

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

In [ ]:

```

```

In [26]:

```
pip install pandas-datareader
```

```
Collecting pandas-datareaderNote: you may need to restart the kernel to u
se updated packages.


[notice] A new release of pip is available: 23.0.1 -> 23.1.2


  Downloading pandas_datareader-0.10.0-py3-none-any.whl (109 kB)
     ---------------------------------------- 109.5/109.5 kB 3.1 MB/s eta
0:00:00
Collecting lxml
  Downloading lxml-4.9.2-cp39-cp39-win_amd64.whl (3.9 MB)
     ---------------------------------------- 3.9/3.9 MB 13.8 MB/s eta 0:
00:00
Collecting requests>=2.19.0
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
     ---------------------------------------- 62.6/62.6 kB ? eta 0:00:00
Requirement already satisfied: pandas>=0.23 in c:\users\mona adlakha\appd
ata\local\programs\python\python39\lib\site-packages (from pandas-datarea
der) (1.4.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\mona ad
lakha\appdata\local\programs\python\python39\lib\site-packages (from pand
as>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\mona adlakha\app
data\local\programs\python\python39\lib\site-packages (from pandas>=0.23-
>pandas-datareader) (1.22.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mona adlakha\appd
ata\local\programs\python\python39\lib\site-packages (from pandas>=0.23->
pandas-datareader) (2021.3)
Collecting urllib3<3,>=1.21.1
  Downloading urllib3-2.0.2-py3-none-any.whl (123 kB)
     ---------------------------------------- 123.2/123.2 kB ? eta 0:00:0
0
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.1.0-cp39-cp39-win_amd64.whl (97 kB)
     ---------------------------------------- 97.1/97.1 kB 5.8 MB/s eta
0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.5/61.5 kB ? eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2023.5.7-py3-none-any.whl (156 kB)
     ---------------------------------------- 157.0/157.0 kB 9.2 MB/s eta
0:00:00
Requirement already satisfied: six>=1.5 in c:\users\mona adlakha\appdata
\local\programs\python\python39\lib\site-packages (from python-dateutil>=
2.8.1->pandas>=0.23->pandas-datareader) (1.16.0)
Installing collected packages: urllib3, lxml, idna, charset-normalizer, c
ertifi, requests, pandas-datareader
Successfully installed certifi-2023.5.7 charset-normalizer-3.1.0 idna-3.4
lxml-4.9.2 pandas-datareader-0.10.0 requests-2.31.0 urllib3-2.0.2

[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Pandas - used to organize and format complex data in table structures called
DataFrames.
```

Pandas-datareader - used to access public financial data from the Internet and import it into Python as a DataFrame.

In [27]:

```
import pandas_datareader.data as web
```

In [32]:

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

Out[32]:

```
" all_data = {ticker: web.get_data_yahoo(ticker)\nfor ticker in ['AAPL',
'IBM', 'MSFT', 'GOOG']} "
```

It's possible by the time you are reading this that Yahoo! Finance no longer exists since Yahoo! was acquired by Verizon in 2017.
Refer to the pandas-datareader documentation online for the latest functionality.

Unique Values, Value Counts, and Membership

In [53]:

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

Out[53]:

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

In [54]:

```
uniques = obj.unique()  # values returned are not in sorted order
uniques
```

Out[54]:

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

In [55]:

```
# these values can be sortedusing sort()
uniques.sort()
```

In [56]:

```
uniques
```

Out[56]:

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

value_counts() computes a Series containing value frequencies

In [ ]:

```

```

In [39]:

```
obj.value_counts()
```

Out[39]:

```
c    3
a    3
b    2
d    1
dtype: int64
```

```
The Series is sorted by value in descending order (of value count) as a convenience.
value_counts is also available as a top-level pandas method that can be used with any
array or
sequence
```

In [57]:

```
obj.values
```

Out[57]:

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

In [65]:

```
pd.value_counts(obj.values, sort=False) # defalut is sort=True
```

Out[65]:

```
c    3
a    3
d    1
b    2
dtype: int64
```

```

```

```
isin()
- performs a vectorized set membership check and can be useful in filtering a dataset
down to a subset of values in a Series or column in a DataFrame
```

In [45]:

```
obj
```

Out[45]:

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

In [66]:

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

Out[66]:

```
0     True
1    False
2    False
3    False
4    False
5     True
6     True
7     True
8     True
dtype: bool
```

In [47]:

```
obj[mask]
```

Out[47]:

```
0    c
5    b
6    b
7    c
8    c
dtype: object
```

```
Index.get_indexer method
- gives you an index array from an array of possibly non-distinct values
into another array of distinct values
```

In [48]:

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

In [49]:

```
to_match
```

Out[49]:

```
0    c
1    a
2    b
3    b
4    c
5    a
dtype: object
```

In [50]:

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

In [ ]:

```

```

In [51]:

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

Out[51]:

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

In [68]:

```
pd.Index(to_match).get_indexer(unique_vals) # incorrect usage
```

```
---------------------------------------------------------------------------
--
InvalidIndexError                         Traceback (most recent call las
t)
C:\Users\MONAAD~1\AppData\Local\Temp/ipykernel_18556/2415082616.py in <mo
dule>
----> 1 pd.Index(to_match).get_indexer(unique_vals) # incorrect usage

~\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\core\in
dexes\base.py in get_indexer(self, target, method, limit, tolerance)
   3719
   3720         if not self._index_as_unique:
-> 3721             raise InvalidIndexError(self._requires_unique_msg)
   3722
   3723         if len(target) == 0:

InvalidIndexError: Reindexing only valid with uniquely valued Index objec
ts
```

Table 5-9. Unique, value counts, and set membership methods

Method	Description
`isin`	Compute boolean array indicating whether each Series value is contained in the passed sequence of values
`match`	Compute integer indices for each value in an array into another array of distinct values; helpful for data alignment and join-type operations
`unique`	Compute array of unique values in a Series, returned in the order observed
`value_counts`	Return a Series containing unique values as its index and frequencies as its values, ordered count in descending order

```
to compute a histogram on multiple related columns in a DataFrame
```

In [52]:

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

Out[52]:

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

In [70]:

```
#Passing pandas.value_counts to this DataFrame's apply function gives:
result = data.apply(pd.value_counts).fillna(0)
result
```

Out[70]:

	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