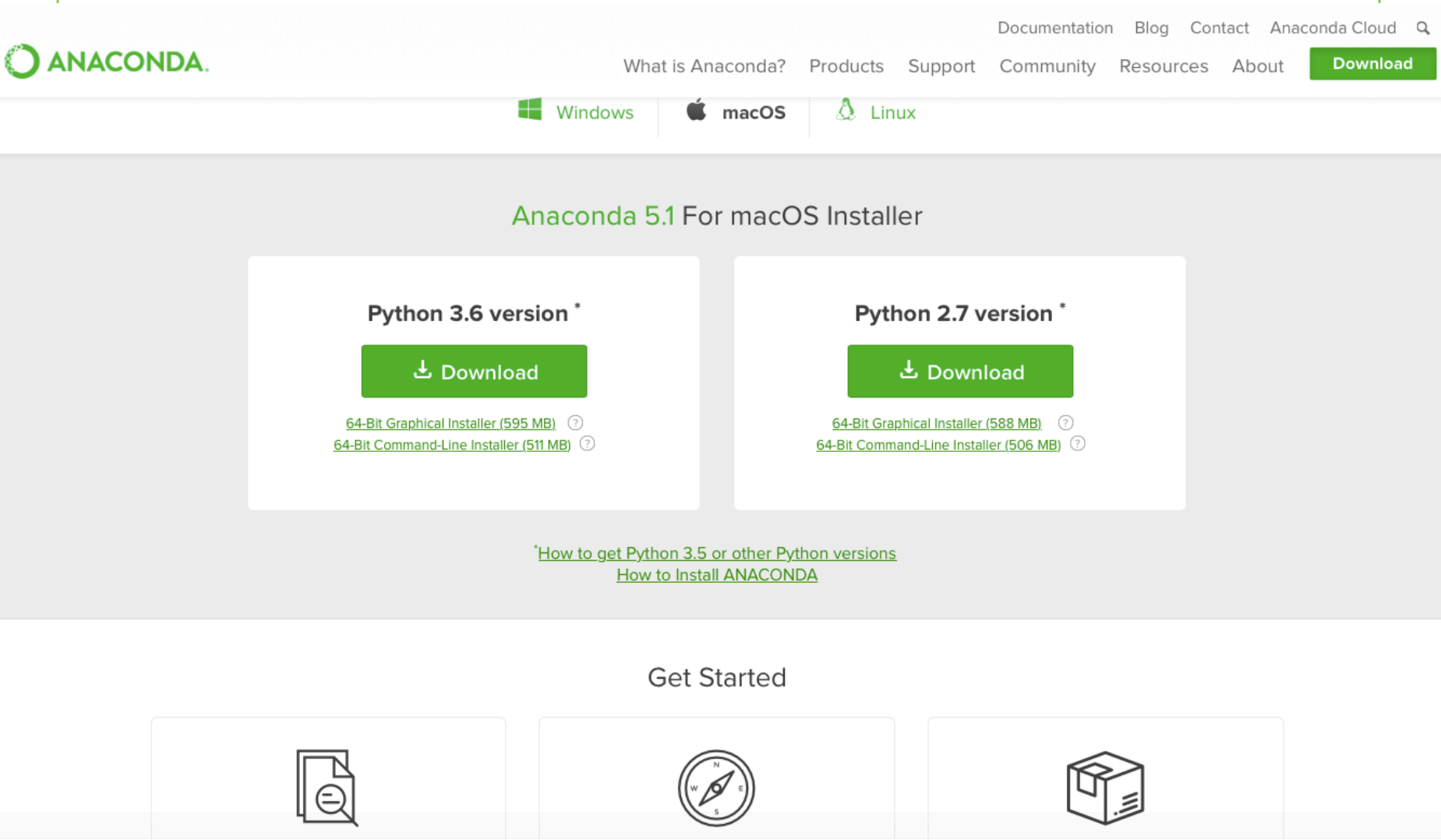
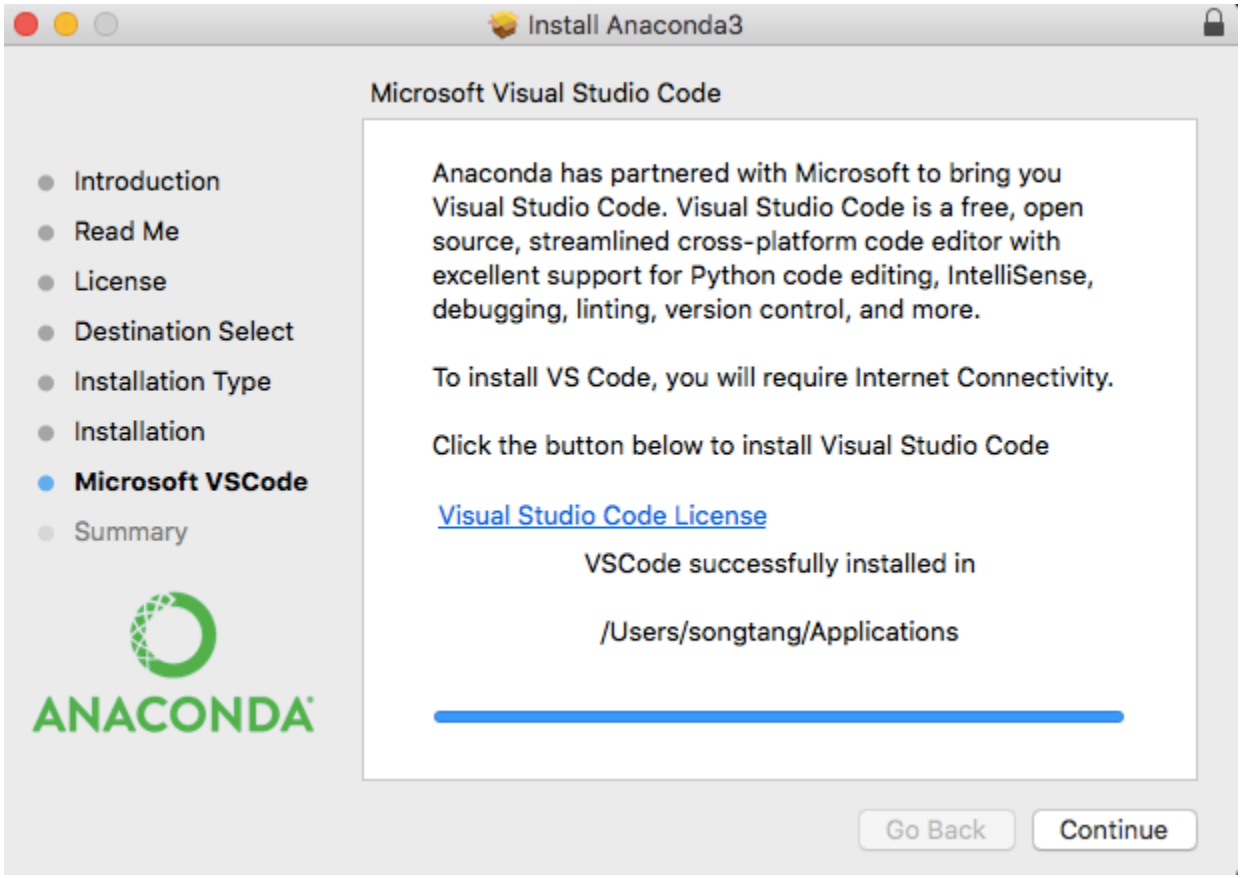
NYU | TANDON SCHOOL OF ENGINEERING

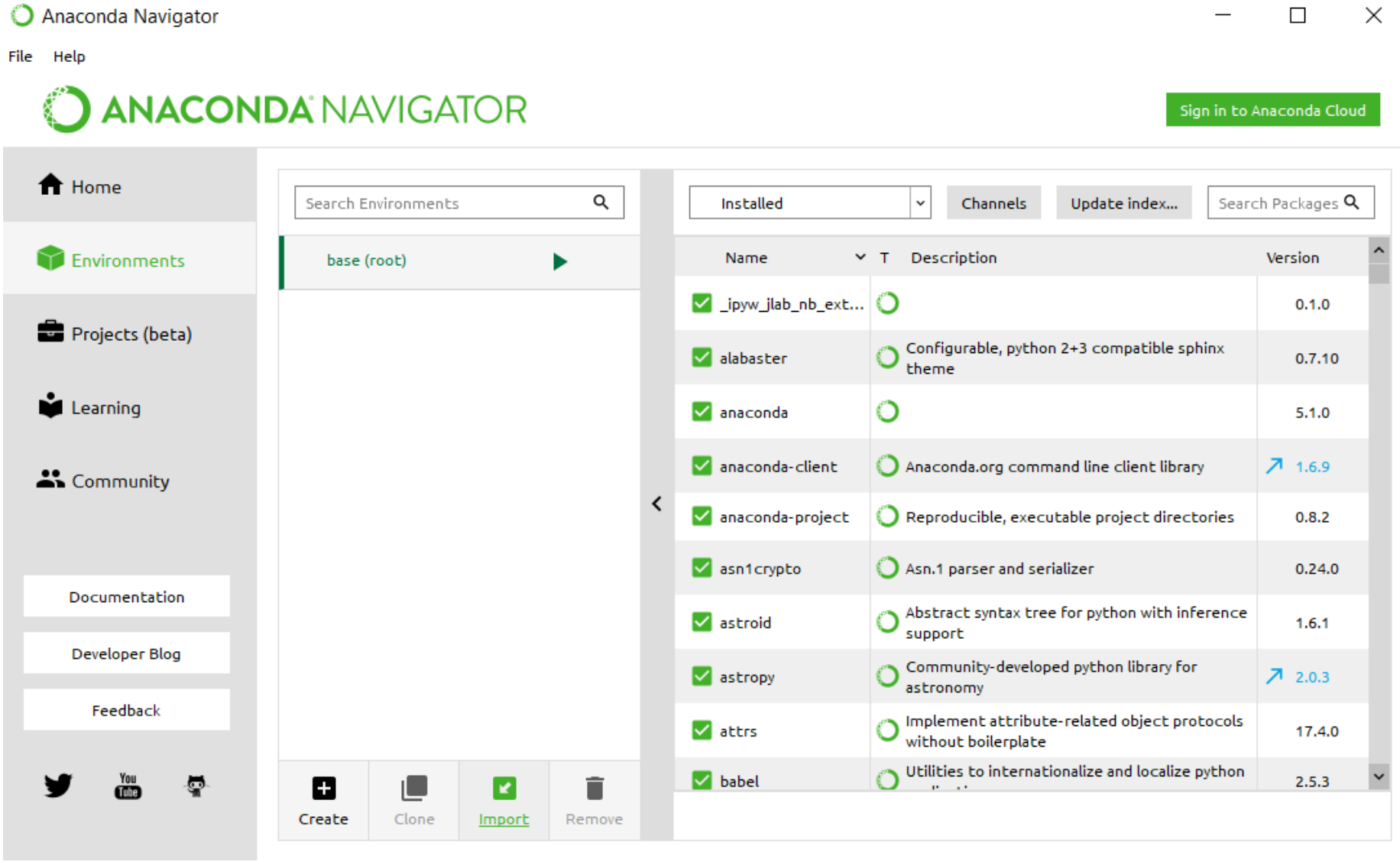
Python for Financial Data Analysis

Anaconda


Documentation
Blog
Contact
Anaconda Cloud
ANACONDA.
What is Anaconda?
Products
Support
Community
Resources
About
Download
Windows
macOS
Linux
Anaconda 5.1 For macOS Installer
Python 3.6 version *
Download
64-Bit Graphical Installer (595 MB)
64-Bit Command-Line Installer (511 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (588 MB)
64-Bit Command-Line Installer (506 MB)
*How to get Python 3.5 or other Python versions
How to Install ANACONDA
Get Started

Microsoft Visual Studio for Python




Anaconda Navigator
File
Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Projects (beta)
Learning
Community
Documentation
Developer Blog
Feedback
Search Environments
base (root)
Create
Clone
Import
Remove
Installed
Channels
Update index...
Search Packages
Name
T
Description
Version
_ipyw_jlab_nb_ext... 0.1.0
alabaster Configurable, python 2+3 compatible sphinx theme 0.7.10
anaconda 5.1.0
anaconda-client Anaconda.org command line client library 1.6.9
anaconda-project Reproducible, executable project directories 0.8.2
asn1crypto Asn.1 parser and serializer 0.24.0
astroid Abstract syntax tree for python with inference support 1.6.1
astropy Community-developed python library for astronomy 2.0.3
attrs Implement attribute-related object protocols without boilerplate 17.4.0
babel Utilities to internationalize and localize python 2.5.3

C:/Users/Song/Documents/NYU-Poly/PolyNYU2018/FRE7831_Spring2018/Python\PythonDataAnalysis - Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Project explorer

- PythonDataAnalysis
  - NumPy_Ex.py
  - SciPy_Ex2.py
  - SciPy_interpolation.p

Editor - C:\Users\Song\Documents\NYU-Poly\PolyNYU2018\FRE7831_Spring2018\Python\PythonDataAnalysis\...

SciPy_interpolation.py | NumPy_Ex.py | SciPy_Ex2.py

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy import interpolate
4
5 x = np.linspace(0, 10, 10)
6 y = np.exp(-x/3.0)
7 f = interpolate.interp1d(x, y)
8 f2 = interpolate.interp1d(x, y, kind='cubic')
9 xnew = np.linspace(0, 10, 40)
10 plt.plot(x,y,'o',xnew,f(xnew),'-', xnew, f2(xnew),'--')
11 plt.legend(['data', 'linear', 'cubic'], loc='best')
12 plt.show()
13
```

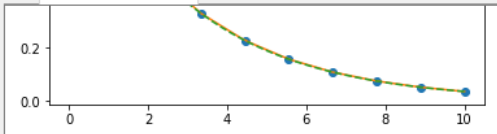
Variable explorer

Name	Type	Size	Value
A	int32	(3, 3)	matrix([[1, 2, 5], [2, 5, 1],
b	int32	(3, 1)	matrix([[10], [8],
x	float64	(10,)	array([0. , 1.11111111, 2…
x1	float64	(2, 3)	array([[1., 2., 3.], [4., 5., 6.]])
x2	float64	(2, 3)	matrix([[1., 2., 3.], [4., 5., 6.]])
xnew	float64	(40,)	array([0. , 0.25641026, 0.51282051, ..., 9.48717949, ...
xy1	float64	(2, 2)	array([[19., 23.], [43., 53.]])

Variable explorer | File explorer

IPython console

Console 1/A




```
In [3]: runfile('C:/Users/Song/Documents/NYU-Poly/
PolyNYU2018/FRE7831_Spring2018/Python/
PythonDataAnalysis/SciPy_Ex2.py', wdir='C:/Users/
Song/Documents/NYU-Poly/PolyNYU2018/
FRE7831_Spring2018/Python/PythonDataAnalysis')
A.I*b= [[-22.45]
 [ 10.09]
 [  2.45]]
[[-22.45]
 [ 10.09]
 [  2.45]]

In [4]: runfile('C:/Users/Song/Documents/NYU-Poly/
PolyNYU2018/FRE7831_Spring2018/Python/
PythonDataAnalysis/NumPy_Ex.py', wdir='C:/Users/Song/
Documents/NYU-Poly/PolyNYU2018/FRE7831_Spring2018/
Python/PythonDataAnalysis')
[[19. 23.]
```

Collection of items – List/Tuple/Dictionary

- List
  - Ordered collection of items
  - Can contain items of any type
  - Items in a list need not be of the same type.

```
>>> digits  = [0,1,2,3,4,5,6,7,8,9]
>>> strings = ["the", "dog", "ran"]
>>> list = ['physics', 'chemistry', 1997, 2000];
```

  - Delete list element:

```
>>> del list[2];
>>> ['physics', 'chemistry', 2000]
```

Accessing list contents

- Indices start from 0

```
>>> strings[0]
'the‘
>>> strings[2]
'ran,
```

- Items in a range

```
>>> digits[2:4]
[2, 3]
```

- Negative indices work backwards

```
>>> digits[-1]
9
>>> digits[-2]
8
```

Some list manipulations

- Add/remove

```
>>> strings.append("fast")
>>> strings.insert(1, "brown")
>>> strings
['the', 'brown', 'dog', 'ran', 'fast']
>>> digits.remove(8)
>>> digits
[0, 1, 2, 3, 4, 5, 6, 7, 9]
```

- Sort

```
>>> digits.sort(reverse=1)
>>> digits
[9, 7, 6, 5, 4, 3, 2, 1, 0]
>>> digits.sort()
>>> digits
[0, 1, 2, 3, 4, 5, 6, 7, 9]
```

Tuple

- Similar to lists
- But cannot be modified

```
>>> first_five = (1,2,3,4,5)
>>> first_five[2]
3
>>> first_five.append(6)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```

Converting between list & tuple

- Tuple to list

```
>>> newlist = list(first_five)
>>> newlist.append(6)
>>> newlist
[1, 2, 3, 4, 5, 6]
```

- List to tuple

```
>>> first_six = tuple(newlist)
>>> first_six.append(7)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> first_six
(1, 2, 3, 4, 5, 6)
```

Dictionary

- <key, value> pairs

```
>>> numbers = {1:"one", 2:"two", 3:"three"}
>>> letters = {"vowel":['a','e','i','o','u'],“consonant":['b','c','d','f','g']}
```

- Get value given key

```
>>> numbers[2]
'two'
>>> letters["consonant"]
['b', 'c', 'd', 'f', 'g']
```

- Changing the value associated with a key

```
>>> letters["consonant"].append('h')
>>> letters["consonant"]
['b', 'c', 'd', 'f', 'g', 'h']
>>> numbers[2]="twosome"
>>> numbers[2]
'twosome'
```

3rd party packages for data analysis

- There are a number of third-party packages available for data analysis, such as
  - NumPy/SciPy – numerical and scientific function libraries
  - Matplotlib – plotting library
  - Pandas – high-performance data structures and data analysis tools.
  - Turbodbc - accessing ODBC databases
  - Sqlite3 - easy interface to SQLite database.

Numpy

- An efficient multi-dimensional container for generic data.
  - The ndarray object, an n-dimensional array of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:
    - NumPy arrays have a fixed size. Modifying the size means creating a new array.
    - NumPy arrays must be of the same data type, but this can include Python objects.
    - More efficient mathematical operations than built-in sequence types.

NumPy – Numerical Python

- The fundamental package required for high performance scientific computing and data analysis:
  - **ndarray**, a fast and space-efficient multidimensional array providing vectorized arithmetic operations
  - Standard mathematical functions for fast operations on entire arrays of data without having to write loops
  - Tools for reading / writing array data to disk and working with memory-mapped files
  - Linear algebra, random number generation, and Fourier transform capabilities
  - Tools for integrating code written in C, C++, and Fortran

```
import numpy as np
# Method I
x1=np.array([[1,2,3],[4,5,6]],float) # 2 by 3
y1=np.array([[1,2],[3,3],[4,5]],float) # 3 by 2
xy1=np.dot(x1,y1) # 2 by 2
print (xy1)
# Method II
x2=np.matrix(x1)
y2=np.matrix(y1)
print (x2*y2)
-----
%run "C:/Users/Song/Documents/NYU-Poly/PolyNYU2018/
FRE7831/Python/NumPy_Ex.py"
[[ 19. 23.]
 [ 43. 53.]]
[[ 19. 23.]
 [ 43. 53.]]
```

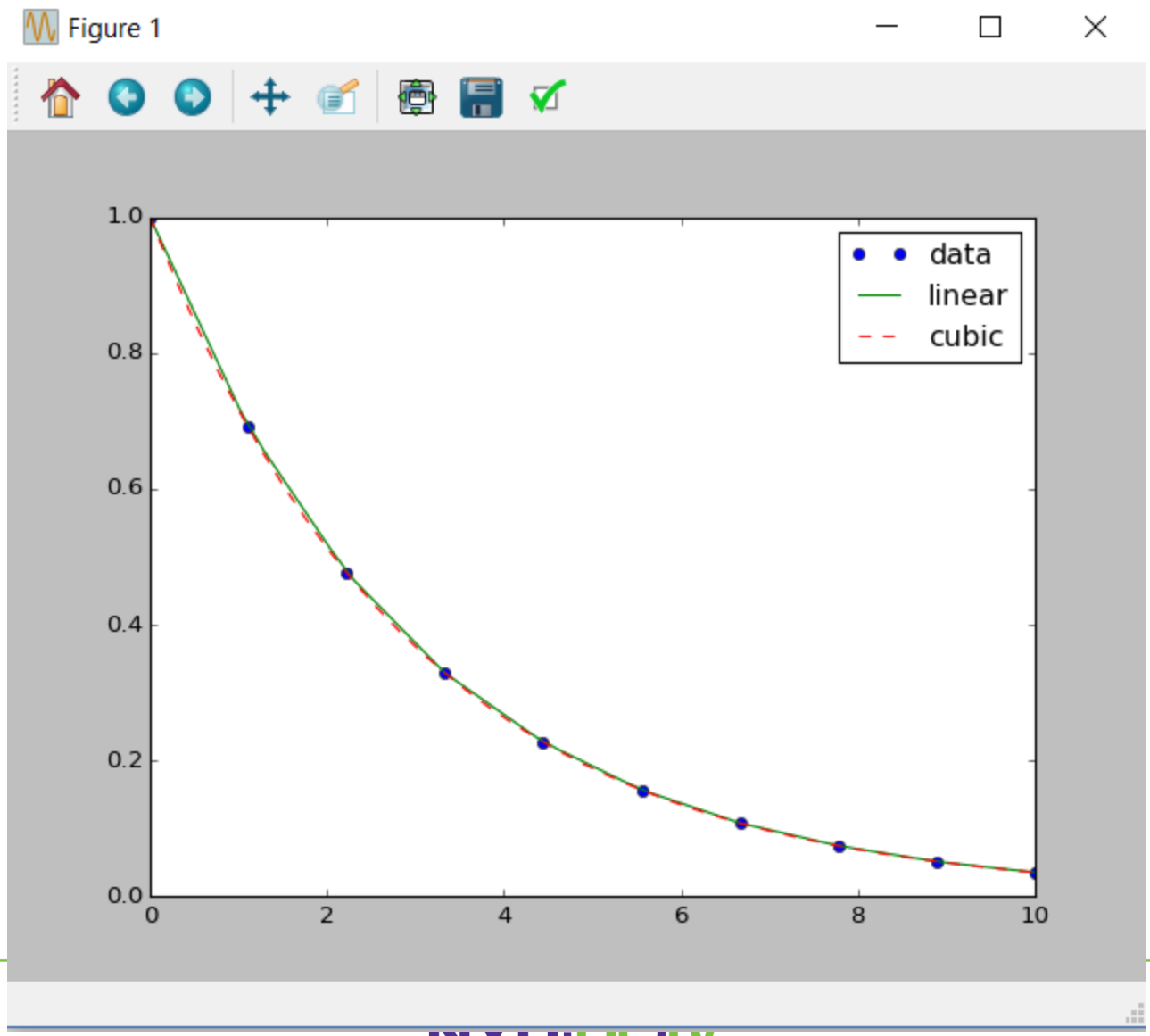
SciPy - tools and functions for scientific computing:

- Special mathematical functions (scipy.special)
- Integration (scipy.integrate)
- Optimization (scipy.optimize)
- Interpolation (scipy.interpolate)
- Fourier Transforms (scipy.fftpack)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)
- Compressed Sparse Graph Routines (scipy.sparse.csgraph)
- Spatial data structures and algorithms (scipy.spatial)
- Statistics (scipy.stats)
- Multidimensional image processing (scipy.ndimage)
- Data IO (scipy.io)
- Weave (scipy.weave)

Interpolation in SciPy

- import numpy as np
- import matplotlib.pyplot as plt
- from scipy.interpolate import interp1d

- x = np.linspace(0, 10, 10)
- y = np.exp(-x/3.0)
- f = interp1d(x, y)
- f2 = interp1d(x, y, kind='cubic')
- xnew = np.linspace(0, 10, 40)
- plt.plot(x,y,'o',xnew,f(xnew),'-', xnew, f2(xnew),'--')
- plt.legend(['data', 'linear', 'cubic'], loc='best')
- plt.show()



Solving linear equations using SciPy

- import scipy as sp
- import numpy as np
- A=sp.mat('[1 2 5; 2 5 1; 2 3 8]')
- b = sp.mat('[10;8;5]')
- print ('A.I*b=\n', np.round(A.I*b, decimals=2))
- print (np.round(np.linalg.solve(A,b), decimals=2))
- ---
  - %run "C:/Users/Song/Documents/NYU-Poly/PolyNYU2018/FRE7831/Python/SciPy_Ex2.py"
  - A.I*b=
  - [[-22.45]
  - [10.09]
  - [2.45]]
  - [[-22.45]
  - [10.09]
  - [2.45]]

$$\begin{cases} x+2y+5z=10 \\ 2x+5y+z=8 \\ 2x+3y+8z=5 \end{cases}$$

Understanding optimization

- In finance, many issues depend on optimization, such as choosing an optimal portfolio with an objective function and with a set of constraints. For those cases, we could use a SciPy optimization module called scipy.optimize. Assume that we want to estimate the *x* value that minimizes the value of *y*, where *y =3 + x^2*. Obviously, the minimum value of *y* is achieved when *x* takes a value of 0.
  - *import scipy.optimize as optimize*
  - *def my_f(x):*
    - *return 3 + x**2*
  - *optimize.fmin(my_f,5) # 5 is initial value*
- *%run "C:/Users/Song/Documents/NYU-Poly/PolyNYU2018/FRE7831/Python/SciPy_optimization.py"*
- *Optimization terminated successfully.*
- *Current function value: 3.000000*
- *Iterations: 20*
- *Function evaluations: 40*

Download Financial Data from Yahoo Finance

- #https://pypi.python.org/pypi/fix-yahoo-finance
- #pip install fix_yahoo_finance --upgrade --no-cache-dir
- from pandas_datareader import data as pdr
- import fix_yahoo_finance as yf
- yf.pdr_override()
- import datetime as dt
- start_date = dt.datetime(2018,5,1)
- end_date = dt.date.today()
-

- # download dataframe
- data = pdr.get_data_yahoo("SPY", start_date, end_date, as_panel=False, group_by = 'ticker')
- print("\n", data)

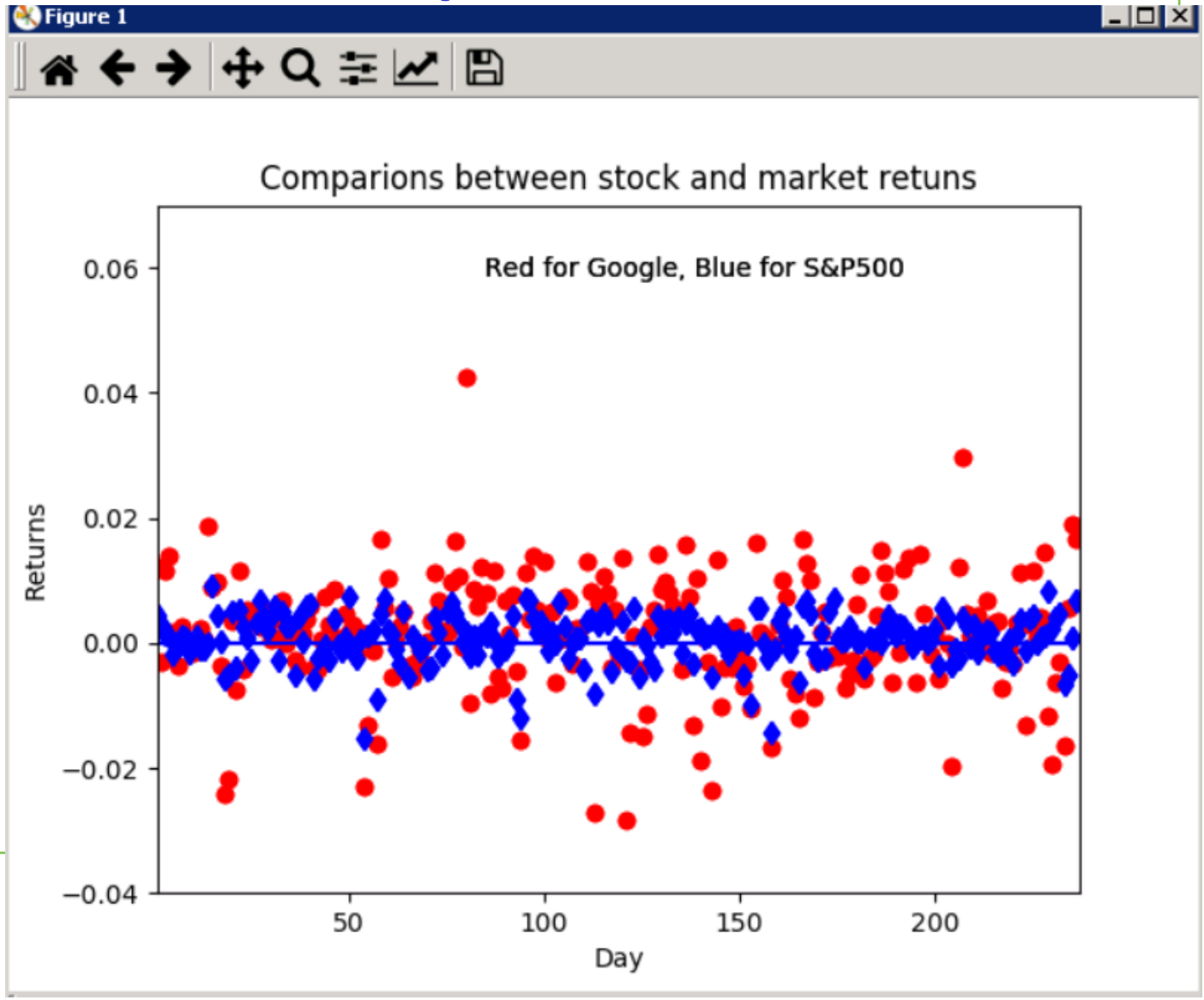
- data = pdr.get_data_yahoo(["SPY", "QQQ"], start_date, end_date, as_panel=False, group_by = 'ticker')
- print ("\n", data)

Download Financial Data from Yahoo Finance

```
In [6]: runfile('C:/Users/stang/Documents/PythonScripts/Fin/Yahoo_Finance_Fix.py', wdir='C:/
Documents/PythonScripts/Fin')
3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)]
[**********************100%**********************]  1 of 1 downloaded

                  Open        High         Low       Close   Adj Close     Volume
Date
2018-05-01  263.869995  265.100006  262.109985  264.980011  264.980011   74203400
2018-05-02  264.760010  265.679993  262.760010  263.200012  263.200012   86368900
2018-05-03  262.260010  263.359985  259.049988  262.619995  262.619995  134401900
[**********************100%**********************]  2 of 2 downloaded

                   SPY                                ...          QQQ
                  Open        High         Low      ...        Close   Adj Close    Volume
Date                                                 ...
2018-05-01  263.869995  265.100006  262.109985      ...   162.779999  162.779999  36513500
2018-05-02  264.760010  265.679993  262.760010      ...   161.820007  161.820007  38646800
2018-05-03  262.260010  263.359985  259.049988      ...   161.800003  161.800003  60129400

[3 rows x 12 columns]

In [7]:
```

Visual Finance via Matplotlib

Retrieving historical price data from Yahoo! Finance



- import datetime
- import fix_yahoo_finance as yf
- from matplotlib.pyplot import *
- import numpy as np
- from pandas_datareader import data as pdr
- yf.pdr_override()
- def ret_f(ticker, begdate, enddate):
  - p = pdr.get_data_yahoo(ticker, begdate, enddate,
    - as_panel=False, group_by = 'ticker',
    - auto_adjust = True, actions = True)
  - return p['Open'].pct_change(1)

- ret1=ret_f('GOOG',begdate,enddate)
- ret2=ret_f('^GSPC',begdate,enddate)
- n=min(len(ret1),len(ret2))
- t=range(n)
- line=np.zeros(n)
- plot(t,ret1[0:n], 'ro')
- plot(t,ret2[0:n], 'bd')
- plot(t,line,'b')
- figtext(0.4,0.8,"Red for Google, Blue for S&P500")
- xlim(1,n)
- ylim(-0.04,0.07)
- title("Comparions between stock and market retuns")
- xlabel("Day")
- ylabel("Returns")
- show()

Pandas - powerful data analysis toolkit

- Pandas - **Pan**el **Da**ta **S**ystem
- Data structures with labeled axes supporting automatic or explicit data alignment.
- Integrated time series functionality.
- The same data structures handle both time series data and non-time series data.
- Arithmetic operations and reductions (like summing across an axis) would pass on the metadata (axis labels).
- Flexible handling of missing data.
- Merge and other relational operations found in popular database databases (SQLbased, for example)

pandas - Data Structures: Series

- One-dimensional array-like object containing data and labels (or index)

```
import pandas as pd
s = pd.Series(list('abcdef'),
dtype=bytes); print (s)
0    b'a'
1    b'b'
2    b'c'
3    b'd'
4    b'e'
5    b'f'
dtype: bytes8
t = pd.Series([2,4,6,8]); print (t)
0    2
1    4
2    6
3    8
dtype: int64
```

Data Frame

- A DataFrame represents a tabular, spreadsheet-like data structure containing an ordered collection of columns, each of which can be a different value type (numeric, string, boolean, etc.). The DataFrame has both a row and column index; it can be thought of as a dict of Series (one for all sharing the same index).

Data Frame

Python

```
In [23]: import pandas as pd

In [24]: data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
    ...: 'year': [2000, 2001, 2002, 2001, 2002],
    ...: 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}

In [25]: frame = pd.DataFrame(data, columns=['year', 'state', 'pop'],
    ...:                          index=['one', 'two', 'three', 'four', 'five'])

In [26]: frame
Out[26]:
```

	year	state	pop
one	2000	Ohio	1.5
two	2001	Ohio	1.7
three	2002	Ohio	3.6
four	2001	Nevada	2.4
five	2002	Nevada	2.9

```
In [27]:
```

- import datetime
- import fix_yahoo_finance as yf
- from pandas_datareader import data as pdr
- import pandas as pd
- yf.pdr_override()
- begdate = datetime.date(2018,3,1)
- enddate = datetime.date.today()
- all_data = {}
- for ticker in ['AAPL', 'IBM', 'MSFT', 'GOOG']:

```
        all_data[ticker] = pdr.get_data_yahoo(ticker,
                                begdate, enddate, as_panel=False)
```

- price = pd.DataFrame({symbol: data['Close’]
  for symbol, data in all_data.items()})
- volume = pd.DataFrame({symbol: data['Volume’]
  for symbol, data in all_data.items()})
- returns = price.pct_change()
- print (returns.tail())
- print (returns.corr())
- pd.options.display.float_format='{:,.6f}'.format
- print (returns.cov())

	AAPL	GOOG	IBM	MSFT
Date				
2018-04-27	-0.01	-0.01	-0.00	0.02
2018-04-30	0.02	-0.01	-0.01	-0.02
2018-05-01	0.02	0.02	0.00	0.02
2018-05-02	0.04	-0.01	-0.02	-0.02
2018-05-03	0.00	-0.00	-0.00	0.01

	AAPL	GOOG	IBM	MSFT
AAPL	1.00	0.57	0.41	0.63
GOOG	0.57	1.00	0.55	0.84
IBM	0.41	0.55	1.00	0.59
MSFT	0.63	0.84	0.59	1.00

	AAPL	GOOG	IBM	MSFT
AAPL	0.000305	0.000191	0.000129	0.000226
GOOG	0.000191	0.000369	0.000188	0.000332
IBM	0.000129	0.000188	0.000320	0.000219
MSFT	0.000226	0.000332	0.000219	0.000427

Panel Data

- A three-dimensional analogue of DataFrame
  - import datetime
  - import fix_yahoo_finance as yf
  - from pandas_datareader import data as pdr
  - yf.pdr_override()
  - begdate = datetime.date(2017,12,1)
  - enddate = datetime.date.today()
  - stk = ['AAPL', 'GOOG', 'MSFT', 'FB']
  - pdata = pdr.get_data_yahoo(stk, begdate, enddate,
    as_panel=True, group_by = 'ticker')

- print (pdata)

```
<class 'pandas.core.panel.Panel'>
Dimensions: 4 (items) x 24 (major_axis) x 6
(minor_axis)
Items axis: AAPL to MSFT
Major_axis axis: 2018-04-02 00:00:00 to 2018-05-03
00:00:00
Minor_axis axis: Open to Volume
```

- pd.options.display.float_format = '{:,.2f}'.format
- stacked = pdata.to_frame()
- print (stacked)

Date	minor	AAPL	FB	GOOG	MSFT
2018-04-02	Open	166.64	157.81	1,022.82	90.47
	High	168.94	159.20	1,034.80	90.88
	Low	164.47	154.11	990.37	87.51
	Close	166.68	155.39	1,006.47	88.52
	Adj Close	166.68	155.39	1,006.47	88.52
	Volume	37,586,800.00	36,796,000.00	2,680,400.00	48,515,400.00
2018-04-03	Open	167.64	156.55	1,013.91	89.58
	High	168.75	157.39	1,020.99	90.05
	Low	164.88	150.81	994.07	87.89
	Close	168.39	156.11	1,013.41	89.71
	Adj Close	168.39	156.11	1,013.41	89.71
	Volume	30,278,000.00	42,034,000.00	2,275,100.00	37,213,800.00
2018-04-04	Open	164.88	152.03	993.41	87.85
	High	172.01	155.56	1,028.72	92.76
	Low	164.77	150.51	993.00	87.73
	Close	171.61	155.10	1,025.14	92.33
	Adj Close	171.61	155.10	1,025.14	92.33
	Volume	34,605,500.00	49,885,600.00	2,484,700.00	35,560,000.00
2018-04-05	Open	172.58	161.56	1,041.33	92.44
	High	174.23	161.57	1,042.79	93.07
	Low	172.08	156.65	1,020.13	91.40
	Close	172.80	159.34	1,027.81	92.38
	Adj Close	172.80	159.34	1,027.81	92.38
	Volume	26,933,200.00	41,449,600.00	1,363,000.00	29,771,900.00

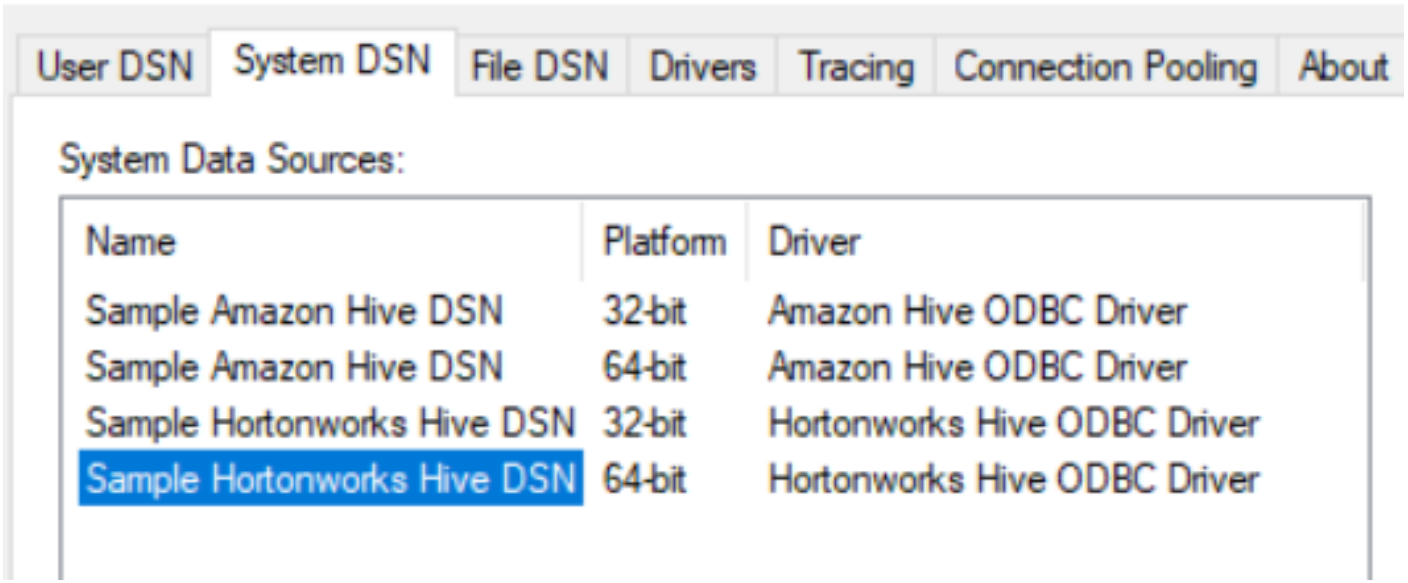
- print (pdata[:, :, 'Volume'].astype('int'))

Date	AAPL	FB	GOOG	MSFT
2017-12-01	39759300	20182500	1909600	29532100
2017-12-04	32542400	24459400	1906400	39094900
2017-12-05	27350200	20184900	2067300	26152300
2017-12-06	28560000	20255800	1272000	26162100
2017-12-07	25673300	20404500	1458200	23184500
2017-12-08	23173700	19404300	1281200	23861800

Connecting to Hortonworks Database

- Match Python env with Hortonworks ODBC driver:

Python 3.6.4 64bits, Qt 5.6.2, PyQt5 5.6 on Windows



Using Python TurbODBC pacakge

- # http://turbodbc.readthedocs.io/en/latest/pages/getting_started.html
- # pip install turbodbc
- import turbodbc
- options = turbodbc.make_options(autocommit=True)
- conn = turbodbc.connect(DSN="Sample Hortonworks Hive DSN;",
  turbodbc_options=options)
- cursor = conn.cursor();
- cursor.execute("select * from stocks.price_data where symbol = 'IBM';")
- result = cursor.fetchall()
- for r in result:
- print(r)

FindODBC.py | hortonworks.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4
5  OBDC for retrieving Hortonworks Database.
6  """
7  # http://turbodbc.readthedocs.io/en/latest/pages/getting_started.html
8  # pip install turbodbc
9
10 import turbodbc
11
12 options = turbodbc.make_options(autocommit=True)
13 conn = turbodbc.connect(DSN="Sample Hortonworks Hive DSN;", turbodbc_options=options)
14
15 cursor = conn.cursor();
16
17 cursor.execute("select * from stocks.price_data where symbol = 'IBM';")
18
19 result = cursor.fetchall()
20 for r in result:
21     print(r)
22
23
```

Name	Type	Size	Value
A	int32	(3, 3)	matrix([[1, 2, 5], [2, 5, 1],
b	int32	(3, 1)	matrix([[10],

Variable explorer | File explorer

IPython console

Console 1/A

```
419200, 2.559999942779541]
['NYSE', 'IBM', '1962-01-16', 566.0, 566.0, 560.5, 560.5,
251200, 2.5999999046325684]
['NYSE', 'IBM', '1962-01-15', 566.0, 567.75, 566.0, 566.5,
251200, 2.630000114440918]
['NYSE', 'IBM', '1962-01-12', 564.0, 568.0, 564.0, 564.0,
435200, 2.619999885559082]
['NYSE', 'IBM', '1962-01-11', 558.5, 563.0, 558.5, 563.0,
315200, 2.609999895095825]
['NYSE', 'IBM', '1962-01-10', 557.0, 559.5, 557.0, 557.0,
299200, 2.5799999237060547]
['NYSE', 'IBM', '1962-01-09', 552.0, 563.0, 552.0, 556.0,
491200, 2.5799999237060547]
['NYSE', 'IBM', '1962-01-08', 559.5, 559.5, 545.0, 549.5,
544000, 2.549999952316284]
['NYSE', 'IBM', '1962-01-05', 570.5, 570.5, 559.0, 560.0,
363200, 2.5999999046325684]
['NYSE', 'IBM', '1962-01-04', 577.0, 577.0, 571.0, 571.25,
256000, 2.6500000953674316]
['NYSE', 'IBM', '1962-01-03', 572.0, 577.0, 572.0, 577.0,
288000, 2.680000066757202]
['NYSE', 'IBM', '1962-01-02', 578.5, 578.5, 572.0, 572.0,
387200, 2.6500000953674316]
```

```
C:\sqlite>sqlite3 Quiz2.db
        SQLite version 3.18.0 2018-04-28 18:48:43
        Enter ".help" for usage hints.
sqlite> .database
        main: C:\sqlite\Quiz2.db
sqlite> .tables
        Executions  Orders     Symbols     Traders
sqlite> .header on
sqlite> select * from executions;
        ExecutionsID|Status|Exchange|FreeText|OrderID|TraderID
        123|6|P|Partial Filled|GSJA000111122015|1234
        124|8|P|Filled|JPAB000111132015|1245
        125|0|N|Pending|CTSA000111132015|2345
        126|4|P|Cancelled|GSJA000211122015|1234
        127|0|N|Pending|JPAB000211132015|1245
        128|8|N|Filled|CTSA000211132015|2345
sqlite> .quit
```

Using SQLite in Python

- As an example, we are going to (1) implement the relational database for Quiz 1 in SQLite; (2) recreate the flat table by joining the 4 tables in our database

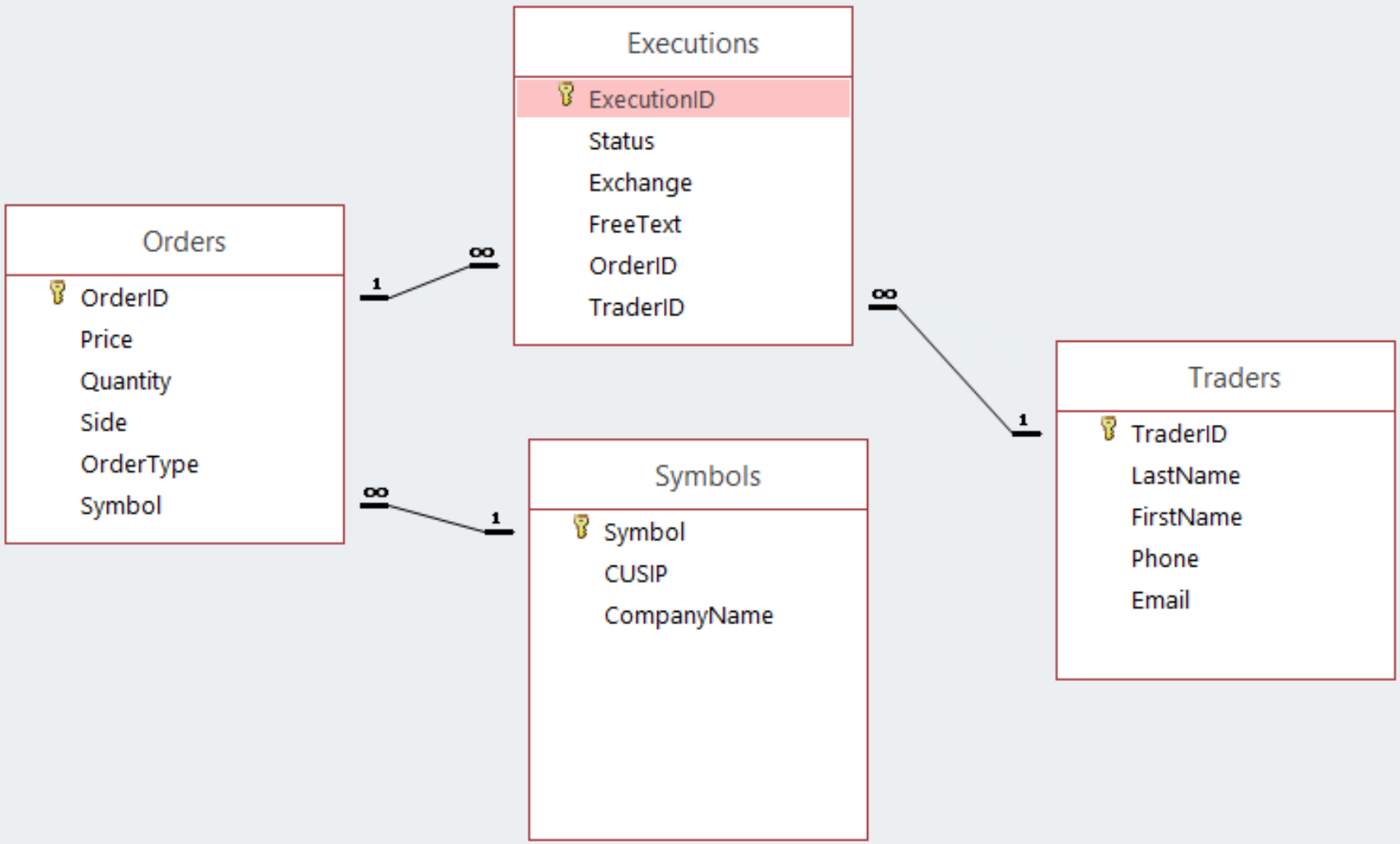
FRE7831 Spring 2018 Quiz 1, Date: 4/21/2018, Name _____

Symbol	CUSIP	Company Name	TraderID	Last Name	First Name	Email	Phone	OrderID
GOOG	38259P508	Alphabet Inc.	1234	ABC	Joe	JA123@gs.com	212-111-1234	GSJA000111122015
GOOG	38259P508	Alphabet Inc.	1245	BCD	Andy	AB124@jp.com	212-222-1234	JPAB000111132015
IBM	459200101	International Business Machines Corporation	2345	ABC	Smith	SA234@citi.com	212-333-1234	CTSA000111132015
C	312072001	Citigroup Inc	1234	ABC	Joe	JA123@gs.com	212-111-1234	GSJA000211122015
IBM	459200101	International Business Machines Corporation	1245	BCD	Andy	AB124@jp.com	212-222-1234	JPAB000211132015
C	312072001	Citigroup Inc	2345	ABC	Smith	SA234@citi.com	212-333-1234	CTSA000211132015

(1) Convert the above flat table into tables in a relational database with 4 tables: symbol table, trader table, order table and execution ta

(2) Show or Label proper primary key and foreign key (if have) for each table. So by joining through PK and FK, the orginal flat table can be

(3) Draw an E-R diagram for your database.


Executions
ExecutionID
Status
Exchange
FreeText
OrderID
TraderID
Orders
OrderID
Price
Quantity
Side
OrderType
Symbol
Traders
TraderID
LastName
FirstName
Phone
Email
Symbols
Symbol
CUSIP
CompanyName
1
∞
1
∞
∞
1

Create and Populate Tables

```
import sqlite3
connection = sqlite3.connect("C:/sqlite/quiz.db")
cursor = connection.cursor()
cursor.execute("""Drop table if exists Symbols;""")
```

```
sql_command = """
Create Table if not exists Symbols (
 Symbol        text              Primary Key   Not Null,
 CUSIP         text              Not Null,
 CompanyName   text              Not Null
);
"""
cursor.execute(sql_command)
cursor.executemany("Insert into Symbols(Symbol, CUSIP, CompanyName)
Values(?, ?, ?)",
[
("GOOG", "38259P508", "Alphabet Inc."),
("IBM", "459200101", "International Business Machines Corporation."),
("C", "312072001", "Citigroup.")
]
)
```

```
# Traders
cursor.execute("""Drop table if exists Traders;""")


sql_command = """
Create Table if not exists Traders (
 TraderID      text          Primary Key   Not Null,
 LastName      text          Not Null,
 FirstName     text          Not Null,
 Email         text          Not Null,
 Phone         text          Not Null
);
"""
cursor.execute(sql_command)
```

- cursor.executemany(“Insert into Traders Values(?, ?, ?, ?, ?)",
- [
- ("1234", "ABC", "Joe", "JA123@gs.com", "212-111-1234"),
- ("1245", "BCD", "Andy", "AB123@jp.com", "212-222-1234"),
- ("2345", "ABC", "Smith", "SA234@citi.com", "212-333-1234")
-]
-)

```
# Orders
cursor.execute("""Drop table if exists Orders;""")
sql_command = """
Create Table if not exists Orders (
 OrderID      text          Primary Key   Not Null,
 Price        real          ,
 Quantity     integer       Not Null,
 Side         char(1)       Not Null,
 OrderType    text          Not Null,
 Symbol       text          Not Null,
 Foreign Key(Symbol) references Symbols(Symbol)
);
"""
cursor.execute(sql_command)
```

- cursor.executemany(“Insert into Orders Values(?, ?, ?, ?, ?, ?)",
- [
- ("GSJA000111122015", "717.01", "200", "B", "Limit", "GOOG"),
- ("JPAB000111132015", "", "125", "S", "Market", "GOOG"),
- ("CTSA000111132015", "131.75", "1000", "B", "Limit", "IBM"),
- ("GSJA000211122015", "53.17", "10000", "B", "Limit", "C"),
- ("JPAB000211132015", "131.70", "2000", "S", "Limit", "IBM"),
- ("CTSA000211132015", "", "200", "B", "Market", "C")
-]
-)

```
# Executions
cursor.execute("""Drop table if exists Executions;""")
sql_command = """
Create Table if not exists Executions (
  ExecutionsID text Primary Key   Not Null,
  Status              char(1)              Not Null,
  Exchange          char(1)              Not Null,
  FreeText           text                    ,
  OrderID             text                 Not Null,
  TraderID            text                 Not Null,
  Foreign Key(OrderID) references Orders(OrderID),
  Foreign Key(TraderID) references Traders(TraderID)
);
"""
cursor.execute(sql_command)
```

```
cursor.executemany("insert into Executions Values(?, ?, ?, ?, ?, ?)",
[
("123", "6", "P", "Partial Filled", "GSJA000111122015", "1234"),
("124", "8", "P", "Filled", "JPAB000111132015", "1245"),
("125", "0", "N", "Pending", "CTSA000111132015", "2345"),
("126", "4", "P", "Cancelled", "GSJA000211122015", "1234"),
("127", "0", "N", "Pending", "JPAB000211132015", "1245"),
("128", "8", "N", "Filled", "CTSA000211132015", "2345")
]
)

connection.commit()
connection.close()
```

```
C:\sqlite>sqlite3 quiz.db
SQLite version 3.18.0 2017-03-28 18:48:43
Enter ".help" for usage hints.
sqlite> .tables
Executions  Orders     Symbols    Traders
sqlite> SELECT Symbols.*, Traders.*, Orders.*, Executions.* FROM Traders INNER JOIN (Symbols INNER JOIN (Orders INNER
JOIN Executions ON Orders.OrderID = Executions.OrderID) ON Symbols.Symbol = Orders.Symbol) ON Traders.
[TraderID]=Executions.TraderID;

GOOG|38259P508|Alphabet Inc.|1234|ABC|Joe|JA123@gs.com|212-111-1234|GSJA00011112
2015|717.01|200|B|Limit|GOOG|123|6|P|Partial Filled|GSJA000111122015|1234
GOOG|38259P508|Alphabet Inc.|1245|BCD|Andy|AB123@jp.com|212-222-1234|JPAB0001111
32015||125|S|Market|GOOG|124|8|P|Filled|JPAB000111132015|1245
IBM|459200101|International Business Machines Corporation.|2345|ABC|Smith|SA234@
citi.com|212-333-1234|CTSA000111132015|131.75|1000|B|Limit|IBM|125|0|N|Pending|C
TSA000111132015|2345
C|312072001|Citigroup.|1234|ABC|Joe|JA123@gs.com|212-111-1234|GSJA000211122015|5
3.17|10000|B|Limit|C|126|4|P|Cancelled|GSJA000211122015|1234
IBM|459200101|International Business Machines Corporation.|1245|BCD|Andy|AB123@j
p.com|212-222-1234|JPAB000211132015|131.7|2000|S|Limit|IBM|127|0|N|Pending|JPAB0
00211132015|1245
C|312072001|Citigroup.|2345|ABC|Smith|SA234@citi.com|212-333-1234|CTSA0002111320
15||200|B|Market|C|128|8|N|Filled|CTSA000211132015|2345
```

```
import sqlite3
connection = sqlite3.connect("C:/sqlite/quiz.db")
cursor = connection.cursor()

cursor.execute(" SELECT Symbols.*, Traders.*, Orders.*, Executions.* \
  FROM Traders INNER JOIN (Symbols INNER JOIN (Orders INNER JOIN \
  Executions ON Orders.OrderID = Executions.OrderID)  \
  ON Symbols.Symbol = Orders.Symbol) ON \
  Traders.[TraderID]=Executions.TraderID ;");

result = cursor.fetchall()
for r in result:
  print(r)
```

Python C:\Users\stang

```
In [1]: %run "C:\Users\stang\Documents\PythonScripts\db3.py"
fetchall:
(u'GOOG', u'38259P508', u'Alphabet Inc.', u'1234', u'ABC', u'Joe', u'JA123@gs.com', u'212-111-1234', u'GSJA000111122015',
717.01, 200, u'B', u'Limit', u'GOOG', u'123', u'6', u'P', u'Partial Filled', u'GSJA000111122015', u'1234')
(u'GOOG', u'38259P508', u'Alphabet Inc.', u'1245', u'BCD', u'Andy', u'AB123@jp.com', u'212-222-1234', u'JPAB000111132015',
u'', 125, u'S', u'Market', u'GOOG', u'124', u'8', u'P', u'Filled', u'JPAB000111132015', u'1245')
(u'IBM', u'459200101', u'International Business Machines Corporation.', u'2345', u'ABC', u'Smith', u'SA234@citi.com',
u'212-333-1234', u'CTSA000111132015', 131.75, 1000, u'B', u'Limit', u'IBM', u'125', u'0', u'N', u'Pending',
u'CTSA000111132015', u'2345')
(u'C', u'312072001', u'Citigroup.', u'1234', u'ABC', u'Joe', u'JA123@gs.com', u'212-111-1234', u'GSJA000211122015', 53.17,
10000, u'B', u'Limit', u'C', u'126', u'4', u'P', u'Cancelled', u'GSJA000211122015', u'1234')
(u'IBM', u'459200101', u'International Business Machines Corporation.', u'1245', u'BCD', u'Andy', u'AB123@jp.com',
u'212-222-1234', u'JPAB000211132015', 131.7, 2000, u'S', u'Limit', u'IBM', u'127', u'0', u'N', u'Pending',
u'JPAB000211132015', u'1245')
(u'C', u'312072001', u'Citigroup.', u'2345', u'ABC', u'Smith', u'SA234@citi.com', u'212-333-1234', u'CTSA000211132015', u'',
200, u'B', u'Market', u'C', u'128', u'8', u'N', u'Filled', u'CTSA000211132015', u'2345')

In [2]:
```

References

- Python for Finance, Build real-life Python applications for quantiative finance and financial engineering. Yuxing Yan, ISBN 978-1-78328-437-5, April 2014
- Python for Data Analysis, Wes, McKinney, 2013, ISBN: 978-1-449-31979-3
- Getting started with Python/NLTK - CIS @ Upenn, *www.cis.upenn.edu/~cis530/recitation-2010/recitation-py.ppt*
- SQLite - https://en.wikipedia.org/wiki/SQLite
- SQLite, wiki.ggc.usg.edu/images/c/ca/SQLite.ppt
- Python for Data Analysis – IDEAL, *ideal.ece.utexas.edu/.../ppt/Research_Paper_Presentation_Pandas_Moshiul_Arefin.pdf*