

# Sessions 03-05

## **Data** Manipulation and Visualization

(in **Python** and **MATLAB**)



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



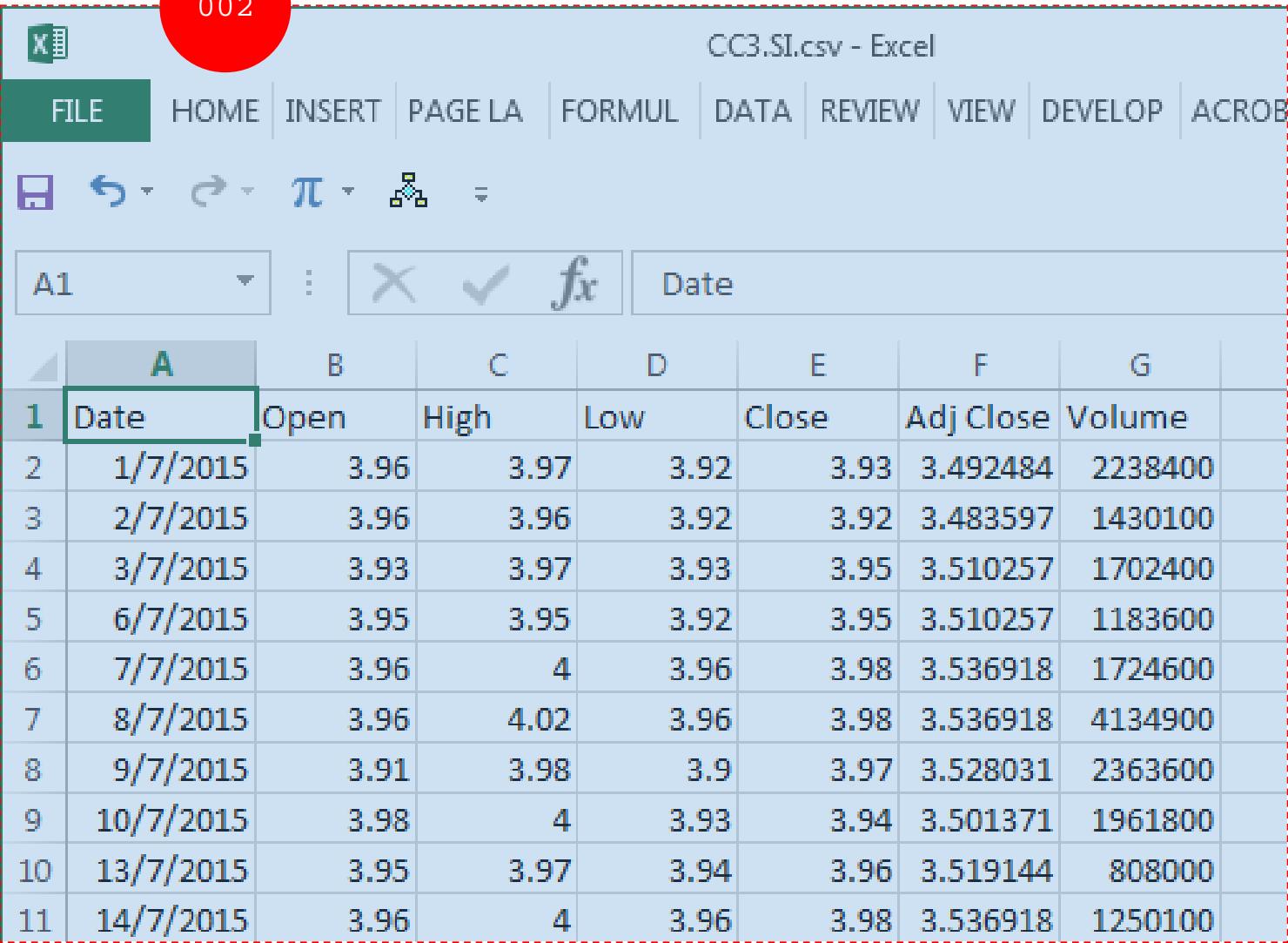
# Python Basics



# MATLAB Basics

Next, ...

CC3.SI.csv - Excel



	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	1/7/2015	3.96	3.97	3.92	3.93	3.492484	2238400
3	2/7/2015	3.96	3.96	3.92	3.92	3.483597	1430100
4	3/7/2015	3.93	3.97	3.93	3.95	3.510257	1702400
5	6/7/2015	3.95	3.95	3.92	3.95	3.510257	1183600
6	7/7/2015	3.96	4	3.96	3.98	3.536918	1724600
7	8/7/2015	3.96	4.02	3.96	3.98	3.536918	4134900
8	9/7/2015	3.91	3.98	3.9	3.97	3.528031	2363600
9	10/7/2015	3.98	4	3.93	3.94	3.501371	1961800
10	13/7/2015	3.95	3.97	3.94	3.96	3.519144	808000
11	14/7/2015	3.96	4	3.96	3.98	3.536918	1250100

ave15



ave50



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

.CSV → Moving Average Crossover



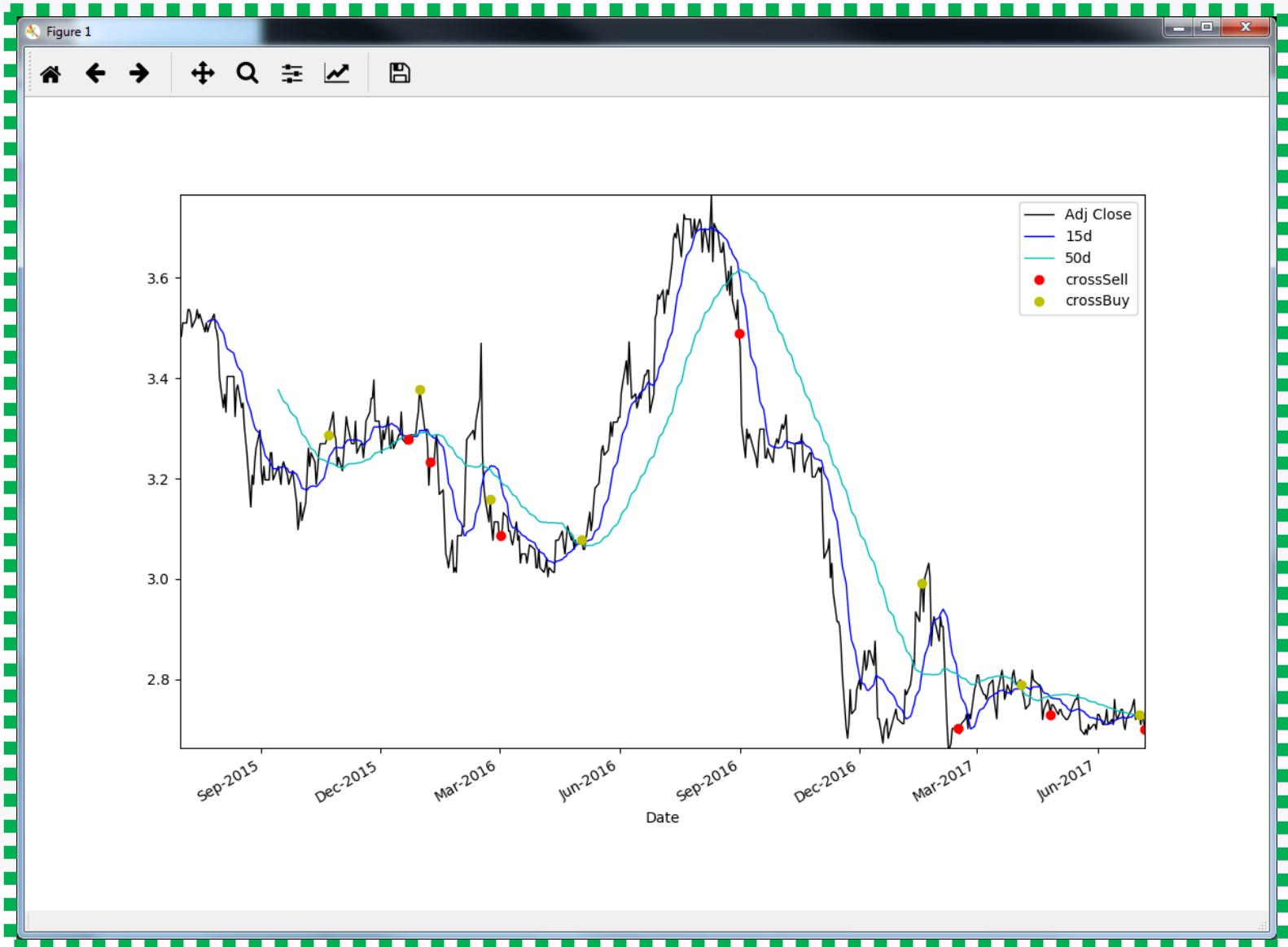
QF666

Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Python



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## How to ...

1. Import data from a CSV file to a **Table/DataFrame**
2. Manipulate data in a **Table/DataFrame**
  - Remove/Insert data
  - Compare data
  - Select data
  - Apply (other) operations/functions on data
3. Visualize data

001



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
1 # Moving Average Crossover
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import matplotlib.dates as mdates
6 plt.rcParams["figure.figsize"]=[12,8] # (optional)
7 data=pd.read_csv('CC3.SI.csv',index_col=0,parse_dates=True)
8 data.drop(data.index[data['Volume']==0],inplace=True)
9 data['15d']= np.round(data['Adj Close'].rolling(window=15).mean(),3)
10 data['50d']= np.round(data['Adj Close'].rolling(window=50).mean(),3)
11 x=data['15d']-data['50d']
12 x[x>0]=1
13 x[x<=0]=0
14 y=x.diff()
15 idxSell=y.index[y<0]
16 idxBuy=y.index[y>0]
17 data['crossSell']=np.nan
18 data.loc[idxSell,'crossSell']=data.loc[idxSell,'Adj Close']
19 data['crossBuy']=np.nan
20 data.loc[idxBuy,'crossBuy']=data.loc[idxBuy,'Adj Close']
21 fig, ax = plt.subplots()
22 data[['Adj Close', '15d', '50d','crossSell','crossBuy']].plot(
23     ax=ax,
24     style=['k-','b-','c-','ro','yo'],
25     linewidth=1)
26 plt.autoscale(enable=True, axis='x', tight=True)
27 plt.autoscale(enable=True, axis='y', tight=True)
28 ax.xaxis.set_major_formatter(mdates.DateFormatter('%b-%Y'))
29 plt.show()
```

**NEW:**  
✓ Add y label  
✓ Add title

✓ plt.ylabel  
✓ plt.title



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

004

```
1 - clear;
2 - clc;
3 - figure('Name','Figure 1', 'units','inch','position',[1.5,1.5,12,8]);
4 - data=readtable('CC3.SI.csv');
5 - data(data.Volume==0,:)=[];
6 - X=datetimelimits(data.Date);
7 - Y=data.AdjClose;
8 - plot(X,Y,'k-','LineWidth',1);
9 - hold on;
10 - ave15=round(movmean(Y,15,'Endpoints','discard'),3);
11 - ave15(1:35)=[];
12 - ave50=round(movmean(Y,50,'Endpoints','discard'),3);
13 - daxis=X(50:end);
14 - paxis=Y(50:end);
15 - plot(daxis,ave15,'b-');
16 - plot(daxis,ave50,'c-');
17 - x=ave15-ave50;
18 - x(x>0)=1;
19 - x(x<=0)=0;
20 - y=diff(x); %size is reduced by 1
21 - idxSell=find(y<0)+1;
22 - idxBuy=find(y>0)+1;
23 - plot(daxis(idxBuy),paxis(idxBuy), ...
24 -         'y','MarkerSize',20,'Linewidth',1);
25 - plot(daxis(idxDell),paxis(idxDell), ...
26 -         'r','MarkerSize',20,'Linewidth',1);
27 - legend('Adj Close', '15d', '50d', 'crossSell', 'crossBuy');
28 - xlabel('Date');
29 - axis tight
30 - set(gca,'XTickLabelRotation',30)
```

**NEW:**  
✓ Add y label  
✓ Add title

✓ ylabel  
✓ title



QF666

Programming and Computational Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

Spyder (Python 3.6)

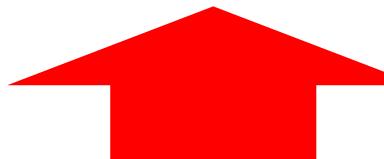
File Edit Search Source Run Debug Consoles Projects Tools View Help

Variable explorer Editor - MAC.py

MAC.py\*

Name	Type
ax	axes._subplots.AxesSubplot
data	DataFrame
fig	figure.Figure
idxBuy	DatetimeIndex
idxSell	DatetimeIndex
mdates	module
np	module
pd	module
plt	module
x	Series
y	Series

```
1 # Moving Average Crossover
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import matplotlib.dates as mdates
6 plt.rcParams["figure.figsize"]=[12,8] # (optional)
7 data=pd.read_csv('CC3.SI.csv',index_col=0,parse_dates=True)
8 data.drop(data.index[data['Volume']==0],inplace=True)
9 data['15d']= np.round(data['Adj Close'].rolling(window=15).mean(),3)
10 data['50d']= np.round(data['Adj Close'].rolling(window=50).mean(),3)
11 x=data['15d']-data['50d']
12 x[x>0]=1
13 x[x<=0]=0
14 y=x.diff()
15 idxSell=y.index[y<0]
16 idxBuy=y.index[y>0]
17 data['crossSell']=np.nan
18 data.loc[idxSell,'crossSell']=data.loc[idxSell,'Adj Close']
19 data['crossBuy']=np.nan
20 data.loc[idxBuy,'crossBuy']=data.loc[idxBuy,'Adj Close']
21 fig, ax = plt.subplots()
22 data[['Adj Close', '15d', '50d','crossSell','crossBuy']].plot(
23     ax=ax,
24     style=['k-','b-','c-','ro','yo'],
25     linewidth=1)
26 plt.autoscale(enable=True, axis='x', tight=True)
27 plt.autoscale(enable=True, axis='y', tight=True)
28 ax.xaxis.set_major_formatter(mdates.DateFormatter('%b-%Y'))
29 plt.show()
```



005



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

Name ▲	Value	Size	Class
ave15	457x1 double	457x1	double
ave50	457x1 double	457x1	double
data	506x7 table	506x7	table
31 daxis	457x1 datetime	457x1	datetime
idxBuy	[28;76;111;159;3...	7x1	double
idxSell	[70;81;117;243;3...	7x1	double
paxis	457x1 double	457x1	double
x	457x1 double	457x1	double
31 X	506x1 datetime	506x1	datetime
y	456x1 double	456x1	double
Y	506x1 double	506x1	double

MA.m

```
1 - clear;
2 - clc;
3 - figure('Name','Figure 1', 'units','inch','position',[1.5,1.5,12,8]);
4 - data=readtable('CC3.SI.csv');
5 - data(data.Volume==0,:)=[];
6 - X=datetime(data.Date);
7 - Y=data.AdjClose;
8 - plot(X,Y,'k-','LineWidth',1);
9 - hold on;
10 - ave15=round(movmean(Y,15,'Endpoints','discard'),3);
11 - ave15(1:35)=[];
12 - ave50=round(movmean(Y,50,'Endpoints','discard'),3);
13 - daxis=X(50:end);
14 - paxis=Y(50:end);
15 - plot(daxis,ave15,'b-');
16 - plot(daxis,ave50,'c-');
17 - x=ave15-ave50;
18 - x(x>0)=1;
19 - x(x<=0)=0;
20 - y=diff(x); %size is reduced by 1
21 - idxSell=find(y<0)+1;
22 - idxBuy=find(y>0)+1;
23 - plot(daxis(idxBuy),paxis(idxBuy), ...
24 -         'y','MarkerSize',20,'LineWidth',1);
25 - plot(daxis(idxSell),paxis(idxSell), ...
26 -         'r','MarkerSize',20,'LineWidth',1);
27 - legend('Adj Close', '15d', '50d', 'crossSell', 'crossBuy');
28 - xlabel('Date');
29 - axis tight
30 - set(gca,'XTickLabelRotation',30)
```



006



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Python First!

007

# import ... as ...

008

```
02 import pandas as pd
03 import numpy as np
04 import matplotlib.pyplot as plt
05 import matplotlib.dates as mdates
```

```
plt.rcParams["figure.figsize"]=[12, 8]
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 01: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
plt.rcParams["figure.figsize"]=[12,8] # (optional)
dir()
```

```
['In',
 'Out',
 '_',
 '_1',
 '_2',
 '_3',
 '__builtin__',
 '__builtins__',
 '__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_dh',
 '_i',
 '_i1',
 '_i2',
 '_ih',
 '_ii',
 '_iii',
 '_oh',
 'exit',
 'get_ipython',
 'mdates',
 'np',
 'pd',
 'plt',
 'quit']
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 02: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
dir(pd)
```

```
pd.read_csv
```

```
'pknow',
'qcut',
'read_clipboard',
'read_csv',
'read_excel',
'read_fwf',
'read_gbq',
'read_hdf',
'read_html',
'read_json',
'read_msgpack',
'read_pickle',
'read_sas',
'read_sql',
'read_sql_query',
'read_sql_table',
'read_stata',
'read_table',
'reset_option',
'rolling_apply',
```



# pd.read\_csv

[https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)

## Read CSV (comma separated values) file into DataFrame

```
pandas.read_csv(filepath_or_buffer, sep=', ', delimiter=None,  
header='infer', names=None, index_col=None, usecols=None,  
squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None,  
engine=None, converters=None, true_values=None, false_values=None,  
skipinitialspace=False, skiprows=None, nrows=None, na_values=None,  
keep_default_na=True, na_filter=True, verbose=False,  
skip_blank_lines=True, parse_dates=False,  
infer_datetime_format=False, keep_date_col=False, date_parser=None,  
dayfirst=False, iterator=False, chunksize=None,  
compression='infer', thousands=None, decimal=b'.',  
lineterminator=None, quotechar='"', quoting=0, escapechar=None,  
comment=None, encoding=None, dialect=None, tupleize_cols=False,  
error_bad_lines=True, warn_bad_lines=True, skipfooter=0,  
skip_footer=0, doublequote=True, delim_whitespace=False,  
as_recarray=False, compact_ints=False, use_unsigned=False,  
low_memory=True, buffer_lines=None, memory_map=False,  
float_precision=None)
```

010

011

013

(Dr. Z: Is there a  
\*- or \*\*-  
parameter?)



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



## header : *int or list of ints, default 'infer'*

Row number(s) to use as the column names, and the start of the data. Default behavior is to infer the column names: if no names are passed the behavior is identical to `header=0` and column names are inferred from the first line of the file, if column names are passed explicitly then the behavior is identical to `header=None`. Explicitly pass `header=0` to be able to replace existing names. The header can be a list of integers that specify row locations for a multi-index on the columns e.g. [0,1,3]. Intervening rows that are not specified will be skipped (e.g. 2 in this example is skipped). Note that this parameter ignores commented lines and empty lines if `skip_blank_lines=True`, so `header=0` denotes the first line of data rather than the first line of the file.

`data=pd.read_csv('CC3.SI.csv', header='infer')`

↔ `header=0`

A	B	C	D	E	F	G	
1	Date	Open	High	Low	Close	Adj Close	Volume
2	1/7/2015	3.96	3.97	3.92	3.93	3.492484	2238400
3	2/7/2015	3.96	3.96	3.92	3.92	3.483597	1430100
4	3/7/2015	3.93	3.97	3.93	3.95	3.510257	1702400

505	4/7/2017	2.71	2.73	2.7	2.71	2.71	2220000
506	5/7/2017	2.7	2.75	2.7	2.73	2.73	1850600
507	6/7/2017	2.73	2.74	2.7	2.72	2.72	2028600
508	7/7/2017	2.71	2.72	2.7	2.7	2.7	1702100

`header='infer'`

`header=0, w/o name`

In-Class Exercise 03: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data=pd.read_csv('CC3.SI.csv')  
data
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
1	2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2	2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
3	2015-07-06	3.95	3.95	3.91	3.95	3.510257	1183600
4	2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600
5	2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

`data=pd.read_csv('CC3.SI.csv', header=None)`

	Date	Open	High	Low	Close	Adj Close	Volume
1	1/7/2015	3.96	3.97	3.92	3.93	3.492484	2238400
2	2/7/2015	3.96	3.96	3.92	3.92	3.483597	1430100
3	3/7/2015	3.93	3.97	3.93	3.95	3.510257	1702400
505	4/7/2017	2.71	2.73	2.7	2.71	2.71	2220000
506	5/7/2017	2.7	2.75	2.7	2.73	2.73	1850600
507	6/7/2017	2.73	2.74	2.7	2.72	2.72	2028600
508	7/7/2017	2.71	2.72	2.7	2.7	2.7	1702100

header=None

In-Class Exercise 04: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data=pd.read_csv('CC3.SI.csv', header=None)  
data
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-07-01	3.960000	3.970000	3.920000	3.930000	3.492484	2238400
1	2015-07-02	3.960000	3.960000	3.920000	3.920000	3.483597	1430100
2	2015-07-03	3.930000	3.970000	3.930000	3.950000	3.510257	1702400
3	2015-07-06	3.950000	3.950000	3.950000	3.950000	3.510257	183600
4	2015-07-07	3.980000	4.000000	3.980000	3.980000	3.516918	1724600
5	2015-07-08	3.960000	4.020000	3.960000	3.980000	3.536918	4134900
6	2015-07-09	3.910000	3.980000	3.900000	3.970000	3.528031	2363600
7	2015-07-10	3.980000	4.000000	3.930000	3.940000	3.501371	1961800

dataframe



```
data=pd.read_csv('CC3.SI.csv', names=range(7,0,-1))
```

A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close
2	1/7/2015	3.96	3.97	3.92	3.93	3.492484
3	2/7/2015	3.96	3.96	3.92	3.92	3.483597
4	3/7/2015	3.93	3.97	3.93	3.95	3.510257
505	4/7/2017	2.71	2.73	2.7	2.71	2.71
506	5/7/2017	2.7	2.75	2.7	2.73	2.73
507	6/7/2017	2.73	2.74	2.7	2.72	2.72
508	7/7/2017	2.71	2.72	2.7	2.7	2.7

names=range(7,0,-1)

In-Class Exercise 05: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data=pd.read_csv('CC3.SI.csv', names=range(7,0,-1))  
data
```

7	6	5	4	3	2	1
0	Date	Open	High	Low	Close	Adj Close
1	2015-07-01	3.960000	3.970000	3.920000	3.930000	3.492484
2	2015-07-02	3.960000	3.960000	3.920000	3.920000	3.483597
3	2015-07-03	3.930000	3.970000	3.930000	3.950000	3.510257
4	2015-07-06	3.900000	3.950000	3.920000	3.910000	3.510257
5	2015-07-07	3.900000	4.000000	3.960000	3.900000	3.536118
6	2015-07-08	3.960000	4.020000	3.960000	3.980000	3.536918
7	2015-07-09	3.910000	3.980000	3.900000	3.970000	3.528031
8	2015-07-10	3.980000	4.000000	3.930000	3.940000	3.501371

dataframe



```
data=pd.read_csv('CC3.SI.csv', header=0, names=range(7,0,-1))
```

In-Class Exercise 06: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data=pd.read_csv('CC3.SI.csv', header=0, names=range(7,0,-1))  
data
```

	7	6	5	4	3	2	1
0	2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
1	2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2	2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
3	2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600
4	2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600
5	2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900
6	2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600
7	2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800
8	2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000

header=0

+

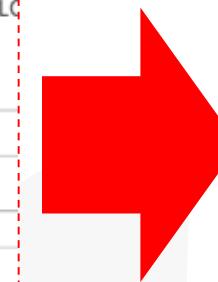
012

names=range(7,0,-1)



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

CC3.SI.csv - Excel						
1	Date	Open	High	Low	Close	Adj Close
2	1/7/2015	3.96	3.97	3.92	3.93	3.492484
3	2/7/2015	3.96	3.96	3.92	3.92	3.483597
4	3/7/2015	3.93	3.97	3.93	3.95	3.510257



	7	6	5	4	3	2	1
0	Date	Open	High	Low	Close	Adj Close	Volume
1	2015-07-01	3.960000	3.970000	3.920000	3.930000	3.492484	2238400
2	2015-07-02	3.960000	3.960000	3.920000	3.920000	3.483597	1430100
3	2015-07-03	3.930000	3.970000	3.930000	3.950000	3.510257	1702400
4	2015-07-06	3.950000	3.950000	3.920000	3.950000	3.510257	1183600
5	2015-07-07	3.960000	4.000000	3.960000	3.980000	3.536918	1724600
6	2015-07-08	3.960000	4.020000	3.960000	3.980000	3.536918	4134900
7	2015-07-09	3.910000	3.980000	3.900000	3.970000	3.528031	2363600
8	2015-07-10	3.980000	4.000000	3.930000	3.940000	3.501371	1961800

- (A) `data=pd.read_csv('CC3.SI.csv', header=0, names=range(7,0,-1))`
- (B) `data=pd.read_csv('CC3.SI.csv', names=range(7,0,-1))`
- (C) `data=pd.read_csv('CC3.SI.csv', header=0)`
- (D) `data=pd.read_csv('CC3.SI.csv')`



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 07: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data=pd.read_csv('CC3.SI.csv', index_col=0)  
data
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600

CC3.SI.csv - Excel

	Date	Open	High	Low	Close	Adj Close	Volume
1	1/7/2015	3.96	3.97	3.92	3.93	3.492484	2238400
2	2/7/2015	3.96	3.96	3.92	3.92	3.483597	1430100
3	3/7/2015	3.93	3.97	3.93	3.95	3.510257	1702400
4	4/7/2017	2.71	2.73	2.7	2.71	2.71	2220000
505	5/7/2017	2.7	2.75	2.7	2.73	2.73	1850600
506	6/7/2017	2.73	2.74	2.7	2.72	2.72	2028600
507	7/7/2017	2.71	2.72	2.7	2.7	2.7	1702100

012

**index\_col=0**

Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

**index\_col : int or sequence or False, default None**

Column to use as the row labels of the DataFrame. If a sequence is given, a MultiIndex is used. If you have a malformed file with delimiters at the end of each line, you might consider index\_col=False to force pandas to not use the first column as the index (row names)



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# DataFrame (data)

014

`data.columns`

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800

`data.index`



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 08: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data.columns, data.columns[0], data.columns[-3:]
```

```
(Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object'),  
 'Open',  
 Index(['Close', 'Adj Close', 'Volume'], dtype='object'))
```

014

### pandas.Index:

- ✓ Indexing
- ✓ Slicing

✓ list-like object (Python)

✓ array-like object, 1-D (Numpy)



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 09: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data.index, data.index[0], data.index[45:55]
```

```
data.index, data.index[0], data.index[45:55]
```

```
(Index(['2015-07-01', '2015-07-02', '2015-07-03', '2015-07-06', '2015-07-07',
       '2015-07-08', '2015-07-09', '2015-07-10', '2015-07-13', '2015-07-14',
       ...
       '2017-06-23', '2017-06-27', '2017-06-28', '2017-06-29', '2017-06-30',
       '2017-07-03', '2017-07-04', '2017-07-05', '2017-07-06', '2017-07-07'],
      dtype='object', name='Date', length=507),
 '2015-07-01',
 Index(['2015-09-04', '2015-09-07', '2015-09-08', '2015-09-09', '2015-09-10',
       '2015-09-14', '2015-09-15', '2015-09-16', '2015-09-17', '2015-09-18'],
      dtype='object', name='Date'))
```

- ✓ Indexing
- ✓ Slicing

014

pandas.Index: list-like/array-like



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



QF666  
Programming and  
Computational  
Finance

`data['Open']`

syntactic sugar

`data.Open``data['Adj Close']`

016

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600





Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 10: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data.Open
```

```
data.Open
```

Date	Open
2015-07-01	3.96
2015-07-02	3.96
2015-07-03	3.93
2015-07-06	3.95
2015-07-07	3.96
2015-07-08	3.96
2015-07-09	3.91
2015-07-10	3.98
2015-07-13	3.95
2015-07-14	3.96
2015-07-15	3.95
2015-07-16	3.97
2015-07-20	3.97
2015-07-21	3.95
2015-07-22	3.97
2015-07-23	3.94
2015-07-24	3.96
2015-07-27	3.97
2015-07-28	3.95

016

This is a Series.

In-Class Exercise 11: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data[ 'Open' ]
```

```
data[ 'Open' ]
```

Date	Open
2015-07-01	3.96
2015-07-02	3.96
2015-07-03	3.93
2015-07-06	3.95
2015-07-07	3.96
2015-07-08	3.96
2015-07-09	3.91
2015-07-10	3.98
2015-07-13	3.95
2015-07-14	3.96
2015-07-15	3.95
2015-07-16	3.97
2015-07-20	3.97
2015-07-21	3.95
2015-07-22	3.97
2015-07-23	3.94
2015-07-24	3.96
2015-07-27	3.97
2015-07-28	3.95

This is a Series.

In-Class Exercise 12: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
type(data.Open), type(data['Open'])
```

```
type(data.Open), type(data['Open'])
```

```
(pandas.core.series.Series, pandas.core.series.Series)
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

- ✓ `pandas.Series`: One-dimensional ndarray with axis labels.
- ✓ `pandas.DataFrame`: Two-dimensional size-mutable, tabular data structure with labeled axes (rows and columns). Can be thought of as a dict-like container for Series objects.





## DataFrame[ colname ]:

Series corresponding to  
colname

(Example: `data[ 'Open' ]`)

016



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 13: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
type(data.index), data.index[0], type(data.index[0])
```

```
type(data.index), data.index[0], type(data.index[0])
```

```
(pandas.core.indexes.base.Index, '2015-07-01', str)
```

In-Class Exercise 14: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data=pd.read_csv('CC3.SI.csv', index_col=0, parse_dates=True)  
data
```

```
data=pd.read_csv('CC3.SI.csv', index_col=0, parse_dates=True)  
data
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000

In-Class Exercise 15: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
type(data.index), data.index[0]
```

In [15]:

```
type(data.index), data.index[0]
```

Out[15]:

```
(pandas.core.indexes.datetimes.DatetimeIndex,  
Timestamp('2015-07-01 00:00:00'))
```





QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000

data.values  
(numpy.ndarray)

In-Class Exercise 16: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
type(data.values)
```

In [16]:

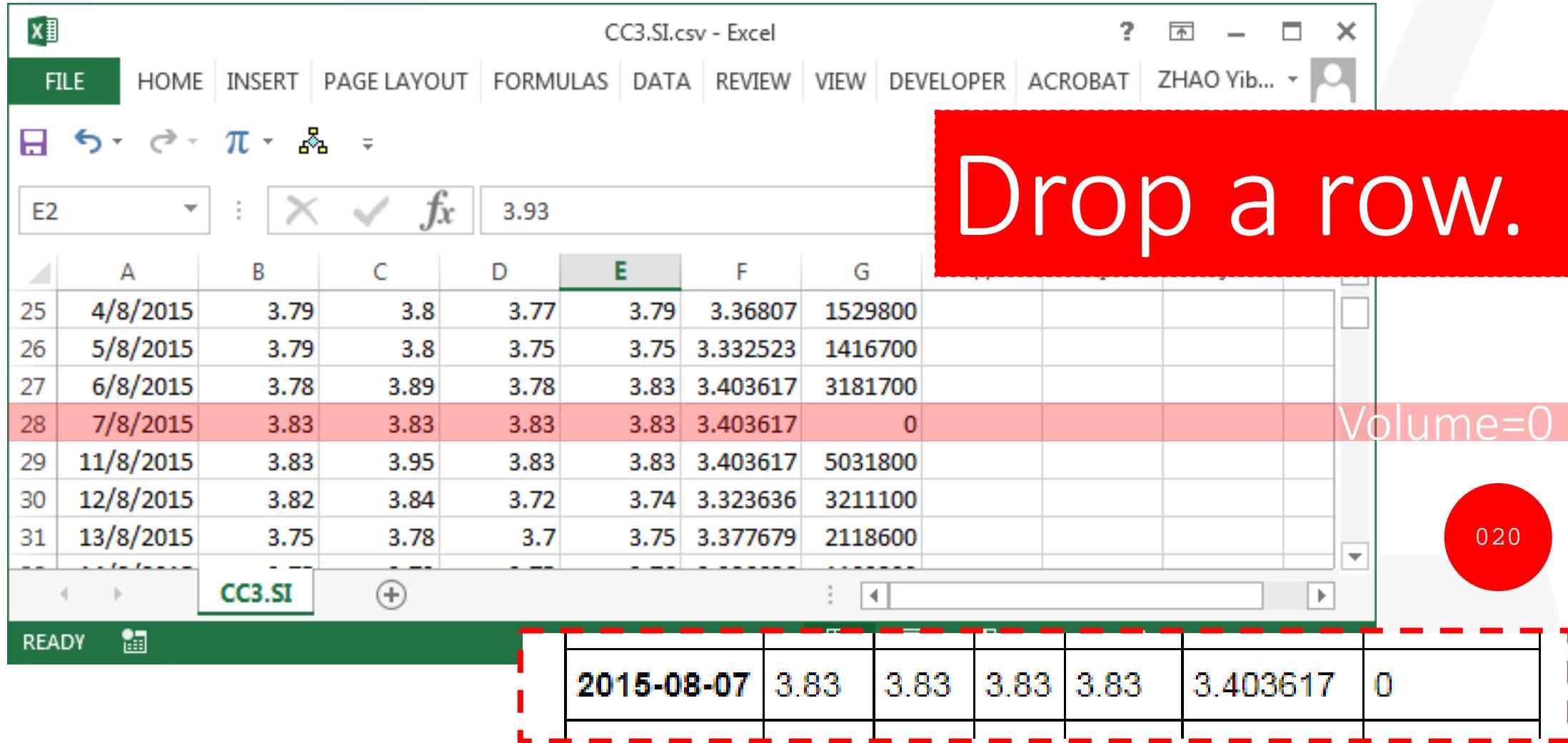
```
type(data.values)
```

Out[16]:

```
numpy.ndarray
```

`DataFrame.drop(labels, axis=0, level=None, inplace=False, errors='raise')`  
<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop.html>

`data.drop(data.index[data['Volume'] == 0], inplace=True)`



The screenshot shows a Microsoft Excel spreadsheet titled "CC3.SI.csv - Excel". The data consists of 7 rows (25 to 31) and 7 columns (A to G). Row 28 is highlighted in pink and has a red border, indicating it is selected for deletion. A large red callout box with the text "Drop a row." is positioned over the top right portion of the table. The formula bar shows "E2" and the value "3.93". The status bar at the bottom left shows "READY". A red dashed box highlights the row 28 data: 2015-08-07, 3.83, 3.83, 3.83, 3.83, 3.403617, 0.

	A	B	C	D	E	F	G
25	4/8/2015	3.79	3.8	3.77	3.79	3.36807	1529800
26	5/8/2015	3.79	3.8	3.75	3.75	3.332523	1416700
27	6/8/2015	3.78	3.89	3.78	3.83	3.403617	3181700
28	7/8/2015	3.83	3.83	3.83	3.83	3.403617	0
29	11/8/2015	3.83	3.95	3.83	3.83	3.403617	5031800
30	12/8/2015	3.82	3.84	3.72	3.74	3.323636	3211100
31	13/8/2015	3.75	3.78	3.7	3.75	3.377679	2118600

Volume=0

020

2015-08-07 3.83 3.83 3.83 3.83 3.403617 0



```
data.drop(data.index[data['Volume'] == 0], inplace=True)
```

In-Class Exercise 17: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data['Volume']
```

```
data['Volume']
```

Date

2015-07-01	2238400
2015-07-02	1430100
2015-07-03	1702400
2015-07-06	1183600
2015-07-07	1724600
2015-07-08	4134900
2015-07-09	2363600
2015-07-10	1961800
2015-07-13	808000
2015-07-14	1250100
2015-07-15	957300
2015-07-16	1621600
2015-07-20	2457000
2015-07-21	2794500
2015-07-22	761300
2015-07-23	765800
2015-07-24	671500
2015-07-27	1720900
2015-07-28	7065100

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	1/7/2015	3.96	3.97	3.92	3.93	3.492484	2238400
3	2/7/2015	3.96	3.96	3.92	3.92	3.483597	1430100
4	3/7/2015	3.93	3.97	3.93	3.95	3.510257	1702400

021

data['Volume']  
is a Series.



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
data.drop(data.index[data['Volume']==0], inplace=True)
```

In-Class Exercise 19: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data['Volume']==0
```

```
data['Volume']==0
2015-07-15    False
2015-07-16    False
2015-07-20    False
2015-07-21    False
2015-07-22    False
2015-07-23    False
2015-07-24    False
2015-07-27    False
2015-07-28    False
2015-07-29    False
2015-07-30    False
2015-07-31    False
2015-08-03    False
2015-08-04    False
2015-08-05    False
2015-08-06    False
2015-08-07    True
2015-08-11    False
2015-08-12    False
2015-08-13    False
```

data[ 'Volume' ] is a Series.

data[ 'Volume' ] == 0  
returns a Series of True and  
False with the same labels.

022



Comparison between  
a Series and a scalar  
results a Series.



# Boolean Indexing

```
>>> x=[1, 2, 3, 4, 5]
>>> x[[True, True, False, False, True]]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: list indices must be integers or slices, not list
```

Lists do not support Boolean indexing.

```
>>> import numpy as np
>>> y=np.array(x)
>>> y[[True, True, False, False, True]]
array([1, 2, 5])
```

Numpy ndarrays support Boolean indexing.

```
>>> import pandas as pd
>>> z=pd.Series(x)
>>> z[[True, True, False, False, True]]
0    1
1    2
4    5
dtype: int64
```

Serieses also support Boolean indexing.



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# (continued)

```
>>> z[[True, True, False, False]]  
Traceback (most recent call last):  
  File "C:\Continuum\anaconda3\lib\site-packages\pandas\core\series.py", line 87  
7, in _get_values  
    return self._constructor(self._data.get_slice(indexer),  
    File "C:\Continuum\  
4706, in get_slice  
    return self.__cl  
    File "C:\Continuum\  
312, in _slice  
    return self.value:  
IndexError: boolean i  
on is 5 but correspondin  
  
During handling of the  
above exception, another  
exception occurred:  
  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
  File "C:\Continuum\anaconda3\lib\site-packages\pandas\core\series.py", line 84  
0, in __getitem__  
    return self._get_with(key)  
  File "C:\Continuum\anaconda3\lib\site-packages\pandas\core\series.py", line 84  
6, in _get_with  
    return self._get_values(key)  
  File "C:\Continuum\anaconda3\lib\site-packages\pandas\core\series.py", line 88  
0, in _get_values  
    return self._values[indexer]  
IndexError: boolean index did not match indexed array along dimension 0; dimension  
is 5 but corresponding boolean dimension is 4
```

IndexError: boolean index  
did not match indexed array  
along dimension 0; dimension  
is 5 but corresponding  
boolean dimension is 4

023



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
data.drop(data.index[data['Volume']==0], inplace=True)
```

data.index

2015-08-07	3.83	3.83	3.83	3.83	3.403617	0
------------	------	------	------	------	----------	---

In-Class Exercise 20: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data.index[data['Volume']==0]
```

024

```
data.index[data['Volume']==0]
```

```
DatetimeIndex(['2015-08-07'], dtype='datetime64[ns]', name='Date', freq=None)
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
data.drop(data.index[data['Volume']==0], inplace=True)
```

In-Class Exercise 21: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data.drop(data.index[data['Volume']==0], inplace=True)
```

```
data.drop(data.index[data['Volume']==0], inplace=True)  
data
```

2015-08-06	3.78	3.89	3.78	3.83	3.403617	3181700
2015-08-11	3.83	3.95	3.83	3.83	3.403617	5031800
2015-08-12	3.82	3.84	3.72	3.74	3.323636	3211100
2015-08-13	3.75	3.78	3.70	3.75	3.377679	2118600
2015-08-14	3.75	3.79	3.75	3.76	3.386686	1103200
...	...	...	...	...	...	...
2017-05-26	2.72	2.72	2.69	2.70	2.700000	2767100
2017-05-29	2.70	2.71	2.68	2.71	2.710000	1995800
2017-05-30	2.71	2.72	2.68	2.71	2.710000	1707900
2017-05-31	2.71	2.74	2.70	2.70	2.700000	13187900
2017-06-01	2.73	2.75	2.72	2.73	2.730000	2969600
2017-06-02	2.72	2.74	2.72	2.72	2.720000	2102000

```
2015-08-07 3.83 3.83 3.83 3.83 3.403617 0
```

It's gone!

025

If using “`inplace=False`”, we’ll get a copy. `data` will remain unchanged.



```
>>> import numpy as np  
>>> import pandas as pd  
>>> df=pd.DataFrame(np.arange(12).reshape(3,4),  
...                   columns=list('ABCD')); df
```

```
   A   B   C   D  
0  0   1   2   3  
1  4   5   6   7  
2  8   9  10  11
```

```
>>> df.drop([0, 2], axis=0)  (default value)
```

```
   A   B   C   D  
1  4   5   6   7
```

**DataFrame.drop(labels, axis=0)**

0 / 'index'

```
>>>
```

```
>>>
```

```
>>> df.drop(['B','C'], axis=1)
```

```
   A   D  
0  0   3  
1  4   7  
2  8  11
```

**DataFrame.drop(labels, axis=1)**

1 / 'columns'

026



```
Series.rolling(window, min_periods=None, freq=None,  
center=False, win_type=None, on=None, axis=0,  
closed=None)
```

Provides rolling window calculations. (.sum( ), .mean( ), ...)

```
data['15d']= np.round(data['Adj Close'].rolling(window=15).mean(),3)  
data['50d']= np.round(data['Adj Close'].rolling(window=50).mean(),3)
```

We cannot use the “dot notation”  
to add a column to a DataFrame.

029

```
data['15d']= np.round(data['Adj Close'].rolling(15).mean(),3)  
data['50d']= np.round(data['Adj Close'].rolling(50).mean(),3)
```

- ✓ Series.rolling ⇒ Rolling Object
- ✓ RollingObject.mean() ⇒ Series

027

028



In-Class Exercise 22: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
# Series.rolling(window, min_periods=None, freq=None, center=False,  
#                           win_type=None, on=None, axis=0, closed=None)  
data['15d']= np.round(data['Adj Close'].rolling(15).mean(),3)  
data['50d']= np.round(data['Adj Close'].rolling(50).mean(),3)  
data
```

029

```
# Series.rolling(window, min_periods=None, freq=None, center=False,  
#                           win_type=None, on=None, axis=0, closed=None)  
data['15d']= np.round(data['Adj Close'].rolling(15).mean(),3)  
data['50d']= np.round(data['Adj Close'].rolling(50).mean(),3)  
data
```

Date	Open	High	Low	Close	Adj Close	Volume	15d	50d
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400	NaN	NaN
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100	NaN	NaN
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400	NaN	NaN
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600	NaN	NaN
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600	NaN	NaN
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900	NaN	NaN
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600	NaN	NaN
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800	NaN	NaN
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000	NaN	NaN

Two columns  
were added.

dict-like

(Dr. Z: How to add  
a new item in a  
dictionary?)

NaN?



	Open	High	Low	Close	Adj Close	Volume	15d	50d
Date								
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400	NaN	NaN
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100	NaN	NaN
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400	NaN	NaN
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600	NaN	NaN
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600	NaN	NaN
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900	NaN	NaN
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600	NaN	NaN
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800	NaN	NaN
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000	NaN	NaN
2015-07-14	3.96	4.00	3.96	3.98	3.536918	1250100	NaN	NaN
2015-07-15	3.95	3.98	3.95	3.96	3.519144	957300	NaN	NaN
2015-07-16	3.97	3.98	3.95	3.97	3.528031	1621600	NaN	NaN
2015-07-20	3.97	3.98	3.91	3.93	3.492484	2457000	NaN	NaN
2015-07-21	3.95	3.98	3.94	3.95	3.510257	2794500	NaN	NaN
2015-07-22	3.97	3.97	3.93	3.93	3.492484	761300	3.513	NaN
2015-07-23	3.94	3.97	3.94	3.94	3.501371	765800	3.514	NaN

030

.rolling(15).mean()



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 23: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data['Adj Close'].rolling(15).mean().round(3)
```

```
data['Adj Close'].rolling(15).mean().round(3)
```

Date

2015-07-01	NaN
2015-07-02	NaN
2015-07-03	NaN
2015-07-06	NaN
2015-07-07	NaN
2015-07-08	NaN
2015-07-09	NaN
2015-07-10	NaN
2015-07-13	NaN
2015-07-14	NaN
2015-07-15	NaN
2015-07-16	NaN
2015-07-20	NaN
2015-07-21	NaN
2015-07-22	3.513
2015-07-23	3.514
2015-07-24	3.516
2015-07-27	3.517

031

np.round(Series, 3)



Series.round(3)



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 24: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
x=data['15d']-data['50d']
type(x), x
```

```
x=data['15d']-data['50d']
type(x), x
```

```
(pandas.core.series.Series, Date
2015-07-01      NaN
2015-07-02      NaN
2015-07-03      NaN
2015-07-06      NaN
2015-07-07      NaN
2015-07-08      NaN
2015-07-09      NaN
2015-07-10      NaN
2015-07-13      NaN
2015-07-14      NaN
2015-07-15      NaN
2015-07-16      NaN
2015-07-20      NaN
2015-07-21      NaN
2015-07-22      NaN
2015-07-23      NaN
2015-07-24      NaN
2015-07-27      NaN
2015-07-28      NaN)
```

032

## Subtraction of Two Series (having the same labels)

(Dr. Z: What if they have different labels?)

x: pandas.core.series.Series

```
>>> import numpy as np
>>> np.nan-1
nan
>>> np.nan-np.nan
nan
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 25: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
x[x>0]=1  
x[x<=0]=0  
x
```



```
x[x>0]=1  
x[x<=0]=0  
x
```

- Boolean Indexing
- Assignment

Date  
2015-07-01 NaN  
2015-07-02 NaN  
2015-07-03 NaN  
2015-07-06 NaN  
2015-07-07 NaN  
2015-07-08 NaN  
2015-07-09 NaN  
2015-07-10 NaN  
2015-07-13 NaN  
2015-07-14 NaN  
2015-07-15 NaN  
2015-07-16 NaN  
2015-07-20 NaN  
2015-07-21 NaN  
2015-07-22 NaN  
2015-07-23 NaN  
2015-07-24 NaN  
2015-07-27 NaN  
2015-07-29 NaN

```
len(x[x>0])
```

143

```
x[x>0]=range(1,144)  
x[x>0]
```



Date  
2015-10-23 1.0  
2015-10-26 2.0  
2015-10-27 3.0  
2015-10-28 4.0  
2015-10-29 5.0  
2015-10-30 6.0

033

```
x[x>0]=[1,2]
```



ValueError

```
<ipython-input-23-de15ae361d6b>  
----> 1 x[x>0]=[1,2]
```



# Series.diff(periods=1, axis=0)

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.diff.html>

In-Class Exercise 26: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
y=x.diff()  
print(len(x), len(y))  
y
```

```
506 506  
Date  
2015-07-01    NaN  
2015-07-02    NaN  
2015-07-03    NaN  
2015-07-06    NaN  
2015-07-07    NaN
```

034

Series.diff  
returns a Series of  
the same length  
(and same labels).



## In-Class Exercise 27: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
y[(y>0) | (y<0)]
```

```
y[(y>0) | (y<0)]
```

- Boolean Indexing
- Bitwise Logical OR: | 

Date

2015-10-22 1.0

2015-12-22 -1.0

2015-12-31 1.0

2016-01-08 -1.0

2016-02-23 1.0

2016-03-02 -1.0

2016-05-03 1.0

2016-08-31 -1.0

2017-01-18 1.0

2017-02-14 -1.0

2017-04-04 1.0

2017-04-26 -1.0

2017-07-03 1.0

2017-07-07 -1.0

**dtype: float64**

**Operator**

lambda

**Description**

Lambda expression

if - else

Conditional expression

or

Boolean OR

and

Boolean AND

not x

Boolean NOT

in, not in, is, is not, <, <=, >, >=, !=, ==

Comparisons, including membership tests and identity tests

|

Bitwise OR

^

Bitwise XOR

&

Bitwise AND

<<, >>

Shifts

+, -

Addition and subtraction

\*, @, /, //, %

Multiplication, matrix multiplication division, remainder [5]

+x, -x, ~x

Positive, negative, bitwise NOT

\*\*

Powers

await

Asynchronous iteration

x[index], x[index:index], x(arguments...),  
x.attribute

Subscription, slicing, call, attribute reference

(expressions...), [expressions...], {key:  
value...}, {expressions...}

Binding or tuple display, list display, dictionary display, set display

035

036

???Bitwise OR ⇔ Element-wise OR???



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Python: **or** (not element-wise)

These are the Boolean operations, ordered by ascending priority:

2 or 3 :

Operation	Result	Notes
x or y	if x is false, then y, else x	(1)
x and y	if x is false, then x, else y	(2)
not x	if x is false, then True, else False	(3)

Truth Table (2)

x	y	x and y	x or y
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

```
>>> 2 or 3
```

```
2
```

```
>>> [2, 2] or [3, 3]
```

```
[2, 2]
```

```
>>> [0, 0] or [3, 3]
```

```
[0, 0]
```

```
>>> [] or [3, 3]
```

```
[3, 3]
```

037

New!



# Python: | (bitwise OR)

(for integers only)

(not element-wise)

```
>>> 1.1 | 2.2
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for |: 'float' and 'float'
```

```
>>> 1|2
```

```
3
```

```
>>> 1&2
```

```
0
```

```
>>> [1, 1] | [2, 2]
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for |: 'list' and 'list'
```

```
>>> [1, 1] & [2, 2]
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for &: 'list' and 'list'
```

1: 00000001

2: 00000010

1|2: 00000011

1&2: 00000000

**Do Not Test**



# Numpy `logical_or`

[https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.logical\\_or.html](https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.logical_or.html)

```
>>> 2 or 3
2
>>> [2, 2] or [3, 3]
[2, 2]
>>> [0, 0] or [3, 3]
[0, 0]
>>> [] or [3, 3]
[3, 3]
>>> 2 or [3, 3]
2
```

```
>>> import numpy as np
>>> np.logical_or(2,3)
True
>>> np.logical_or([2,2],[3,3])
array([ True,  True])
>>> np.logical_or([0,0],[3,3])
array([ True,  True])
>>> np.logical_or([], [3,3])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: operands could not be broadcast together with shapes (0,) (2,)
>>> np.logical_or([True,True,False],[True,False,False])
array([ True,  True, False])
>>> np.logical_or(True,[True,False,False])
array([ True,  True,  True])
```

Element-wise OR

039



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
>>> np.array([2,2]) or np.array([3,3])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: The truth value of an array with more than one element is ambiguous.
Use a.any() or a.all()
```

ndarray has no OR

# Numpy | (bitwise OR)

(element-wise)

```
>>> import numpy as np
>>> x=np.array([0, 1, 2])
>>> y=np.array([1, 2, 3])
>>> x|y
array([1, 3, 3], dtype=int32)
>>> x=np.array([True, True, False])
>>> y=np.array([True, False, False])
>>> x|y
array([ True,  True, False])
>>> int(True)
1
>>> int(False)
0
```

0:	00000000
1:	00000001
2:	00000010
3:	00000011



It is a common practice to use the Bitwise OR operator (|) on Numpy Logical ndarrays for the “Element-Wise OR” operation on logical arrays.

In-Class Exercise 28: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
idxSell=y.index[y<0]  
idxSell
```

- ✓ `y` is a Series
- ✓ `y.index` is DatetimeIndex type
- ✓ Boolean Indexing

```
idxSell=y.index[y<0]  
idxSell
```

```
DatetimeIndex(['2015-12-22', '2016-01-08', '2016-03-02', '2016-08-31',  
                '2017-02-14', '2017-04-26', '2017-07-07'],  
               dtype='datetime64[ns]', name='Date', freq=None)
```

In-Class Exercise 29: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
idxBuy=y.index[y>0]  
idxBuy
```

- ✓ `idxBuy`
- ✓ `idxSell`

```
idxBuy=y.index[y>0]  
idxBuy
```

```
DatetimeIndex(['2015-10-22', '2015-12-31', '2016-02-23', '2016-05-03',  
                '2017-01-18', '2017-04-04', '2017-07-03'],  
               dtype='datetime64[ns]', name='Date', freq=None)
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

In-Class Exercise 30: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data['crossSell']=np.nan  
data
```

	Open	High	Low	Close	Adj Close	Volume	15d	50d	crossSell
Date									
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400	NaN	NaN	NaN
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100	NaN	NaN	NaN
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400	NaN	NaN	NaN
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600	NaN	NaN	NaN
2015-07-07	3.96	4.00	3.96	3.98	3.536363	'2015-12-22', '2016-01-08', '2016-03-02', '2016-08-31', '2017-02-14', '2017-04-26', '2017-07-07'],			
2015-07-08	3.96	4.02	3.96	3.98	3.536363				
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600	NaN	NaN	NaN
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800	NaN	NaN	NaN
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000	NaN	NaN	NaN

✓ np.nan

Another column is created.



## In-Class Exercise 31: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

✓ Chained Index

```
data.loc[idxSell, 'crossSell']=data['Adj Close'][idxSell]  
data
```

```
data.loc[idxSell, 'crossSell']=data['Adj Close'][idxSell]  
data
```

- **.loc** Indexing
- Assignment

Date	Open	High	Low	Close	Adj Close	Volume	15d	50d	crossSell
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400	NaN	NaN	NaN
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100	NaN	NaN	NaN
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400	NaN	NaN	NaN
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600	NaN	NaN	NaN
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600	NaN	NaN	NaN
2015-07-08	3.96	4.02	3.96	3.98	3.536918	1724600	NaN	NaN	NaN
2015-07-09	3.91	3.98	3.90	3.91	3.501371	1901000	NaN	NaN	NaN
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1901000	NaN	NaN	NaN
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000	NaN	NaN	NaN

'2015-12-22', '2016-01-08', '2016-03-02', '2016-08-31',  
'2017-02-14', '2017-04-26', '2017-07-07'],

In-Class Exercise 32: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
data['crossBuy']=np.nan  
data.loc[idxBuy,'crossBuy']=data['Adj Close'][idxBuy]  
data
```

Similarly,

```
data['crossBuy']=np.nan  
data.loc[idxBuy,'crossBuy']=data['Adj Close'][idxBuy]  
data
```

' crossBuy '

	Open	High	Low	Close	Adj Close	Volume	15d	50d	crossSell	crossBuy
Date										
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400	NaN	NaN	NaN	NaN
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100	NaN	NaN	NaN	NaN
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400	NaN	NaN	NaN	NaN
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600	NaN	NaN	NaN	NaN
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600	NaN	NaN	NaN	NaN
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900	NaN	NaN	NaN	NaN
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600	NaN	NaN	NaN	NaN
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800	NaN	NaN	NaN	NaN
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000	NaN	NaN	NaN	NaN



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

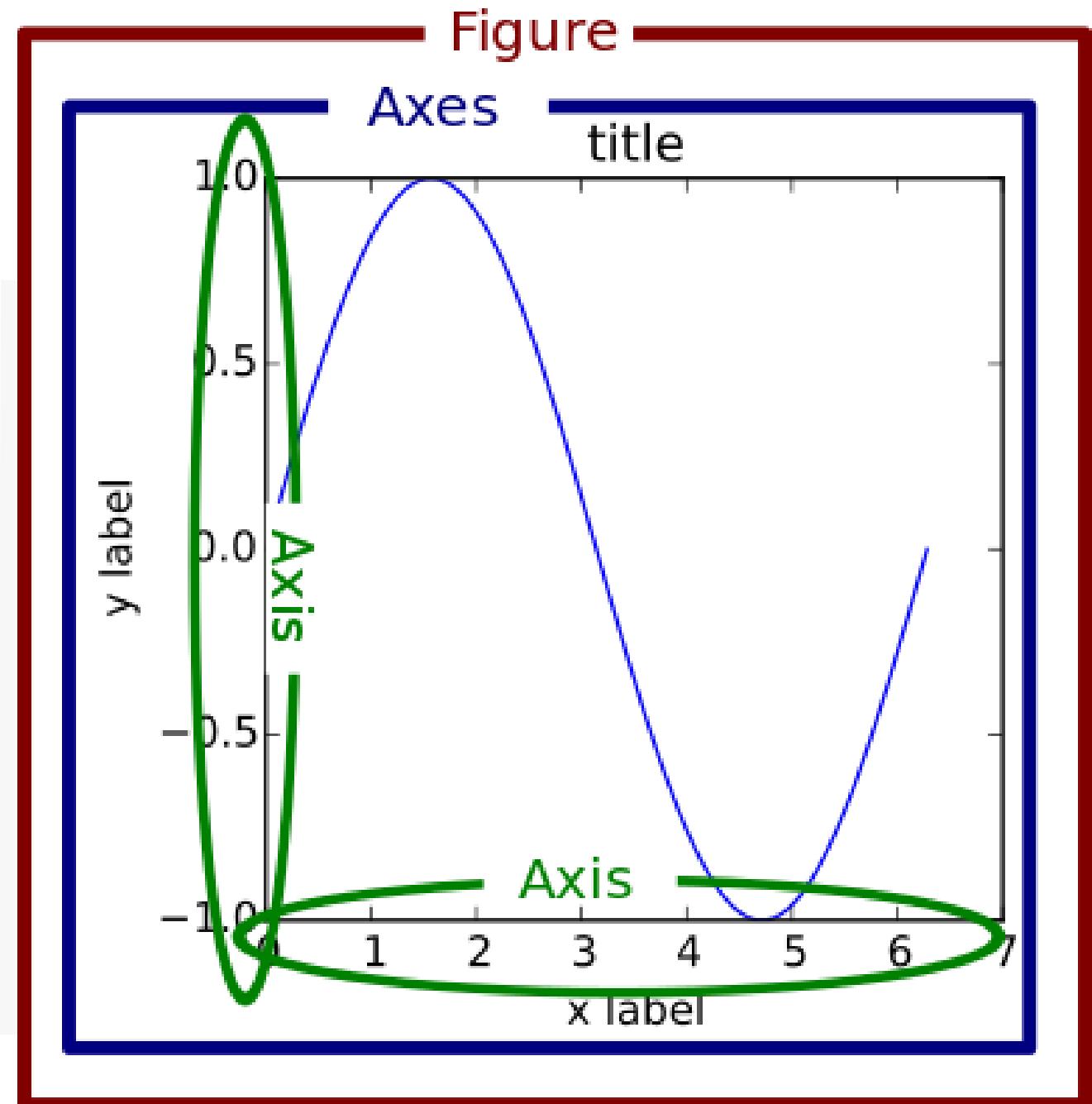
# ✓ Data Manipulation

# → Visualization

## What is ...

042

- ✓ Figure
- ✓ Axes
- ✓ Axis



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
fig, ax = plt.subplots()
```

[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.subplots.html](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplots.html)

## matplotlib.pyplot.subplots

```
matplotlib.pyplot.subplots(nrows=1, ncols=1, sharex=False, sharey=False,  
squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw) [source]
```

Create a figure and a set of subplots

043

```
nrows=2, ncols=2  
⇒ Numpy ndarray of Axes objects  
ax[0, 0] or ax[0][0]  
ax[0, 1] or ax[0][1]  
ax[1, 0] or ax[1][0]  
ax[1, 1] or ax[1][1]
```

✓ Figure: fig  
✓ Axes: ax





Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
python
(base) C:\Users\ybzhao>python
Python 3.6.4 |Anaconda custom (64-bit)| (default, Jan 16 2018, 10:22:32) [MSC v.
1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots(2,2)
>>> fig
<matplotlib.figure.Figure object at 0x0000000002B0F2B0>
>>> ax
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000000005C8B8D0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x0000000005CF16A0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000000005D2E6A0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000000061566A0>]],
      dtype=object)
>>> -
```

✓ Figure: fig  
✓ Axes: ax

043

```
fig = plt.figure()
```

Alternatively, ...

## matplotlib.pyplot.figure

```
matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None,  
frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kwargs)
```

[source]

Creates a new figure.

and make it the  
current figure.

044

```
ax=plt.axes()
```

✓ Figure: fig

✓ Axes: ax

## matplotlib.pyplot.axes

```
matplotlib.pyplot.axes(arg=None, **kwargs)
```

[source]

Add an axes to the current figure and make it the current axes.

✓ Current Figure

✓ Current Axes



# Confusing Matplotlib

<https://realpython.com/python-matplotlib-guide/>

Almost all functions from pyplot, such as plt.plot(), are implicitly either referring to an existing current Figure and current Axes, or creating them anew if none exist.

Therefore, everything in matplotlib is organized in a hierarchy. At the top of the hierarchy is the matplotlib “state-machine environment” which is provided by the `matplotlib.pyplot` module. At this level, simple functions are used to add plot elements (lines, images, text, etc.) to the current axes in the current figure.

At this level, simple functions are used to add plot elements (lines, images, text, etc.) to the current axes in the current figure.



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

For a complicated plot, it will be clearer and more convenient, if we

046

- ✓ Create figures with names

e.g. `fig1=plt.figure()`

- ✓ Add axes with names to a specific figure

[left, bottom, width, height]

e.g. `ax1=fig1.add_axes([0.1, 0.1, 0.8, 0.8])`

or `ax1=fig1.add_subplot(2, 2, 3)`

nrows, ncols, index

1	2
3	4

- ✓ Plot on axes

`plt.plot`  $\Rightarrow$  `ax1.plot`



[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.subplot2grid.html](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot2grid.html)

# matplotlib.pyplot.subplot2grid

Do Not Test

`matplotlib.pyplot.subplot2grid(shape, loc, rowspan=1, colspan=1, fig=None, **kwargs)`

Create an axis at specific location inside a regular grid.

**shape** : sequence of 2 ints

Shape of grid in which to place axis. First entry is number of rows, second entry is number of columns.

**loc** : sequence of 2 ints

Location to place axis within grid. First entry is row number, second entry is column number.

**rowspan** : int

Number of rows for the axis to span to the right.

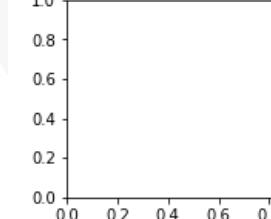
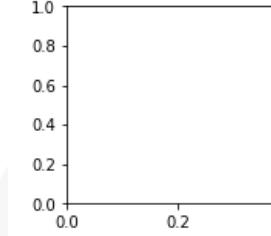
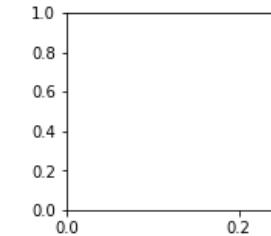
**colspan** : int

Number of columns for the axis to span downwards.

**fig** : Figure, optional

Figure to place axis in. Defaults to current figure.

```
f1=plt.figure(figsize=[10,8])
ax1=plt.subplot2grid((3,3),(0,0),colspan=3,fig=f1)
ax2=plt.subplot2grid((3,3),(1,0),colspan=2,fig=f1)
ax3=plt.subplot2grid((3,3),(1,1),rowspan=2,fig=f1)
ax4=plt.subplot2grid((3,3),(2,0),rowspan=2,fig=f1)
ax5=plt.subplot2grid((3,3),(2,1),rowspan=2,fig=f1)
```



047

Axes or axis?

???

???



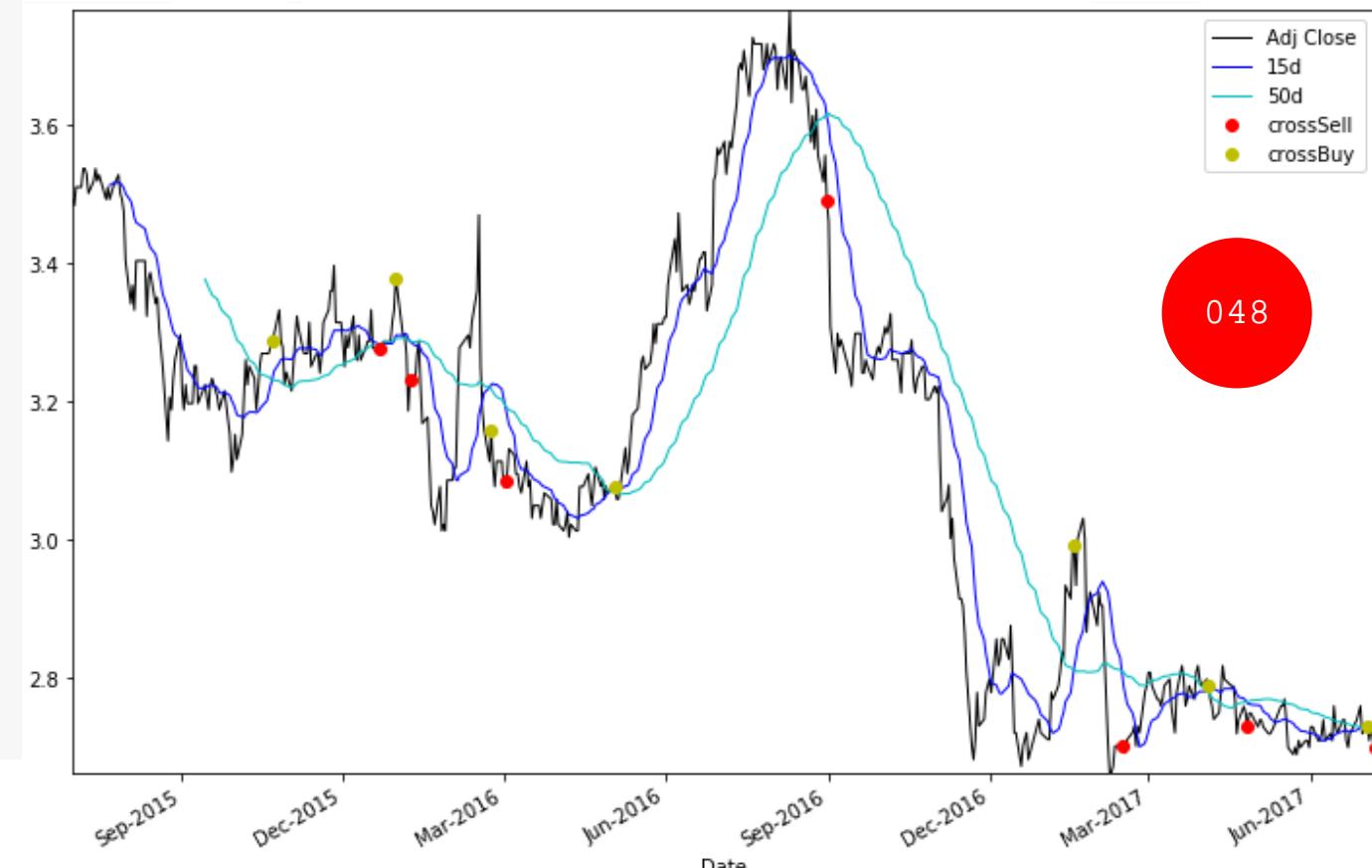
# pandas.DataFrame.plot

```
DataFrame.plot(x=None,  
y=None, kind='line',  
ax=None, subplots=False,  
sharex=None, sharey=False,  
layout=None, figsize=None,  
use_index=True,  
title=None, grid=None,  
legend=True, style=None,  
logx=False, logy=False,  
loglog=False, xticks=None,  
yticks=None, xlim=None,  
ylim=None, rot=None,  
fontsize=None,  
colormap=None,  
table=False, yerr=None,  
xerr=None,  
secondary_y=False,  
sort_columns=False,  
**kwds)
```

In-Class Exercise 33: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
fig, ax=plt.subplots()  
data[['Adj Close', '15d', '50d','crossSell','crossBuy']].plot(  
    ax=ax,  
    figsize=[12, 8],  
    style=['k-','b-','c-','ro','yo'],  
    linewidth=1)  
plt.autoscale(enable=True, axis='x', tight=True)  
plt.autoscale(enable=True, axis='y', tight=True)  
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b-%Y'))
```

? ax=ax



## Format Strings

A format string consists of a part for color, marker and line:

```
fmt = '[color][marker][line]'
```

Each of them is optional. If not provided, the value from the style cycle is used.  
Exception: If line is given, but no marker, the data will be a line without markers.

### Colors

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

If the color is the only part of the format string, you can additionally use any `matplotlib.colors` spec, e.g. full names ('green') or hex strings ('#008000').

### Markers

character	description
'.'	point marker
'.'	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
triangle_left marker	
triangle_right marker	
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'l'	vline marker
'_'	hline marker

### Line Styles

character	description
'-	solid line style
'--'	dashed line style
'-. '	dash-dot line style
'::'	dotted line style



## Colors

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

049



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

If the color is the only part of the format string, you can additionally use any `matplotlib.colors` spec, e.g. full names ('green') or hex strings ('#008000').



Dr. Zhao Yibao  
 Senior Lecturer  
 Of Quantitative  
 Finance

## Markers

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*''	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
' '	vline marker
'_'	hline marker

## Line Styles

character	description
'-'	solid line style
'--'	dashed line style
'-. '	dash-dot line style
'::'	dotted line style

Example format strings:

```
'b'    # blue markers with default shape
'r o'   # red circles
'g-'   # green solid line
'--'   # dashed line with default color
'k^:'  # black triangle_up markers connected by a dotted line
```

## In-Class Exercise 34: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
type(data[['Adj Close', '15d', '50d','crossSell','crossBuy']])
```

052

```
type(data[['Adj Close', '15d', '50d','crossSell','crossBuy']])
```

```
pandas.core.frame.DataFrame
```

	Open	High	Low	Close	Adj Close	Volume	15d	50d	crossSell	crossBuy
Date										
2015-07-01	3.96	3.97	3.92	3.93	3.492484	2238400	NaN	NaN	NaN	NaN
2015-07-02	3.96	3.96	3.92	3.92	3.483597	1430100	NaN	NaN	NaN	NaN
2015-07-03	3.93	3.97	3.93	3.95	3.510257	1702400	NaN	NaN	NaN	NaN
2015-07-06	3.95	3.95	3.92	3.95	3.510257	1183600	NaN	NaN	NaN	NaN
2015-07-07	3.96	4.00	3.96	3.98	3.536918	1724600	NaN	NaN	NaN	NaN
2015-07-08	3.96	4.02	3.96	3.98	3.536918	4134900	NaN	NaN	NaN	NaN
2015-07-09	3.91	3.98	3.90	3.97	3.528031	2363600	NaN	NaN	NaN	NaN
2015-07-10	3.98	4.00	3.93	3.94	3.501371	1961800	NaN	NaN	NaN	NaN
2015-07-13	3.95	3.97	3.94	3.96	3.519144	808000	NaN	NaN	NaN	NaN

# DataFrame





`DataFrame[list_colnames]:`

`DataFrame corresponding to  
list_colnames.`

(Example: `data[['Adj Close', '15d']]`)

(Dr. Z: It is also known as “fancy indexing”.)



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## matplotlib.pyplot.show

`matplotlib.pyplot.show(*args, **kw)`

Display a figure. When running in ipython with its pylab mode, display all figures and return to the ipython prompt.

In non-interactive mode, display all figures and block until the figures have been closed; in interactive mode it has no effect unless figures were created prior to a change from non-interactive to interactive mode (not recommended). In that case it displays the figures but does not block.

A single experimental keyword argument, *block*, may be set to True or False to override the blocking behavior described above.

%matplotlib inline



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

That's all for the explanation of the sample Python code.

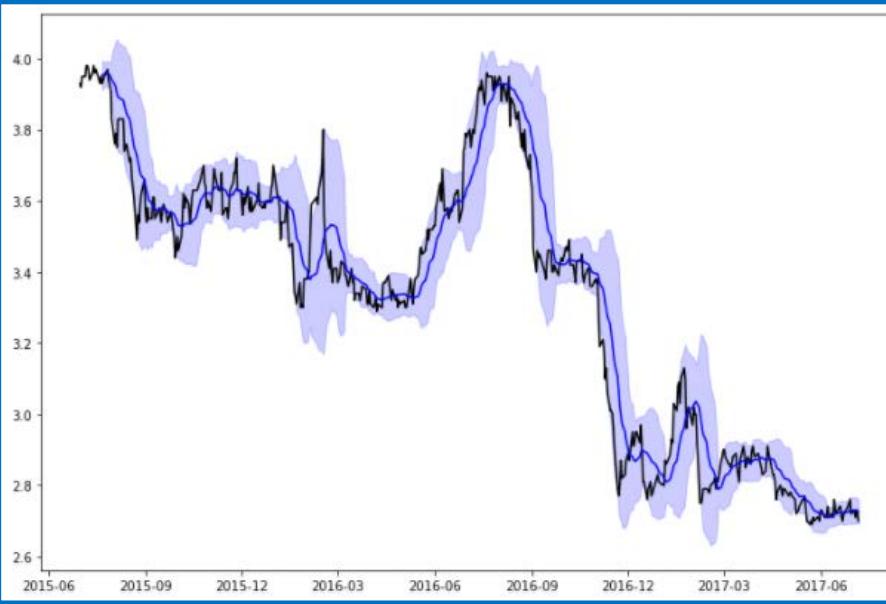


QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Next, ...



## fill\_between

## candlestick\_ohlc



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.fill\\_between.html](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.fill_between.html)



Version 2.1.2

x, y1, y2=0

\* \*kwargs

[home](#) | [examples](#) | [tutorials](#) | [pyplot](#) | [docs](#) » The Matplotlib API » [matplotlib.pyplot](#)

plt

## matplotlib.pyplot.fill\_between

`matplotlib.pyplot.fill_between(x, y1, y2=0, where=None, interpolate=False, step=None, hold=None, data=None, **kwargs)`

Make filled polygons between two curves.

Create a [PolyCollection](#) filling the regions between  $y_1$  and  $y_2$  where  $where==True$

053

054



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

[https://matplotlib.org/api/\\_as\\_gen/matplotlib.axes.Axes.fill\\_between.html](https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.fill_between.html)

Another `fill_between`.



Version 2.1.2

[home](#) | [examples](#) | [tutorials](#) | [pyplot](#) | [docs](#) » The Matplotlib API » Axes class »

053

## `matplotlib.axes.Axes.fill_between`

054

`Axes.fill_between(x, y1, y2=0, where=None, interpolate=False, step=None, *, data=None, **kwargs)`

Make filled polygons between two curves.

Create a `PolyCollection` filling the regions between `y1` and `y2` where `where==True`



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

**Parameters:****x** : array

An N-length array of the x data

**y1** : array

An N-length array (or scalar) of the y data

**y2** : array

An N-length array (or scalar) of the y data

**where** : array, optional

If **None**, default to fill between everywhere. If not **None**, it is an N-length numpy boolean array and the fill will only happen over the regions where **where==True**.

**interpolate** : bool, optional

If **True**, interpolate between the two lines to find the precise point of intersection. Otherwise, the start and end points of the filled region will only occur on explicit values in the **x** array.

**step** : {'pre', 'post', 'mid'}, optional

If not **None**, fill with step logic.

List also can! 😊

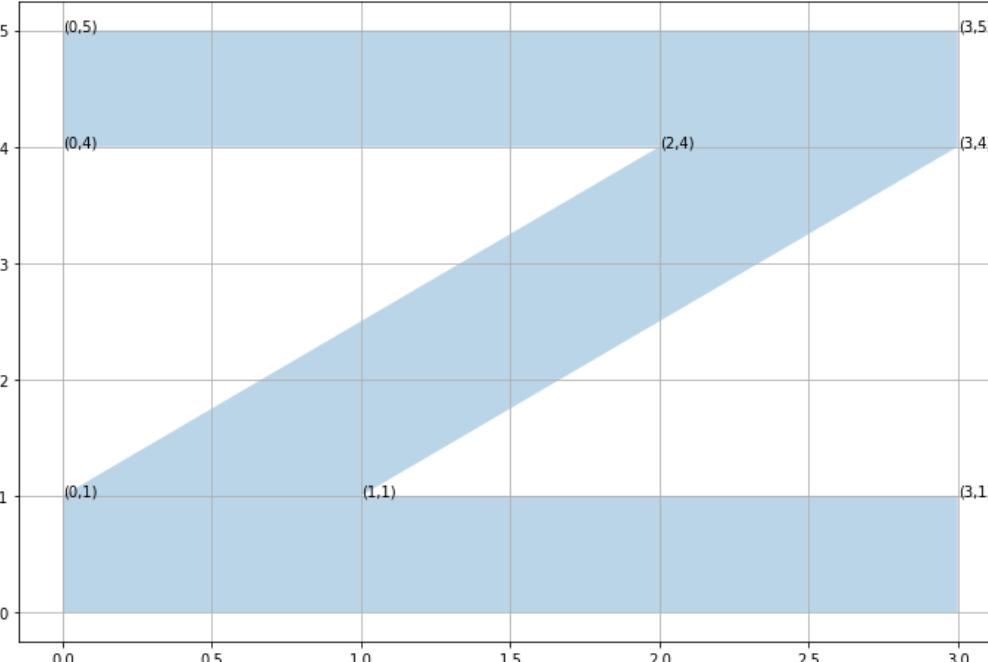
y1 can also be a  
scalar! 😊

# Let's play play.

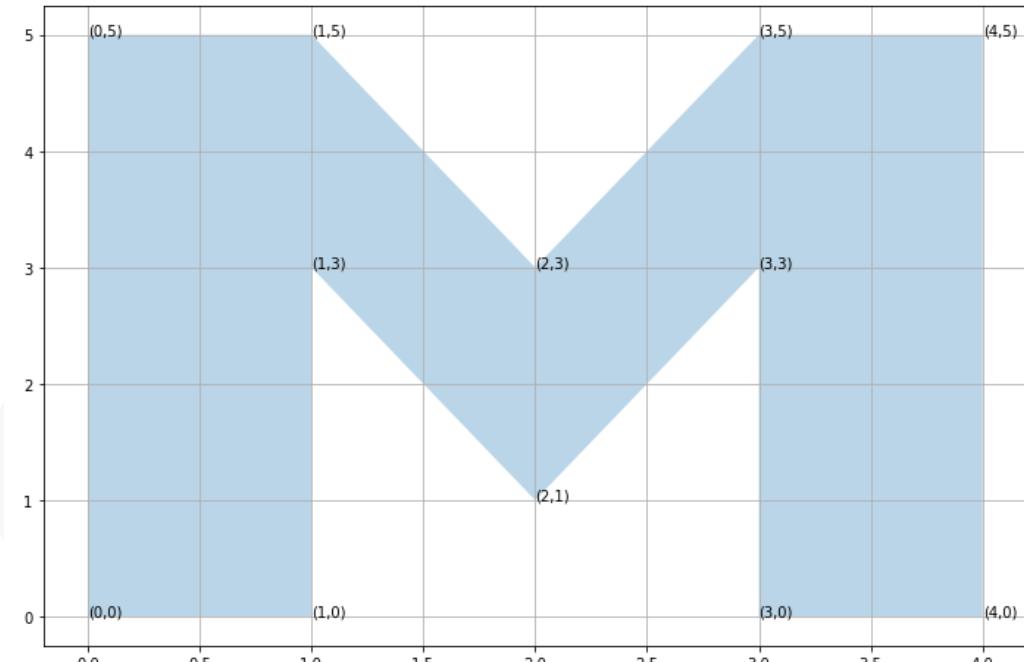
Dr. Z: Are you confused by the xy in the code.

055

```
#fill_between
x=[0,2,0,0,3,3,1,3]
y1=[1,4,4,5,5,4,1,1]
plt.fill_between(x,y1,alpha=0.3)
plt.grid(which='major')
for xy in zip(x,y1):
    plt.annotate('(%s,%s)' % (xy), xy=xy)
```



```
#fill_between
x=[0,1,2,3,4,4,3,3,2,1,1,0]
y1=[5,5,3,5,5,0,0,3,1,3,0,0]
plt.fill_between(x,y1,alpha=0.3)
plt.grid(which='major')
for xy in zip(x,y1):
    plt.annotate('(%s,%s)' % (xy), xy=xy)
```



- ✓ list
- ✓ tuple
- ✓ np.array
- ✓ pd.Series



## Notes

Additional Keyword args passed on to the `PolyCollection`.

kwargs control the `Polygon` properties:

# \*\*kwargs

Property	Description
<code>agg_filter</code>	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
<code>alpha</code>	float or None
<code>animated</code>	bool
<code>antialiased</code> or <code>antialiaseds</code>	Boolean or sequence of booleans
<code>array</code>	ndarray
<code>clim</code>	a length 2 sequence of floats
<code>clip_box</code>	a <code>Bbox</code> instance
<code>clip_on</code>	bool
<code>clip_path</code>	<code>[(Path, Transform)   Patch   None]</code>
<code>cmap</code>	a colormap or registered colormap name
<code>color</code>	matplotlib color arg or sequence of rgba tuples
<code>contains</code>	a callable function
<code>edgecolor</code> or <code>edgecolors</code>	matplotlib color spec or sequence of specs
<code>facecolor</code> or <code>facecolors</code>	matplotlib color spec or sequence of specs
<code>pickradius</code>	float distance in points
<code>rasterized</code>	bool or None
<code>sketch_params</code>	(scale: float, length: float, randomness: float)
<code>snap</code>	bool or None
<code>transform</code>	<code>Transform</code>
<code>url</code>	a url string
<code>urls</code>	List[str] or None
<code>visible</code>	bool
<code>zorder</code>	float


 Dr. Zhao Yibao  
 Senior Lecturer  
 Of Quantitative  
 Finance

x

y1

y2

```
# fill_between
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"]=[12,8] # (optional)

data=pd.read_csv('CC3.SI.csv',index_col=0,parse_dates=True)
data.drop(data.index[data['Volume']==0],inplace=True)

ma=data['Adj Close'].rolling(15).mean()
mstd=data['Adj Close'].rolling(15).std()

plt.plot(data.index, data['Adj Close'], 'k')
plt.plot(ma.index, ma, 'b')
plt.fill_between(mstd.index, ma-2*mstd, ma+2*mstd,
                 color='b', alpha=0.2)

plt.show()
```

They are **Serieses**.



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Fill\_Between

Moving Average (ma)

objectName: checkbox\_ma

Moving Std (mstd)

objectName: checkbox\_mstd

layoutName: verticalLayout



- ✓ 1 label
- ✓ 2 checkbox
- ✓ 1 axes

GUI

PyQt5



058

a .ui file +

```
1 import sys
2 from PyQt5.QtWidgets import QMainWindow, QApplication # more widgets
3 from PyQt5 import uic
4 # more imports
5
6 qtCreatorFile = "FromQtDesigner.ui"
7 Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
8
9 class Main(QMainWindow, Ui_MainWindow):
10     def __init__(self):
11         super().__init__()
12         self.setupUi(self)
13         # more initialization
14
15     # more functions/methods
16
17 if __name__ == '__main__':
18     app=QApplication(sys.argv)
19     main=Main()
20     main.show()
21     sys.exit(app.exec_())
```

PyQT5 template.py



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
1 import sys
2 from PyQt5.QtWidgets import QMainWindow, QApplication
3 from PyQt5 import uic
4 # more imports
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 from matplotlib.figure import Figure
8 from matplotlib.backends.backend_qt5agg \
9     import FigureCanvasQTAgg as FigureCanvas
10
11 qtCreatorFile = "AppFillBetween.ui"
12 Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
```

Template for adding a figure in a GUI.

058

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1059, 711)
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



QF666  
Programming and  
Computational  
Finance

```
14 class Main(QMainWindow, Ui_MainWindow):  
15     def __init__(self):  
16         super().__init__()  
17         self.setupUi(self) —————> Ui_MainWindow.setupUi  
18  
19         # more initialization  
20         self.data=pd.read_csv('CC3.SI.csv',index_col=0,parse_dates=True)  
21         self.data.drop(self.data.index[self.data['Volume']==0],inplace=True)  
22         self.ma=self.data['Close'].rolling(15).mean()  
23         self.mstd=self.data['Close'].rolling(15).std()  
24  
25         self.checkBox_ma.setChecked(True)  
26         self.checkBox_mstd.setChecked(True)  
27  
28         self.fig1 = Figure()  
29         self.ax1 = self.fig1.add_subplot(111)  
30         self.ax1.plot(self.data.index, self.data['Close'], 'k')  
31         self.plot2=self.ax1.plot(self.ma.index, self.ma, 'b')  
32         self.plot3=self.ax1.fill_between(self.mstd.index, self.ma-2*self.mstd,  
33                                         self.ma+2*self.mstd, color='b', alpha=0.2)  
34         self.canvas1 = FigureCanvas(self.fig1)  
35         self.verticalLayout.addWidget(self.canvas1)  
36         self.canvas1.draw()  
37  
38         self.checkBox_ma.stateChanged.connect(self.updategraph)  
39         self.checkBox_mstd.stateChanged.connect(self.updategraph)
```

self

058

(Dr. Z: The rest is to add an axes in a figure in a canvas widget in the verticalLayout in the QMainWindow.)



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
40
41     # more functions/methods
42     def updategraph(self):
43         if self.checkBox_ma.isChecked():
44             plt.setp(self.plot2, visible=True)
45         else:
46             plt.setp(self.plot2, visible=False)
47         if self.checkBox_mstd.isChecked():
48             plt.setp(self.plot3, visible=True)
49         else:
50             plt.setp(self.plot3, visible=False)
51         self.canvas1.draw()
52
53 if __name__ == '__main__':
54     app = QApplication(sys.argv)
55     main = Main()
56     main.show()
57     sys.exit(app.exec_())
```

self

058

plt.setp  
visible

[https://matplotlib.org/api/finance\\_api.html#matplotlib.finance.candlestick\\_ohlc](https://matplotlib.org/api/finance_api.html#matplotlib.finance.candlestick_ohlc)

```
matplotlib.finance.candlestick_ohlc(ax, quotes, width=0.2, colorup='k', colordown='r', alpha=1.0)
```

**ax** : Axes

an Axes instance to plot to

A sequence  
of sequences.

**quotes** : sequence of (time, open, high, low, close, ...) sequences

As long as the first 5 elements are these values, the record can be as long as you want (e.g., it may store volume).

time must be in float days format - see date2num



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 35: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
from matplotlib.finance import candlestick_ohlc
from matplotlib.dates import date2num
r=data.iloc[:15, :]
1 fig, ax = plt.subplots()
2 d=date2num(r.index.date)
3 candlestick_ohlc(ax, zip(d, r.Open, r.High, r.Low, r.Close),
                   width=0.5, colorup='g', colordown='r', alpha=1)
plt.setp(ax.get_xticklabels(), rotation=30)
ax.xaxis_date()
plt.show()
```

060  
fig=plt.figure()  
ax=plt.axes()

(Dr. Z: Can we develop a GUI for this?)

ax

1. Use `.subplots()` to return `ax`, or ...
2. Use `date2num` to convert dates to float days
3. Use `zip` to create the sequence of tuples.



Challenge: Modify “[AppFillBetween.ui](#)” (using Qt Designer) and the code “[AppFillBetweenUI.py](#)” to obtain the same GUI as shown in the following two slides.

061

References:

- <http://doc.qt.io/qt-5/qfiledialog.html>
- <https://pythonspot.com/pyqt5-file-dialog/>

See sample code

**QFileDialog.getOpenFileName**

```
fname = QFileDialog.getOpenFileName(self,'Open file',
                                     os.getcwd(),'CSV(*.csv') )
```

```
fname,_ = QFileDialog.getOpenFileName(self,'Open file',
                                      os.getcwd(),'CSV(*.csv') )
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

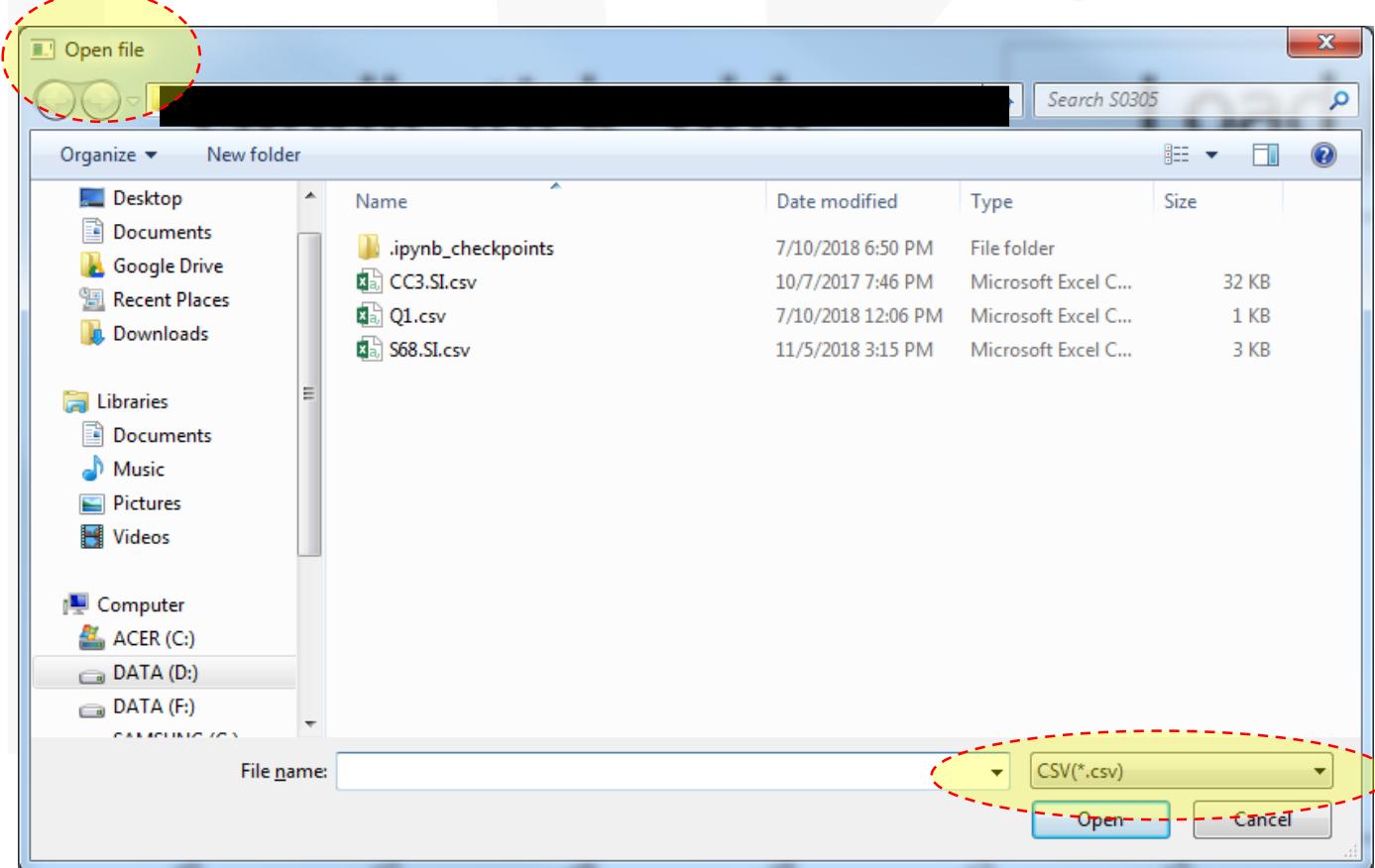


QF666  
Programming and  
Computational  
Finance

```
fname = QFileDialog.getOpenFileName(self,'Open file',  
                                     os.getcwd(),'CSV(*.csv)')
```

get current working directory

('filename/S68.SI.csv', 'CSV(\*.csv)')



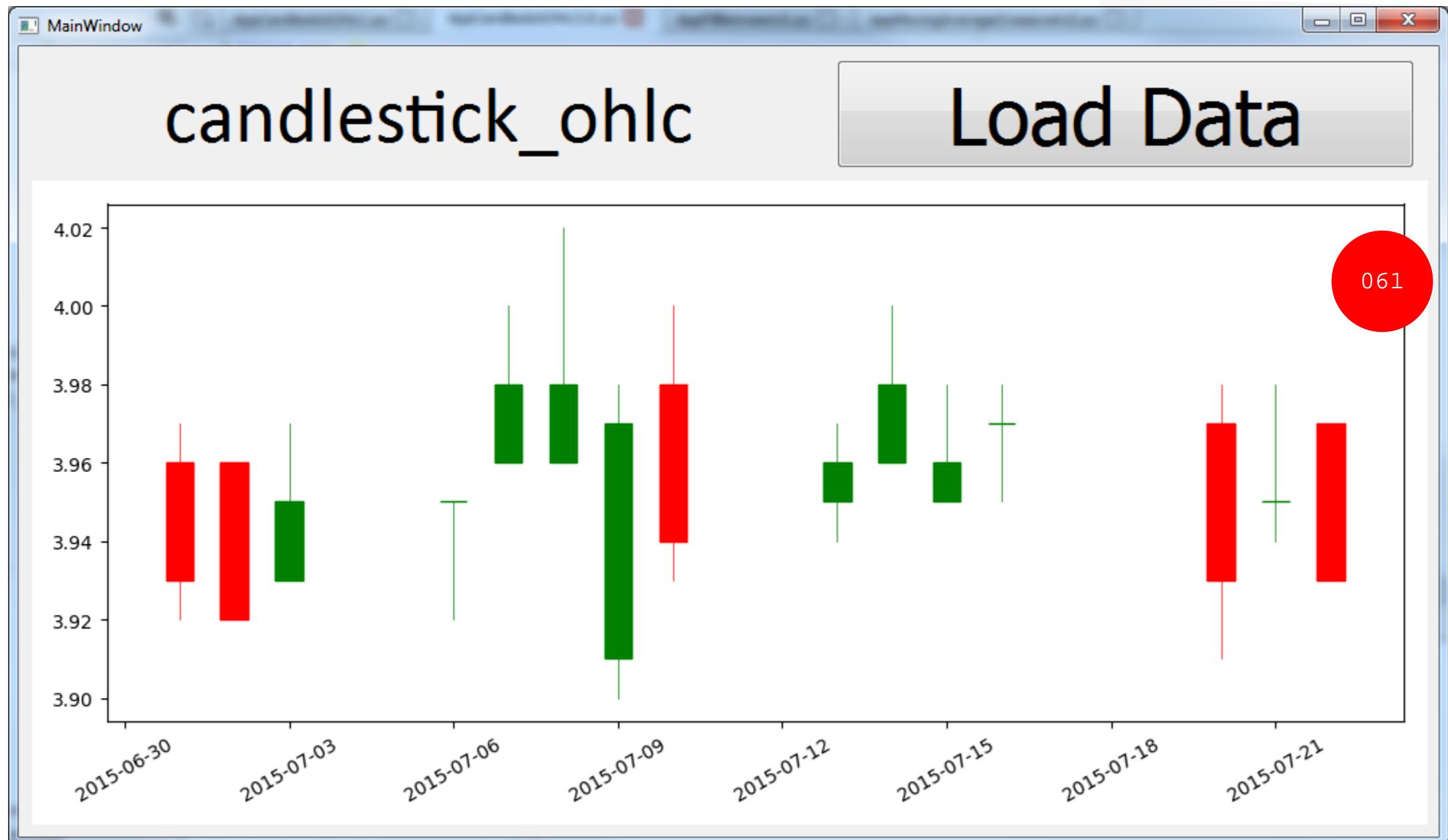
How about  
print it?

061





Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



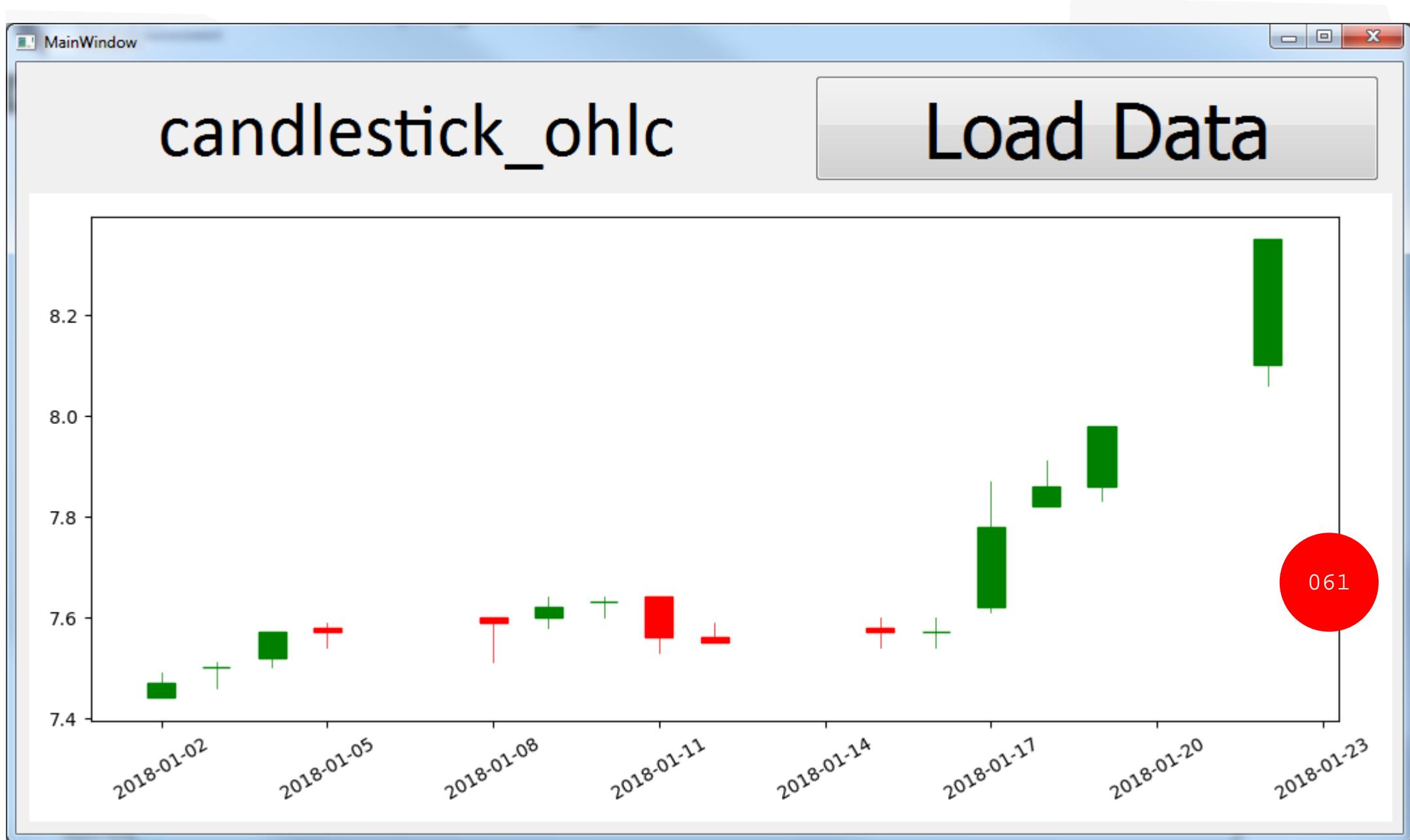
(Clicking on “Load Data” button, followed by choosing another csv file, say **S68.SI.csv**, we obtain another plot.)



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



Next, let's learn  
some basics of  
NumPy and Pandas.

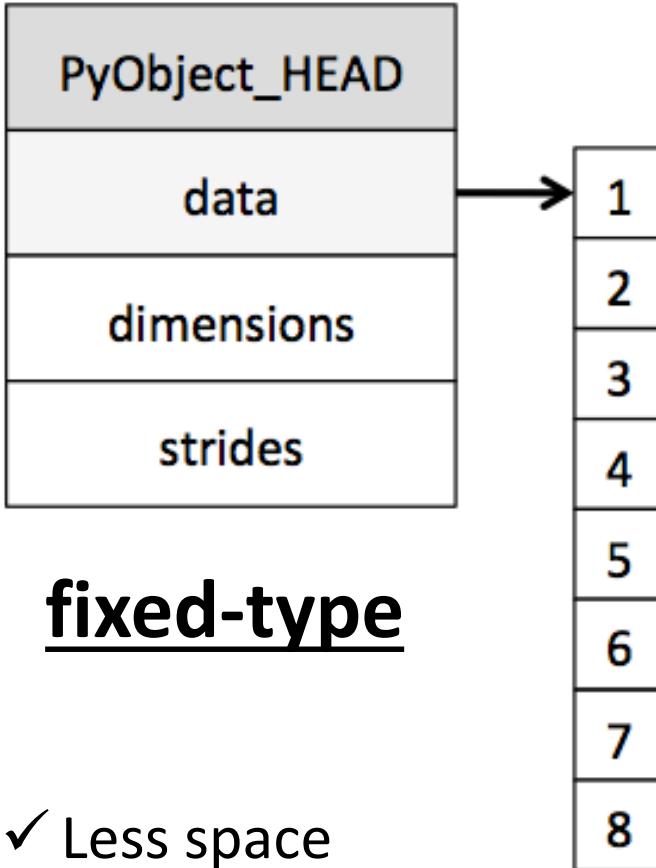


QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

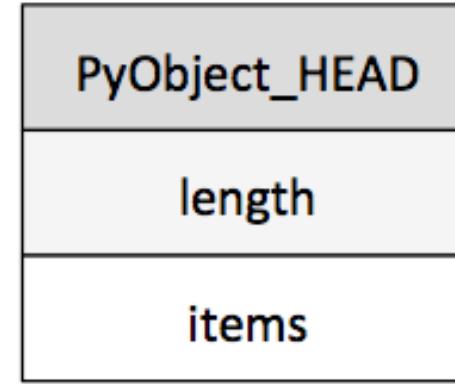
## Numpy Array



**fixed-type**

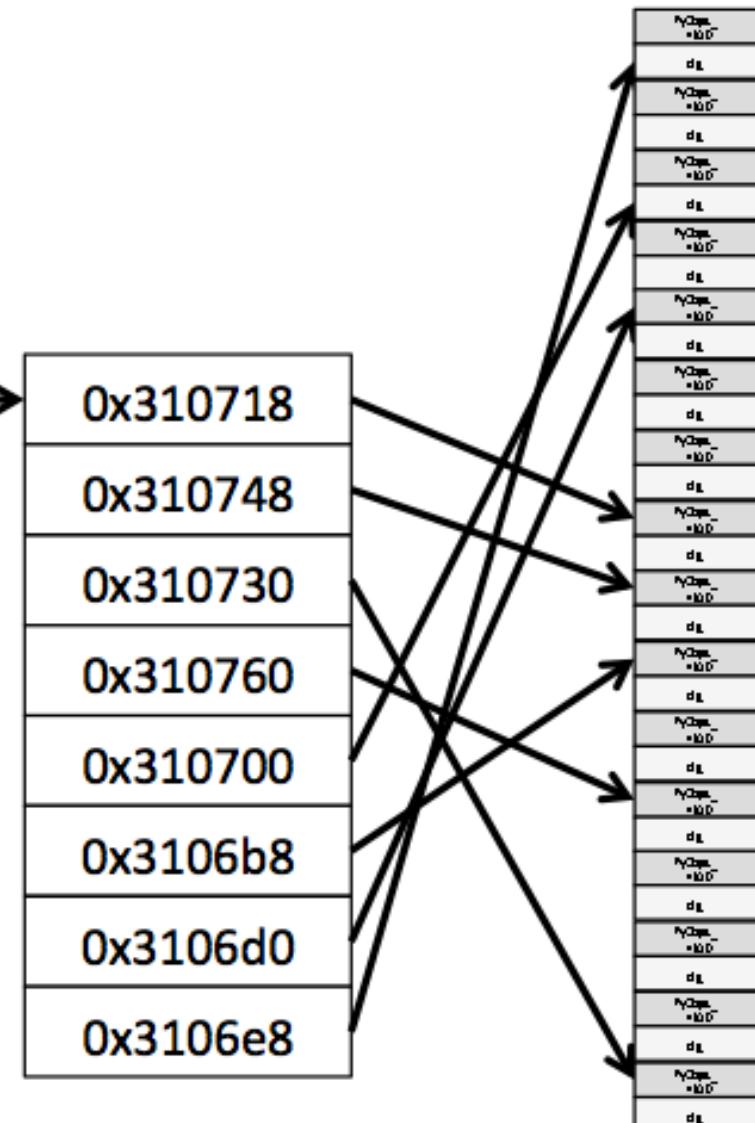
- ✓ Less space
- ✓ Faster
- ✓ Optimized functions

## Python List



**dynamic-type**

- ✓ Flexibility



# Learning Outcomes:

## ndarrays and DataFrames/Serieses

- ✓ Creation
- ✓ Indexing and Slicing
- ✓ Assignment
- ✓ Some Operations
- ✓ Some Functions



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



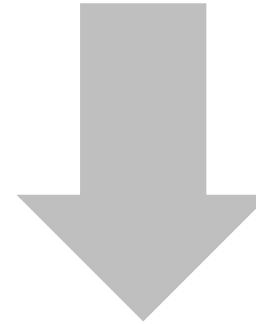
QF666

Programming and  
Computational  
Finance

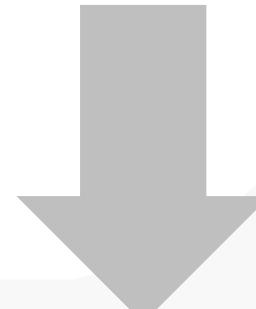


Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Python



## Numpy



## Pandas

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ybzhao>python
Python 3.6.0 |Anaconda custom (64-bit)| (default, Dec 23 2016, 11:57:41) [MSC v.1900 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> x=[1,2,3,4]
>>> x>1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'list' and 'int' Error!
>>> x+1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate list (not "int") to list Error!
>>> x**2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int' Error!
>>> x in range(5)
False
>>>
```

063

Solution:  
■ List comprehension

Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

Built-in Functions				
<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

## all(*iterable*)

Return True if all elements of the *iterable* are true (or if the iterable is empty). Equivalent to:

```
def all(iterable):
    for element in iterable:
        if not element:
            return False
    return True
```

064

```
>>> all([1,2,3])
True
>>> all([1,0,3])
False
>>> all([])
True
>>> all([[]])
False
```

## any(*iterable*)

Return True if any element of the *iterable* is true. If the iterable is empty, return False.

Equivalent to:

```
def any(iterable):
    for element in iterable:
        if element:
            return True
    return False
```

```
>>> any([1,2,3])
True
>>> any([1,0,3])
True
>>> any([])
False
>>> any([[]])
False
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 36: Use list comprehension to solve the following problem.

```
x=[1, 2, 3, 4]
```

#Q1: Compare  $x > 1$  element-by-element

#Q2: Is there any element of  $x$  greater than 1?

#Q3: Are all elements of  $x$  greater than 1?

#Q4: Compute  $x+1$  element-by-element

#Q5: Compute  $x^{**2}$  element-by-element

#Q6: Test  $x$  in  $\text{range}(5)$  element-by-element

#Q7: Is there any element of  $x$  in  $\text{range}(5)$ ?

#Q8: Are all elements of  $x$  in  $\text{range}(5)$ ?

063

065



[False, True, True, True]

True

False

[2, 3, 4, 5]

[1, 4, 9, 16]

[True, True, True, True]

True

True

Q1

Q2

Q3

Q4

Q5

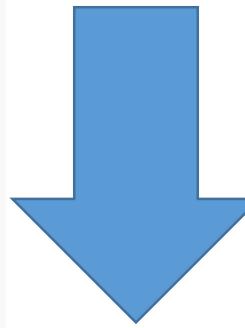
Q6

Q7

Q8



Python



Numpy



Pandas

&gt;&gt;&gt; np.isin(x, range(5))

```
C:\ Command Prompt - python
C:\Users\ybzhao>python
Python 3.6.0 |Anaconda custom (64-bit)| (default, Dec 23 2016, 11:57:41) [MSC v.1900 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x=np.array([1,2,3,4])
>>> x>1
array([False, True, True, True], dtype=bool)
>>> x+1
array([2, 3, 4, 5])
>>> x**2
array([ 1,  4,  9, 16], dtype=int32)
>>> x in range(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
>>> np.isin(x,range(5))
array([ True,  True,  True,  True], dtype=bool)
>>> -
```

Error!

Numpy has multi-dimensional arrays and vectorized computation



## In-Class Exercise 37: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import numpy as np  
x=np.array([1,2,3,4])  
print(type(x))  
print(x>1)  
print(x+1)  
print(x**2)  
print(np.isin(x,range(5)))
```

numpy.ndarray

066



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
<class 'numpy.ndarray'>  
[False True True True]  
[2 3 4 5]  
[ 1  4  9 16]  
[ True True True True]
```

multi-Dimensional  
**1D**, 2D, 3D, ..., n-D, ...

[Scipy.org](#) [Docs](#) [NumPy v1.14 Manual](#) [NumPy Reference](#) [Routines](#) [Array creation routines](#)

## numpy.array

`numpy.array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)`

Create an array.

**Parameters:** `object : array_like`

An array, any object exposing the array interface, an object whose `__array__` method returns an array, or any (nested) sequence.

`dtype : data-type, optional`

The desired data-type for the array. If not given, then the type will be determined as the minimum type required to hold the objects in the sequence. This argument can only be used to 'upcast' the array. For downcasting, use the `.astype(t)` method.

`copy : bool, optional`

If true (default), then the object is copied. Otherwise, a copy will only be made if `__array__` returns a copy, if `obj` is a nested sequence, or if a copy is needed to satisfy any of the other requirements (`dtype`, `order`, etc.).

`order : {'K', 'A', 'C', 'F'}, optional`

Specify the memory layout of the array. If `object` is not an array, the newly created array will be in C order (row major) unless 'F' is specified, in which case it will be in Fortran order (column major). If `object` is an array the following holds.

`order` no copy `copy=True`

'K' unchanged F & C order preserved, otherwise most similar order

'A' unchanged F order if input is F and not C, otherwise C order

'C' C order C order

'F' F order F order

When `copy=False` and a copy is made for other reasons, the result is the same as if `copy=True`, with some exceptions for A, see the Notes section. The default order is 'K'.  
`subok : bool, optional`

If True, then sub-classes will be passed-through, otherwise the returned array will be forced to be a base-class array (default).

`ndmin : int, optional`

Specifies the minimum number of dimensions that the resulting array should have. Ones will be pre-pended to the shape as needed to meet this requirement.

`out : ndarray`

An array object satisfying the specified requirements.

# numpy . array



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

067

## In-Class Exercise 38: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import numpy as np
x=np.array([[1,2],[3,4]])
print(type(x))
print(x>1)
print(x+1)
print(x**2)
print(np.isin(x,range(5)))
```

1	2
3	4

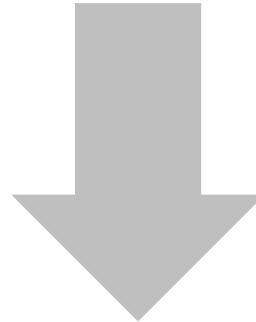
```
<class 'numpy.ndarray'>
[[False  True]
 [ True  True]]
[[2 3]
 [4 5]]
[[ 1  4]
 [ 9 16]]
[[ True  True]
 [ True  True]]
```

numpy.ndarray

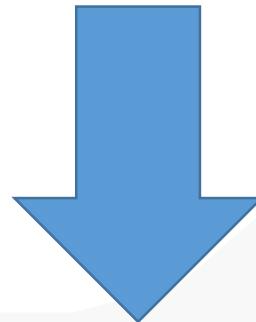
multi-Dimensional  
1D, **2D**, 3D, ..., n-D, ...



# Python



# Numpy



# Pandas

```
>>> import pandas as pd  
>>> x=pd.DataFrame([[1,2],[3,4]])
```

```
>>> x>1  
          0      1  
0  False  True  
1  True  True
```

```
>>> x+1  
          0      1  
0  2      3  
1  4      5
```

```
>>> x**2  
          0      1  
0  1      4  
1  9     16
```

```
>>> x.isin(range(5))  
          0      1  
0  True  True  
1  True  True
```

068

1	2
3	4

DataFrame  
( 2D only )

Series  
( 1D only )



# pandas.DataFrame

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html>

class `pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)`

[source]

Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure

069

**data** : *numpy ndarray (structured or homogeneous), dict, or DataFrame*

Dict can contain Series, arrays, constants, or list-like objects

**index** : *Index or array-like*

Index to use for resulting frame. Will default to np.arange(n) if no indexing information part of input data and no index provided

**columns** : *Index or array-like*

Column labels to use for resulting frame. Will default to np.arange(n) if no column labels are provided

**dtype** : *dtype, default None*

Data type to force. Only a single dtype is allowed. If None, infer

**copy** : *boolean, default False*

Copy data from inputs. Only affects DataFrame / 2d ndarray input

## Parameters:



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 39: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import pandas as pd
x=pd.DataFrame([[1,2],[3,4]], columns=list('ab'))
print(type(x))
print(x>1)
print(x+1)
print(x**2)
print(x.isin(range(5)))
```

```
<class 'pandas.core.frame.DataFrame'>
   a    b
0  False  True
1   True  True
   a    b
0    2    3
1    4    5
   a    b
0    1    4
1    9   16
   a    b
0   True  True
1   True  True
```

1	2
3	4





Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
>>> import numpy as np
>>> x=np.array([1,2,3,4]);x
array([1, 2, 3, 4])
>>> x>1
array([False,  True,  True,  True], dtype=bool)
>>> any(x>1)
True
>>> all(x>1)
False
>>> y=np.array([[1,2],[3,4]]);y
array([[1, 2],
       [3, 4]])
>>> y>1
array([[False,  True],
       [ True,  True]], dtype=bool)
>>> any(y>1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
>>> all(y>1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

any( ) and all( ) work  
on 1D arrays,

```
>>> np.any(x>1), np.all(x>1)
(True, False)
>>> np.any(y>1), np.all(y>1)
(True, False)
```

but not on 2D arrays.

# numpy.any

`numpy.any(a, axis=None, out=None, keepdims=<class 'numpy._globals._NoValue'>)`

[source]

Test whether any array element along a given axis evaluates to True.

Returns single boolean unless axis is not `None`

Parameters:

`a : array_like`  
Input array or object that can be converted to an array.

`axis : None or int or tuple of ints, optional`

Axis or axes along which a logical OR reduction is performed. The default (`axis = None`) is to perform a logical OR over all the dimensions of the input array. `axis` may be negative, in which case it counts from the last to the first axis.

New in version 1.7.0.

If this is a tuple of ints, a reduction is performed on multiple axes, instead of a single axis or all the axes as before.

axis=None

071

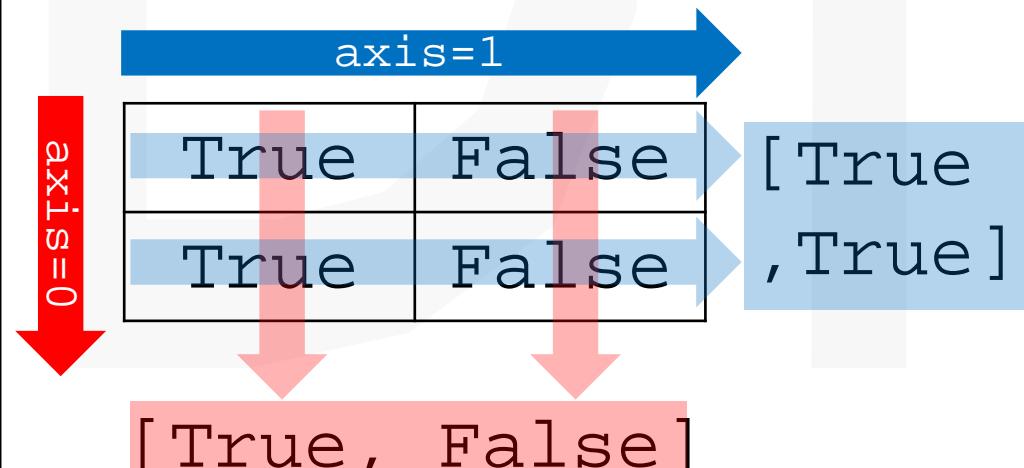
072



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



```
>>> np.any([[True, False], [True, False]])
True
>>> np.any([[True, False], [True, False]], axis=0)
array([ True, False], dtype=bool)
>>> np.any([[True, False], [True, False]], axis=1)
array([ True,  True], dtype=bool)
>>> np.any([[True, False], [True, False]], axis=-1)
array([ True,  True], dtype=bool)
```

# numpy.ndarray.any

`ndarray.` `any (axis=None, out=None, keepdims=False)`

Returns True if any of the elements of a evaluate to True.

Refer to [numpy.any](#) for full documentation.

## See also:

`numpy.any` equivalent function

```
>>> import numpy as np
>>> x=[[True,False],[True,False]]
>>> x.any()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute 'any'
>>> np.array(x).any()
True
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
>>> np.any([[True, False], [True, False]])  
True  
>>> np.any([[True, False], [True, False]], axis=0)  
array([ True, False], dtype=bool)  
>>> np.any([[True, False], [True, False]], axis=1)  
array([ True,  True], dtype=bool)  
>>> np.any([[True, False], [True, False]], axis=-1)  
array([ True,  True], dtype=bool)
```

```
>>> x=[[True, False], [True, False]]  
>>> #axis=0  
... [any([x[i][j] for i in range(2)]) for j in range(2)]  
[True, False]  
>>> #axis=1  
... [any([x[i][j] for j in range(2)]) for i in range(2)]  
[True, True]
```

Do Not Test

Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# A 3-level nested list (2-by-2-by-2):

Do Not Test

```
import numpy as np
x=[[ [True, False], [True, False] ], [[True, True], [False, False]] ]
#axis=1
print(np.any(x, axis=1))

print([[any([x[i0][i1][i2] for i1 in range(2)]) for i2 in range(2)] for i0 in range(2)]) ← for i2 in range(2) ] for i0 in range(2) )

#axis=(0,2)
print(np.any(x, axis=(0,2)))

print([any([x[i0][i1][i2] for i2 in range(2) for i0 in range(2)]) for i1 in range(2)]) ← for i1 in range(2) ]
```

```
[[ True False]
 [ True  True]]
[[True, False], [True, True]]
[ True  True]
[True, True]
```

axis = ( 0 , 2 )



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Indexing & Slicing

(for Numpy ndarray and Pandas DataFrame/Series)

Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Q: What is the output? (Indexing vs Slicing)

```
>>> x=[[0,1,2],  
...     [3,4,5],  
...     [6,7,8]]; x  
[[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```

```
>>> x[0]
```

```
[0, 1, 2]
```

```
>>> x[:1]
```

```
[[0, 1, 2]]
```

```
>>> x[1][1]
```

```
4
```

```
>>> x[:1][:1]
```

```
[[0, 1, 2]]
```

```
>>> x[1,1]
```

```
Traceback (most recent call last)
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: list indices must be integers or
```

```
>>> x[:1,:1]
```

```
Traceback (most recent call last)
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: list indices must be integers or
```

0	1	2
3	4	5
6	7	8

```
>>> import numpy as np  
>>> x=np.arange(9).reshape(3,-1);x  
array([[0, 1, 2],  
       [3, 4, 5],  
       [6, 7, 8]])
```

```
>>> x[0]
```

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

```
>>> x[:1]
```

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

```
>>> x[1][1]
```

```
4
```

```
>>> x[:1][:1]
```

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

```
>>> x[1,1]
```

```
4
```

```
>>> x[:1,:1]
```

```
array([[0]])
```

073

074

```
>>> x=[[0,1,2],[3,4,5],[6,7,8]]; x  
[[0, 1, 2], [3, 4, 5], [6, 7, 8]]  
>>> x[1,1]
```

Traceback (most recent call last):  
File "<stdin>", line 1, in <module>

Error

**TypeError: list indices must be integers or slices, not tuple**

```
>>> x[1]  
[3, 4, 5]
```

```
>>> x[1][[True, False, True]]
```

Traceback (most recent call last):  
File "<stdin>", line 1, in <module>

Error

**TypeError: list indices must be integers or slices, not list**

```
>>> x[1][[1,2,0,0]]
```

Traceback (most recent call last):  
File "<stdin>", line 1, in <module>

Error

**TypeError: list indices must be integers or slices, not list**



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



## In-Class Exercise 40: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import numpy as np
x=np.arange(0,9).reshape(3,-1)
print(x)
print(x[1])
print(x[1][1], x[1,1])      multidimensional indexing
print(x[0::2,1:])           multidimensional slicing
print(x>4)
x[x>4]=-1                  boolean indexing
print(x)
print(x[1][[1,2,0,0]])       fancying indexing
print(x[1::-1,[1,2,0,0]])    combined indexing
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[3 4 5]
4 4
[[1 2]
 [7 8]]
[[False False False]
 [False False True]
 [ True  True True]]
[[ 0  1  2]
 [ 3  4 -1]
 [-1 -1 -1]]
[ 4 -1  3  3]
[[ 4 -1  3  3]
 [ 1  2  0  0]]]
```

## Indexing & Slicing (numpy.ndarray)

<https://docs.scipy.org/doc/numpy-1.13.0/user/basics.indexing.html>



## In-Class Exercise 41: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import numpy as np  
x=np.arange(0,9).reshape(3,-1)  
print(x[1])  
print(x[[1]])
```

077

**x[ 1 ]**

```
import numpy as np  
x=np.arange(0,9).reshape(3,-1)  
print(x[1])  
print(x[[1]])
```

**VS****[ 3 4 5 ]**  
**[[ 3 4 5 ]]****x[ [ 1 ] ]**



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## Copy or View? (list)

117

```
>>> x=[0, 1, 2, 3]; x  
[0, 1, 2, 3]
```

```
>>> x[:2]=-1; x
```

```
Traceback (most recent  
File "<stdin>", line  
TypeError: can only as
```

```
>>> x[:2]=[-1,-2]; x  
[-1, -2, 2, 3]
```

```
>>> y=x[:2]; y
```

```
[-1, -2]
```

```
>>> y[0]=100; y
```

```
[100, -2]
```

```
>>> x
```

```
[-1, -2, 2, 3]
```

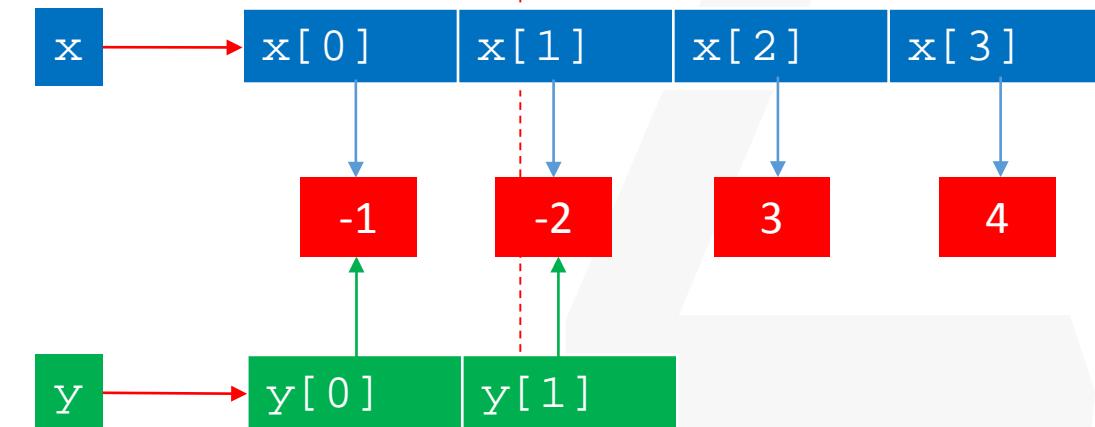
→  $x[:2][0]=100$

$y=x[:2]$

$y$  is a copy here.

Q: How to validate this?

```
>>> x=[0, 1, 2, 3]; x  
[0, 1, 2, 3]  
>>> x[:2]=-1; x  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can only assign an iterable  
>>> x[:2]=[-1,-2]; x  
[-1, -2, 2, 3]  
>>> y=x[:2]; y  
[-1, -2]  
>>> id(x), id(y)  
(45133000, 45157448)  
>>> id(x[0]), id(y[0])  
(1628486032, 1628486032)  
>>> id(x[1]), id(y[1])  
(1628486000, 1628486000)
```



(for illustration only)





# Copy or View? (1D ndarray)

```
>>> import numpy as np
>>> x=np.arange(4); x
array([0, 1, 2, 3])
>>> x[:2]=-1; x
array([-1, -1, 2, 3])
>>> x[:2]=[-1,-2]; x
array([-1, -2, 2, 3])
>>> y=x[:2]; y
array([-1, -2])
>>> y[0]=100; y
array([100, -2])
>>> x
array([100, -2, 2, 3])
```

A NEW, VALID operation.

→  $x[:2][0]=100$

$y=x[:2]$

$y$  is a view.



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
>>> import numpy as np
>>> x=np.arange(4); x
array([0, 1, 2, 3])
>>> y=x[:2];y
array([0, 1])
>>> z=x[[0,1]];z
array([0, 1])
>>> id(x), id(y), id(z)
(56356656, 39900464, 56401152)
>>> y.base is x, z.base is x
(True, False)
>>> x[0]=100
>>> y,z
(array([100, 1]), array([0, 1]))
```

numpy.ndarray is too  
complicated to use `id()`  
to tell us anything.

080

✓ `numpy.ndarray.base`

Slicing of ndarray  
returns a view;  
fancy indexing  
returns a copy.

`x[ :2][0]=100`



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

(Dr. Z: This is called *chained indexing*. It is difficult to tell whether the assignment happens to a copy or a view of `x`. It depends on the implementation of `__getitem__` and `__setitem__`. So, we won't explore further what will happen on it.)

# DataFrame

- ✓ Selection via `[ ]`
- ✓ Selection by Position (`.iloc`)
- ✓ Selection by Label (`.loc`)
- ✓ Boolean Indexing



## In-Class Exercise 42: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"



```
import numpy as np
import pandas as pd
x=pd.DataFrame(np.arange(16).reshape(4,4))
print(x, type(x))
print(x[1], type(x[1]))
print(x[[1]], type(x[[1]]))
print(x[:1], type(x[:1]))
print(x[1][1], type(x[1][1]))
x[x>9]=-1
print(x)
```

1

2

3

4

5

6



```
0   0   1   2   3  
0   0   1   2   3  
1   4   5   6   7  
2   8   9   10  11  
3   12  13  14  15 <class 'pandas.core.frame.DataFrame'>
```

1

```
0   1  
1   5  
2   9  
3   13  
Name: 1, dtype: int32 <class 'pandas.core.series.Series'>
```

2

```
1  
0   1  
1   5  
2   9  
3   13 <class 'pandas.core.frame.DataFrame'>
```

3

```
0   1   2   3  
0   0   1   2   3 <class 'pandas.core.frame.DataFrame'>  
5 <class 'numpy.int32'>
```

4

```
0   1   2   3  
0   0   1   2   3  
1   4   5   6   7  
2   8   9   -1  -1  
3  -1  -1  -1  -1
```

5

# DataFrame

Indexing:  
column  
+  
Series

081

Fancy indexing:  
column  
+  
DataFrame

Slicing: row + DataFrame

[ ]



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# ndarray



# DataFrame

```
>>> import numpy as np
>>> x=np.arange(9).reshape(3,3); x
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
>>> x[1,1]
4
```

081

```
>>> import pandas as pd
>>> import numpy as np
>>> x=pd.DataFrame(np.arange(9).reshape(3,3));x
   0  1  2
0  0  1  2
1  3  4  5
2  6  7  8
>>> x[1,1]
Traceback (most recent call last):
  File "pandas\_libs\index.pyx", line 154, in pa
dex.c:5126>
  File "pandas\_libs\hashtable_class_helper.pxi"
```

(Dr. Z: DataFrames use .loc and .iloc for multidimensional indexing.)

# .iloc

# integer position based

- .iloc is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array. .iloc will raise IndexError if a requested indexer is out-of-bounds, except slice indexers which allow out-of-bounds indexing. (this conforms with python/numpy slice semantics). Allowed inputs are:

- An integer e.g. 5 indexing
- A list or array of integers [4, 3, 0] fancy indexing
- A slice object with ints 1:7 slicing
- A boolean array Boolean indexing
- A callable function with one argument (the calling Series, DataFrame or Panel) and that returns valid output for indexing (one of the above)



`df.iloc[1]`  $\leftrightarrow$  `df.iloc[1, : ]`



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## In-Class Exercise 43: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"



```
import numpy as np
import pandas as pd
x=pd.DataFrame(np.arange(16).reshape(4,4),
               index=[2,1,4,3],columns=list('bcad'))
print(x)
print(x.iloc[2], type(x.iloc[2]))
print(x.iloc[[2]], type(x.iloc[[2]]))
print(x.iloc[:2], type(x.iloc[:2]))
print(x.iloc[[True,False,False,True]],
      type(x.iloc[[True,False,False,True]]))
print(x.iloc[1,1])
```

1

2

3

4

5

6



	b	c	a	d
2	0	1	2	3
1	4	5	6	7
4	8	9	10	11
3	12	13	14	15

# iloc

082

b	8
c	9
a	10
d	11

row 2

```
Name: 4, dtype: int32 <class 'pandas.core.series.Series'>
```

	b	c	a	d
--	---	---	---	---

row 3

```
4 8 9 10 11 <class 'pandas.core.frame.DataFrame'>
```

b	c	a	d
---	---	---	---

row 4

2	0	1	2	3
---	---	---	---	---

```
1 4 5 6 7 <class 'pandas.core.frame.DataFrame'>
```

b	c	a	d
---	---	---	---

row 5

2	0	1	2	3
---	---	---	---	---

```
3 12 13 14 15 <class 'pandas.core.frame.DataFrame'>
```

5
---

row 6



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# .loc

083

- .loc is primarily label based, but may also be used with a boolean array. .loc will raise KeyError when the items are not found. Allowed inputs are:
  - A single label, e.g. 5 or 'a', (note that 5 is interpreted as a *label* of the index. This use is **not** an integer position along the index)
  - A list or array of labels ['a', 'b', 'c']
  - A slice object with labels 'a': 'f' (note that contrary to usual python slices, **both** the start and the stop are included, when present in the index! - also see [Slicing with labels](#))
  - A boolean array
  - A callable function with one argument (the calling Series, DataFrame or Panel) and that returns valid output for indexing (one of the above)



## In-Class Exercise 44: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"



```
import numpy as np
import pandas as pd
x=pd.DataFrame(np.arange(16).reshape(4,4),
               index=[2,1,4,3],columns=list('bcad'))
print(x)
print(x.loc[2], type(x.loc[2]))
print(x.loc[[2]], type(x.loc[[2]]))
print(x.loc[:2], type(x.loc[:2]))
print(x.loc[[True,False,False,True]],
      type(x.loc[[True,False,False,True]]))
print(x.loc[1,'a'])
```



	b	c	a	d
2	0	1	2	3
1	4	5	6	7
4	8	9	10	11
3	12	13	14	15

b 0  
c 1  
a 2  
d 3

Name: 2, dtype: int32 <class 'pandas.core.series.Series'>

	b	c	a	d
2	0	1	2	3

<class 'pandas.core.frame.DataFrame'>

	b	c	a	d
2	0	1	2	3

<class 'pandas.core.frame.DataFrame'>



	b	c	a	d
2	0	1	2	3
3	12	13	14	15

<class 'pandas.core.frame.DataFrame'>

6

1

2

3

4

5

6





SciPy.org

Sponsored By  
ENTHOUGHT

Scipy.org

Docs

NumPy v1.13 Manual

NumPy Reference

Routines

Array creation routines

## numpy.arange

`numpy. arange ([start, ]stop, [step, ]dtype=None)`

Return evenly spaced values within a given interval.

Values are generated within the half-open interval `[start, stop]` (in other words, the interval including `start` but excluding `stop`). For integer arguments the function is equivalent to the Python built-in `range` function, but returns an ndarray rather than a list.

When using a non-integer step, such as 0.1, the results will often not be consistent. It is better to use `linspace` for these cases.

```
>>> import numpy as np
>>> np.arange(1,2,0.2)
array([ 1. ,  1.2,  1.4,  1.6,  1.8])
>>> np.linspace(1,2,6)
array([ 1. ,  1.2,  1.4,  1.6,  1.8,  2. ])
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# numpy.linspace

`numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)`

Return evenly spaced numbers over a specified interval.

Returns `num` evenly spaced samples, calculated over the interval `[start, stop]`.

The endpoint of the interval can optionally be excluded.

084

# numpy.ndarray.reshape

`ndarray.reshape(shape, order='C')`

Row-major order

Returns an array containing the same data with a new shape.

Refer to `numpy.reshape` for full documentation.

085



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



## In-Class Exercise 45: Type (or copy and paste) the following lines of code followed by pressing "Shift-Enter"

```
import numpy as np
print(np.arange(3)+5)
print(np.ones((3,3))+np.arange(3))
print(np.arange(3).reshape((3,1))+np.arange(3))
```

<https://jakevdp.github.io/PythonDataScienceHandbook/02.05-computation-on-arrays-broadcasting.html>

```
import numpy as np
print(np.arange(3)+5)
print(np.ones((3,3))+np.arange(3))
print(np.arange(3).reshape((3,1))+np.arange(3))
```

```
[5 6 7]
[[ 1.  2.  3.]
 [ 1.  2.  3.]
 [ 1.  2.  3.]]
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

# Stretching and Broadcasting (ndarray)

086

`np.arange(3)+5`

$$\begin{matrix} \begin{matrix} 0 & 1 & 2 \end{matrix} \\ + \end{matrix} \begin{matrix} \begin{matrix} 5 & 5 & 5 \end{matrix} \\ = \end{matrix} \begin{matrix} \begin{matrix} 5 & 6 & 7 \end{matrix} \end{matrix}$$

`np.ones((3,3))+np.arange(3)`

$$\begin{matrix} \begin{matrix} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \end{matrix} \\ + \end{matrix} \begin{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{matrix} \end{matrix} \\ = \end{matrix} \begin{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{matrix} \end{matrix} \end{matrix}$$

`np.arange(3).reshape((3,1))+np.arange(3)`

$$\begin{matrix} \begin{matrix} \begin{matrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{matrix} \end{matrix} \\ + \end{matrix} \begin{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{matrix} \end{matrix} \\ = \end{matrix} \begin{matrix} \begin{matrix} \begin{matrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{matrix} \end{matrix} \end{matrix}$$



# numpy.ones

```
numpy. ones (shape, dtype=None, order='C')
```

Return a new array of given shape and type, filled with ones.

```
>>> import numpy as np  
>>> np.ones((3,3))  
array([[ 1.,  1.,  1.],  
       [ 1.,  1.,  1.],  
       [ 1.,  1.,  1.]])
```

Default: float type

dtype=int  $\Rightarrow$  int type

```
>>> np.ones(3,3)  
Traceback (most recent call last):  
  File "<stdin>", line 1,  
    File "C:\Continuum\Anaconda3\lib\site-packages\numpy\__init__.py", line 143, in ones  
      a = empty(shape, dtype=dtype, order=order)  
TypeError: data type not understood
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
>>> import pandas as pd
>>> x=pd.DataFrame([1,2,3]); x
   0
0  1
1  2
2  3
>>> y=pd.DataFrame([[1,2,3]]); y
   0  1  2
0  1  2  3
>>> x.values+y.values
array([[2, 3, 4],
       [3, 4, 5],
       [4, 5, 6]], dtype=int64)
>>> x+y
   0  1  2
0  2.0 NaN NaN
1  NaN NaN NaN
2  NaN NaN NaN
>>> pd.DataFrame(x.values+y.values)
   0  1  2
0  2  3  4
1  3  4  5
2  4  5  6
```

It seems that  
pandas.DataFrame  
does not have  
ndarray's  
“Stretching and  
Broadcasting”.

So what? 😊

088



```
>>> import pandas as pd
>>> x=pd.Series(range(5)); x
0    0
1    1
2    2
3    3
4    4
dtype: int32
>>> x[:3]
0    0
1    1
2    2
dtype: int32
>>> x[2:]
2    2
3    3
4    4
dtype: int32
>>> x[:3]-x[2:]
0    NaN
1    NaN
2    0.0
3    NaN
4    NaN
dtype: float64
```

089

```
>>> x[:3]-x[2:].values
0    -2
1    -2
2    -2
dtype: int32
```

Subtraction of Two Series  
with different labels.

## Algorithm (Binomial Tree):

Given  $S, K, r, q, t, T, \sigma$  and  $N$  (1 day=1/365 years)

$$\tau = T - t$$

090

1.  $\Delta t = \frac{T-t}{N}, u = e^{\sigma\sqrt{\Delta t}}, d = \frac{1}{u}, p = \frac{e^{(r-q)\Delta t} - d}{u - d}$
2.  $f_{N,j} = \max(0, Su^j d^{N-j} - K)$  (Call),  $f_{N,j} = \max(0, K - Su^j d^{N-j})$  (Put),  
for  $j = 0, 1, \dots, N$
3.  $f_{i,j} = e^{-r\Delta t} [p \cdot f_{i+1,j+1} + (1-p) \cdot f_{i+1,j}]$  for  $i = N-1, N-2, \dots, 1, 0; j = 0, 1, \dots, i$
4. Option (call/put) price:  $V = f_{0,0}$
5. Option Greeks (\*Delta and \*Gamma):

$$\Delta = \frac{f_{1,1} - f_{1,0}}{S \cdot u - S \cdot d}, \Gamma = \frac{d(f_{1,2} - f_{1,1}) + u(f_{2,1} - f_{2,0})}{S^2(u - d)^2}$$

(optional)



For example:  $S = 50$ ,  $K = 50$ ,  $r = 0.04$ ,  $q = 0.01$ ,  $\tau = 183/365$ ,  $\sigma = 0.4$ , and  $N = 3$ .

$$1. \quad \Delta t = \frac{r}{N}, u = e^{\sigma\sqrt{\Delta t}}, d = \frac{1}{u}, p = \frac{e^{(r-q)\Delta t} - d}{u - d}$$

2. ( $i = 3$ )

- ( $j = 0$ )  $f_{3,0} = \max(0, Su^0 d^{3-0} - K)$  (Call),  $f_{3,0} = \max(0, K - Su^0 d^{3-0})$  (Put)
- ( $j = 1$ )  $f_{3,1} = \max(0, Su^1 d^{3-1} - K)$  (Call),  $f_{3,1} = \max(0, K - Su^1 d^{3-1})$  (Put)
- ( $j = 2$ )  $f_{3,2} = \max(0, Su^2 d^{3-2} - K)$  (Call),  $f_{3,2} = \max(0, K - Su^2 d^{3-2})$  (Put)
- ( $j = 3$ )  $f_{3,3} = \max(0, Su^3 d^{3-3} - K)$  (Call),  $f_{3,3} = \max(0, K - Su^3 d^{3-3})$  (Put)

3.

- ( $i = 2$ )
  - ( $j = 0$ )  $f_{2,0} = e^{-r\Delta t} [p \cdot f_{3,1} + (1-p) \cdot f_{3,0}]$
  - ( $j = 1$ )  $f_{2,1} = e^{-r\Delta t} [p \cdot f_{3,2} + (1-p) \cdot f_{3,1}]$
  - ( $j = 2$ )  $f_{2,2} = e^{-r\Delta t} [p \cdot f_{3,3} + (1-p) \cdot f_{3,2}]$
- ( $i = 1$ )
  - ( $j = 0$ )  $f_{1,0} = e^{-r\Delta t} [p \cdot f_{2,1} + (1-p) \cdot f_{2,0}]$
  - ( $j = 1$ )  $f_{1,1} = e^{-r\Delta t} [p \cdot f_{2,2} + (1-p) \cdot f_{2,1}]$
- ( $i = 0$ )
  - ( $j = 0$ )  $f_{0,0} = e^{-r\Delta t} [p \cdot f_{1,1} + (1-p) \cdot f_{1,0}]$

- $\tau \rightarrow \text{tau}$
- $\Delta t \rightarrow \text{deltaT}$
- $f_{i,j} \rightarrow f[i][j]$

```
>>> f=[[0.0],
...     [0.0, 0.0],
...     [0.0, 0.0, 0.0],
...     [0.0, 0.0, 0.0, 0.0]]
```



```
>>> f
```

```
[[0.0], [0.0, 0.0], [0.0, 0.0, 0.0],
```

```
>>> f[3][0]=1.1
```

```
>>> f
```

```
[[0.0], [0.0, 0.0], [0.0, 0.0, 0.0],
```

```
[1.1, 0.0, 0.0, 0.0]]
```

f[0][0]

f[1][0]

f[1][1]

f[2][0]

f[2][1]

f[2][2]

f[3][0]

f[3][1]

f[3][2]

f[3][3]

$f_{i,j} \rightarrow f[i][j]$

$f_{3,0} = 1.1 \rightarrow f[3][0] = 1.1$



## Three Implementations

1. Nested list with for loop
2. Nested list with list comprehension
3. Numpy ndarray



QF666

Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

Given  $N$ , ...

$f_{N,j}, j = 0, 1, \dots, N$

$N = 3: f_{3,0}, f_{3,1}, f_{3,2}, f_{3,3}$

$f_{i,j}, i = N - 1, \dots, 0; j = 0, 1, \dots, i$

$N = 3: f_{2,0}, f_{2,1}, f_{2,2}; f_{1,0}, f_{1,1}; f_{0,0}$

For example,  $N = 3$

$f_{0,0}$			
$f_{1,0}$	$f_{1,1}$		
$f_{2,0}$	$f_{2,1}$	$f_{2,2}$	
$f_{3,0}$	$f_{3,1}$	$f_{3,2}$	$f_{3,3}$

$f_{i,j} \rightarrow f[i][j]$

list, ndarray

$f_{i,j} \rightarrow f[i, j]$

ndarray



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

$f_{0,0}$			
$f_{1,0}$	$f_{1,1}$		
$f_{2,0}$	$f_{2,1}$	$f_{2,2}$	
$f_{3,0}$	$f_{3,1}$	$f_{3,2}$	$f_{3,3}$

N=3

```
f=[[0.0 for j in range(i+1)]  
   for i in range(N+1)]
```

 $f_{i,j} \rightarrow f[i][j]$ 

(list)

```
>>> N=3  
>>> f=[[0.0 for j in range(i+1)]  
...       for i in range(N+1)]  
>>> f  
[[0.0], [0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0]]
```



Dr.Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

$f_{0,0}$			
$f_{1,0}$	$f_{1,1}$		
$f_{2,0}$	$f_{2,1}$	$f_{2,2}$	
$f_{3,0}$	$f_{3,1}$	$f_{3,2}$	$f_{3,3}$

(Numpy ndarray)

$f_{i,j} \rightarrow f[i, j]$

$f_{i,j} \rightarrow f[i][j]$

```
import numpy as np  
N=3  
f=np.zeros((N+1,N+1))
```

```
>>> import numpy as np  
>>> N=3  
>>> f=np.zeros((N+1,N+1))  
>>> f  
array([[ 0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.]])
```



2.  $f_{N,j} = \max(0, Su^j d^{N-j} - K)$  (Call),  $f_{N,j} = \max(0, K - Su^j d^{N-j})$  (Put),  
for  $j = 0, 1, \dots, N$

$f_{i,j} \rightarrow f[i][j]$

for  $j$  in range( $N+1$ ):

$fc[N][j] = \max(0, S * (u**j) * (d** (N-j)) - K)$

$fp[N][j] = \max(0, K - S * (u**j) * (d** (N-j)))$



3.  $f_{i,j} = e^{-r\Delta t} [p \cdot f_{i+1,j+1} + (1-p) \cdot f_{i+1,j}]$  for  $i = N-1, N-2, \dots, 1, 0; j = 0, 1, \dots, i$

$f_{i,j} \rightarrow f[i][j]$

```
p1=1-p  
ert=math.exp(-r*deltaT)  
  
for i in range(N-1,0-1,-1):  
    for j in range(i+1):  
        fc[i][j]=ert*(p*fc[i+1][j+1]+p1*fc[i+1][j])  
        fp[i][j]=ert*(p*fp[i+1][j+1]+p1*fp[i+1][j])
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
# nested list with for loop
import math
def V_binomial_tree_py(S,K,r,q,tau,sigma,N=100):
    # 1.
    deltaT=tau/N
    u=math.exp(sigma*math.sqrt(deltaT))
    d=1/u
    p=(math.exp((r-q)*deltaT)-d)/(u-d)
    # 2.
    fc=[[0.0 for j in range(i+1)] for i in range(N+1)]
    fp=[[0.0 for j in range(i+1)] for i in range(N+1)]

    for j in range(N+1):
        fc[N][j]=max(0, S*(u**j)*(d**(N-j))-K)
        fp[N][j]=max(0, K-S*(u**j)*(d**(N-j)))

    # 3.
    p1=1-p
    ert=math.exp(-r*deltaT)

    for i in range(N-1,0,-1):
        for j in range(i+1):
            fc[i][j]=ert*(p*fc[i+1][j+1]+p1*fc[i+1][j])
            fp[i][j]=ert*(p*fp[i+1][j+1]+p1*fp[i+1][j])

    # 4.
    c=fc[0][0]
    p=fp[0][0]
    return (c, p)

if __name__=='__main__':
    S=50.0; K=50.0; tau=183/365;
    sigma=0.4; r=0.04; q=0.01
    print('Call: {0[0]}, Put: {0[1]}\'.format(
        V_binomial_tree_py(S,K,r,q,tau,sigma)))
```

2 & 3



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



```
# nested list with list comprehension
import math
def V_binomial_tree_py2(S,K,r,q,tau,sigma,N=100):
    # 1.
    deltaT=tau/N
    u=math.exp(sigma*math.sqrt(deltaT))
    d=1/u
    p=(math.exp((r-q)*deltaT)-d)/(u-d)
    # 2.
    fc=[0]*(N+1)
    fp=[0]*(N+1)

    fc[N]=[max(0, S*(u**j)*(d**(N-j))-K) for j in range(N+1)]
    fp[N]=[max(0, K-S*(u**j)*(d**(N-j))) for j in range(N+1)]

    # 3.
    p1=1-p
    ert=math.exp(-r*deltaT)

    for i in range(N-1,0,-1):
        fc[i]=[ert*(p*fc[i+1][j+1]+p1*fc[i+1][j]) for j in range(i+1)]
        fp[i]=[ert*(p*fp[i+1][j+1]+p1*fp[i+1][j]) for j in range(i+1)]

    # 4.
    c=fc[0][0]
    p=fp[0][0]
    return (c, p)

if __name__=='__main__':
    S=50.0; K=50.0; tau=183/365;
    sigma=0.4; r=0.04; q=0.01
    print('Call: {}, Put: {}'.format(
        V_binomial_tree_py2(S,K,r,q,tau,sigma)))
```

(using list comprehension)

2.  $f_{N,j} = \max(0, Su^j d^{N-j} - K)$  (Call),  $f_{N,j} = \max(0, K - Su^j d^{N-j})$  (Put),  
for  $j = 0, 1, \dots, N$

(ndarray)

$f_{i,j} \rightarrow f[i, j]$

(Dr. Z: Remember ndarrays  
allow **vectorized operations?**)

for  $j$  in range( $N+1$ ):

$fc[N, j] = \max(0, S * (u**j) * (d** (N-j)) - K)$

$fp[N, j] = \max(0, K - S * (u**j) * (d** (N-j)))$



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
for j in range(N+1):  
    fc[N, j]=max( 0 , S*( u**j ) * ( d** ( N-j ) ) -K )  
    fp[N, j]=max( 0 , K-S*( u**j ) * ( d** ( N-j ) ) )
```

```
j=np.arange( 0 ,N+1 ,1 );
```

```
Ss=S*( u**j ) * ( d** ( N-j ) )
```

```
fc[N, : ]=np.maximum( 0 , Ss-K )
```

```
fp[N, : ]=np.maximum( 0 , K-Ss )
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

## numpy.maximum

```
numpy. maximum (x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, extobj]) = <ufunc 'maximum'>
```

Element-wise maximum of array elements.

Compare two arrays and returns a new array containing the element-wise maxima. If one of the elements being compared is a NaN, then that element is returned. If both elements are NaNs then the first is returned. The latter distinction is important for complex NaNs, which are defined as at least one of the real or imaginary parts being a NaN. The net effect is that NaNs are propagated.

092

```
>>> import numpy as np
>>> np.maximum([2,3,4],[1,5,2])
array([2, 5, 4])
>>> np.maximum(np.eye(2),[0.5,2]) # broadcasting
array([[ 1.,  2.],
       [ 0.5,  2.]])
>>> np.maximum([np.nan, 0, np.nan], [0, np.nan, np.nan])
__main__:1: RuntimeWarning: invalid value encountered in maximum
array([ nan,  nan,  nan])
>>> np.maximum(np.Inf, 1)
inf
```



3.  $f_{i,j} = e^{-r\Delta t} [p \cdot f_{i+1,j+1} + (1-p) \cdot f_{i+1,j}]$  for  $i = N-1, N-2, \dots, 1, 0; j = 0, 1, \dots, i$

$f_{i,j} \rightarrow f[i, j]$

$p1=1-p$   
 $ert=math.exp(-r*deltaT)$

```
for i in range(N-1, 0-1, -1):  
    for j in range(i+1):  
        fc[i, j]=ert*(p*fc[i+1, j+1]+p1*fc[i+1, j])  
        fp[i, j]=ert*(p*fp[i+1, j+1]+p1*fp[i+1, j])
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
for i in range(N-1,0-1,-1):  
    for j in range(i+1):  
        fc[i,j]=ert*(p*fc[i+1,j+1]+p1*fc[i+1,j])  
        fp[i,j]=ert*(p*fp[i+1,j+1]+p1*fp[i+1,j])
```

(using vectorized operations)

```
for i in range(N-1,0-1,-1):  
    fc[i,0:i+1]=ert*(p*fc[i+1,0+1:i+1+1]+p1*fc[i+1,0:i+1])  
    fp[i,0:i+1]=ert*(p*fp[i+1,0+1:i+1+1]+p1*fp[i+1,0:i+1])
```



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



QF666  
Programming and  
Computational  
Finance

```
# Numpy ndarray
import numpy as np

def V_binomial_tree_np(S,K,r,q,tau,sigma,N=100):
    # 1.
    deltaT=tau/N
    u=np.exp(sigma*np.sqrt(deltaT))
    d=1/u
    p=(np.exp((r-q)*deltaT)-d)/(u-d)
    # 2.
    fc=np.zeros((N+1,N+1))
    fp=np.zeros((N+1,N+1))
    j=np.arange(0,N+1,1);
    Ss=S*(u**j)*(d***(N-j))
    fc[N,:]=np.maximum(0, Ss-K)
    fp[N,:]=np.maximum(0, K-Ss)
    # 3.
    p1=1-p
    ert=np.exp(-r*deltaT)
    for i in range(N-1,0,-1):
        fc[i,0:i+1]=ert*(p*fc[i+1,0+1:i+1+1]+p1*fc[i+1,0:i+1])
        fp[i,0:i+1]=ert*(p*fp[i+1,0+1:i+1+1]+p1*fp[i+1,0:i+1])
    # 4.
    c=fc[0,0]
    p=fp[0,0]
    return (c, p)

if __name__=='__main__':
    S=50.0; K=50.0; tau=183/365;
    sigma=0.4; r=0.04; q=0.01
    print('Call: {0[0]}, Put: {0[1]}\'.format(
        V_binomial_tree_np(S,K,r,q,tau,sigma)))
```



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# Numpy <https://docs.scipy.org/doc/numpy/reference/routines.html>

- `numpy.array`
- `numpy.arange`
- `numpy.reshape` or `numpy.ndarray.reshape`
- `numpy.isin`
- `numpy.linspace`
- `numpy.ones`, `numpy.zeros`, `numpy.empty`
- **`numpy.exp`, `numpy.sqrt`**
- **`numpy.mean`, `numpy.std`, `numpy.sum`, `numpy.cumsum`**
- `numpy.maximum`
- `numpy.random`
- `numpy.append`, `numpy.concatenate`
- `... (numpy.linalg) ...`



# numpy.random.randn

`numpy.random.randn(d0, d1, ..., dn)`

Return a sample (or samples) from the "standard normal" distribution.

If positive, `int_like` or `int-convertible` arguments are provided, `randn` generates an array of shape  $(d_0, d_1, \dots, d_n)$ , filled with random floats sampled from a univariate "normal" (Gaussian) distribution of mean 0 and variance 1 (if any of the  $d_i$  are floats, they are first converted to integers by truncation). A single float randomly sampled from the distribution is returned if no argument is provided.

This is a convenience function. If you want an interface that takes a tuple as the first argument, use

`numpy.random.standard_normal` instead.

**Parameters:** `d0, d1, ..., dn : int, optional`

The dimensions of the returned array, should be all positive. If no argument is given a single Python float is returned.

**Returns:** `Z : ndarray or float`

A  $(d_0, d_1, \dots, d_n)$ -shaped array of floating-point samples from the standard normal distribution, or a single such float if no parameters were supplied.

# np.random.randn(3, 4)



## numpy.random.standard\_normal

`numpy.random.standard_normal(size=None)`

Draw samples from a standard Normal distribution (mean=0, stdev=1).

Parameters: `size : int or tuple of ints, optional`

Output shape. If the given shape is, e.g., `(m, n, k)`, then `m * n * k` samples are drawn. Default is `None`, in which case a single value is returned.

Returns: `out : float or ndarray`

Drawn samples.

### Examples

```
>>> s = np.random.standard_normal(8000) >>>
>>> s
array([ 0.6888893 ,  0.78096262, -0.89086505, ...,  0.49876311, #random
       -0.38672696, -0.4685006 ]) #random
>>> s.shape
(8000,)
>>> s = np.random.standard_normal(size=(3, 4, 2))
>>> s.shape
(3, 4, 2)
```

094

`np.random.standard_normal( ( 3 , 4 ) )`



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# numpy.random.random

## numpy.random.random(size=None)

Return random floats in the half-open interval [0.0, 1.0).

Results are from the "continuous uniform" distribution over the stated interval. To sample  $Unif[a, b], b > a$  multiply the output of `random_sample` by  $(b-a)$  and add  $a$ :

```
(b - a) * random_sample() + a
```

**Parameters:** `size : int or tuple of ints, optional`

Output shape. If the given shape is, e.g., `(m, n, k)`, then  $m * n * k$  samples are drawn. Default is `None`, in which case a single value is returned.

**Returns:** `out : float or ndarray of floats`

Array of random floats of shape `size` (unless `size=None`, in which case a single float is returned).

`np.random.random( (3, 4) )`



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

```
Anaconda Prompt - python
Python 3.6.4 |Anaconda custom (64-bit)| (default, Jan 16 2018, 10:22:32) [MSC v.
1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.random.randn(3,4)
array([[-1.02131365, -1.29791614,  0.55330455,  0.98542153],
       [ 0.16692936,  1.0607157 , -0.1153914 , -0.5761791 ],
       [ 0.58843827, -0.64756756,  0.00655177, -0.08050467]])
>>> np.random.random((3,4))
array([[0.72118706, 0.07525318, 0.32270341, 0.77687445],
       [0.13139268, 0.9516377 , 0.65025612, 0.47259156],
       [0.9623561 , 0.01804862, 0.17095629, 0.99188202]])
>>> np.random.randn((3,4))
xxxxxxxx
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "mtrand.pyx", line 1420, in mtrand.RandomState.randn
  File "mtrand.pyx", line 1550, in mtrand.RandomState.standard_normal
  File "mtrand.pyx", line 167, in mtrand.cont0_array
TypeError: 'tuple' object cannot be interpreted as an integer
>>> np.random.random(3,4)
xxxxxxxx
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "mtrand.pyx", line 819, in mtrand.RandomState.random_sample
TypeError: random_sample() takes at most 1 positional argument (2 given)
>>>
```

# numpy.append

`numpy.append(arr, values, axis=None)`

[source]

Append values to the end of an array.

**Parameters:** `arr : array_like`

Values are appended to a copy of this array.

095

`values : array_like`

These values are appended to a copy of `arr`. It must be of the correct shape (the same shape as `arr`, excluding `axis`). If `axis` is not specified, `values` can be any shape and will be flattened before use.

`axis : int, optional`

The axis along which `values` are appended. If `axis` is not given, both `arr` and `values` are flattened before use.

**Returns:**

`append : ndarray`

A copy of `arr` with `values` appended to `axis`. Note that `append` does not occur in-place: a new array is allocated and filled. If `axis` is `None`, `out` is a flattened array.



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance



Anaconda Prompt - python

```
<base> C:\Users\ybzhao>python
Python 3.6.4 |Anaconda custom (64-bit)| (default, Jan 16 2018, 10:22:32) [MSC v.
1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.append([1,2,3],[[4,5,6],[7,8,9]])
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.append([[1,2,3],[4,5,6]],[[7,8,9]],axis=0)
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>> np.append([[1,2,3],[4,5,6]],[[7],[8]],axis=1)
array([[1, 2, 3, 7],
       [4, 5, 6, 8]])
>>> np.append([[1,2,3],[4,5,6]],[7,8,9],axis=0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "C:\Continuum\anaconda3\lib\site-packages\numpy\lib\function_base.py", li
ne 5166, in append
      return concatenate((arr, values), axis=axis)
ValueError: all the input arrays must have same number of dimensions
>>>
```

095

# numpy.concatenate

`numpy.concatenate((a1, a2, ...), axis=0, out=None)`

Join a sequence of arrays along an existing axis.

**Parameters:** `a1, a2, ... : sequence of array_like`

The arrays must have the same shape, except in the dimension corresponding to `axis` (the first, by default).

`axis : int, optional`

The axis along which the arrays will be joined. If `axis` is `None`, arrays are flattened before use. Default is 0.

`out : ndarray, optional`

If provided, the destination to place the result. The `shape` must be correct, matching that of what `concatenate` would have returned if no `out` argument were specified.

**Returns:**

`res : ndarray`

The concatenated array.

096

`np.append([1, 2], [3, 4])`  $\Leftrightarrow$  `np.concatenate(([1, 2], [3, 4]))`



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

# numpy.random.seed

```
numpy.random.seed(seed=None)
```

Seed the generator.

Seed Number

097

This method is called when `RandomState` is initialized. It can be called again to re-seed the generator. For details, see `RandomState`.

Parameters: `seed : int or 1-d array_like, optional`

Seed for `RandomState`. Must be convertible to 32 bit unsigned integers.

```
>>> import numpy as np
>>> np.random.seed(0)
>>> print(np.random.randn(2))
[1.76405235 0.40015721]
>>> print(np.random.randn(2))
[0.97873798 2.2408932 ]
>>> np.random.seed(0)
>>> print(np.random.randn(4))
[1.76405235 0.40015721 0.97873798 2.2408932 ]
```





QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

(See sample exam paper, Question 6)



QF666  
Programming and  
Computational  
Finance



Dr. Zhao Yibao  
Senior Lecturer  
Of Quantitative  
Finance

The image displays three overlapping windows of a software application for calculating option prices and Greeks. The top window is titled "Option Price (and Greeks) Calculator" and shows the final results. The middle window is titled "Option" and contains input fields for stock price, exercise price, value date, expiration date, volatility, interest rate, dividend method, and yield rate. The bottom window is also titled "Option" and has a dropdown menu for "Model" (Black-Scholes, Binomial, or Monte Carlo). The "Monte Carlo Simulation" model is selected in the main calculator window.

	Call	Put
Theoretical Value	5.9383	5.1966
Delta	0.5743	-0.4207
Delta 100's	57.4298	-42.0682
Lambda (%)	4.8355	-4.0477
Gamma	0.0277	0.0277
Gamma (1%)	0.2871	-0.2104
Theta	-0.0168	-0.0128
Theta (7 days)	-0.1185	-0.0905
Vega	0.1378	0.1379
Rho	0.1142	-0.1315
Psi	-0.1440	0.1055
Strike Sensitivity	-0.4555	0.5246
Intrinsic Value	0.0000	0.0000
Time Value	5.9383	5.1966
Zero Volatility	0.7427	0.0000