Computer Simulations and Risk Assessment – Lecture 2

Fall 2019

Brandeis International Business School

Course Information - Schedule

Class Date	Text Chapters
Aug. 30, 2019 – L1	• Course Introduction/Python Installation • Introduction to Quantitative Finance Career • Python basics
Sep. 6, 2019 – L2	• Advanced Python Topics
Sep. 13, 2019 – L3	• Advanced Python Topics
Sep. 20, 2019 – L4	• Sourcing and handling Data • Stylized financial data analysis using Python
Sep. 27, 2019 – L5	• Value at Risk
Oct. 4, 2019 – L6	• Conditional Value at Risk (Expected Shortfall) + Mid-term Review
Oct. 11, 2019	• Mid-term
Oct. 18, 2019 – L7	• Modeling Volatility I
Oct. 25, 2019 – L8	• Modeling Volatility II
Nov. 1, 2019 – L9	• Practical application case Studies I
Nov. 8, 2019 – L10	• Practical application case Studies II
Nov. 15, 2019 – L11	• Back Testing + Conditional risk prediction
Nov. 22, 2019 – L12	• Research project presentation
Dec. 6, 2019 – L13	• Final Review

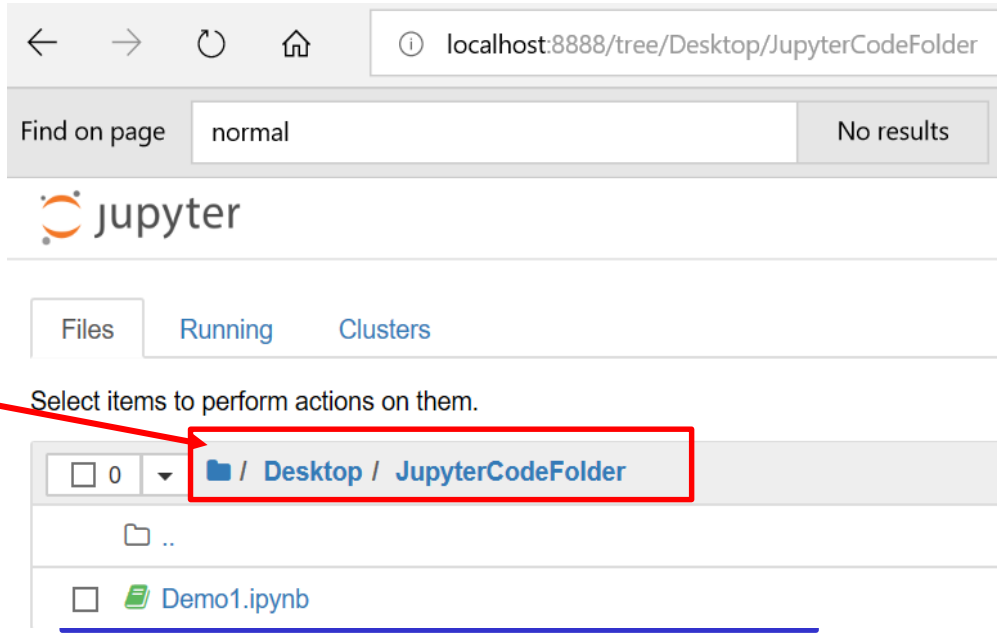
Introduction to Python Programming

- **Jupyter notebook introduction**
- **Packages**
- **Functions and Methods**
- **Matrix operation and manipulation**
- **Control structure**
- **Data I/Os**
- **Data types and structures**
- **Charting**

Jupyter Notebook Introduction

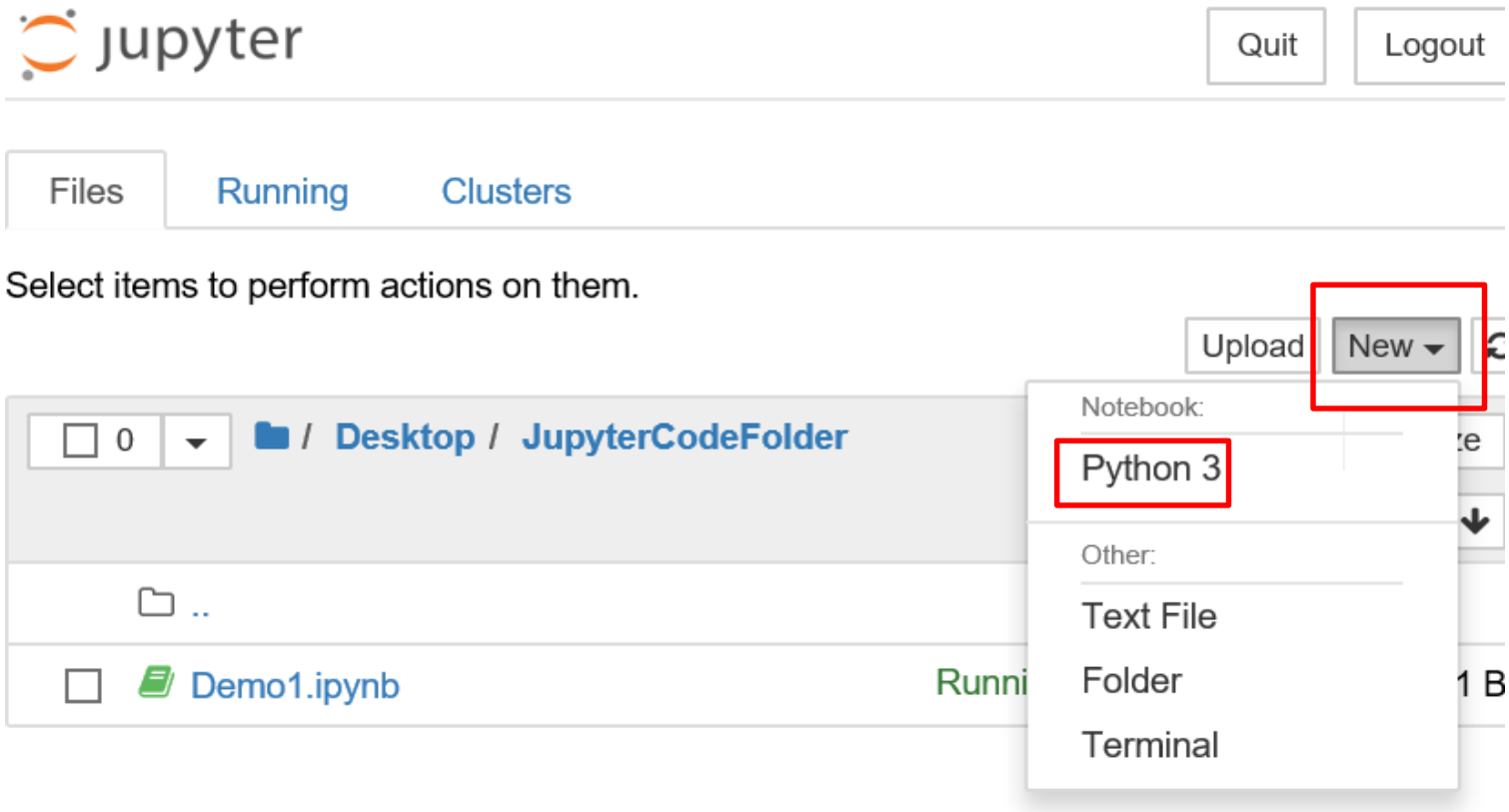
- **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text
- For submitting homework, please use Jupyter Notebook to organize your code for each problem into cells and submit one notebook for each homework assignment (The TAs will help you on this)
- To begin Jupyter notework, double click the icon 

- Then go to the folder where you want to have the code saved



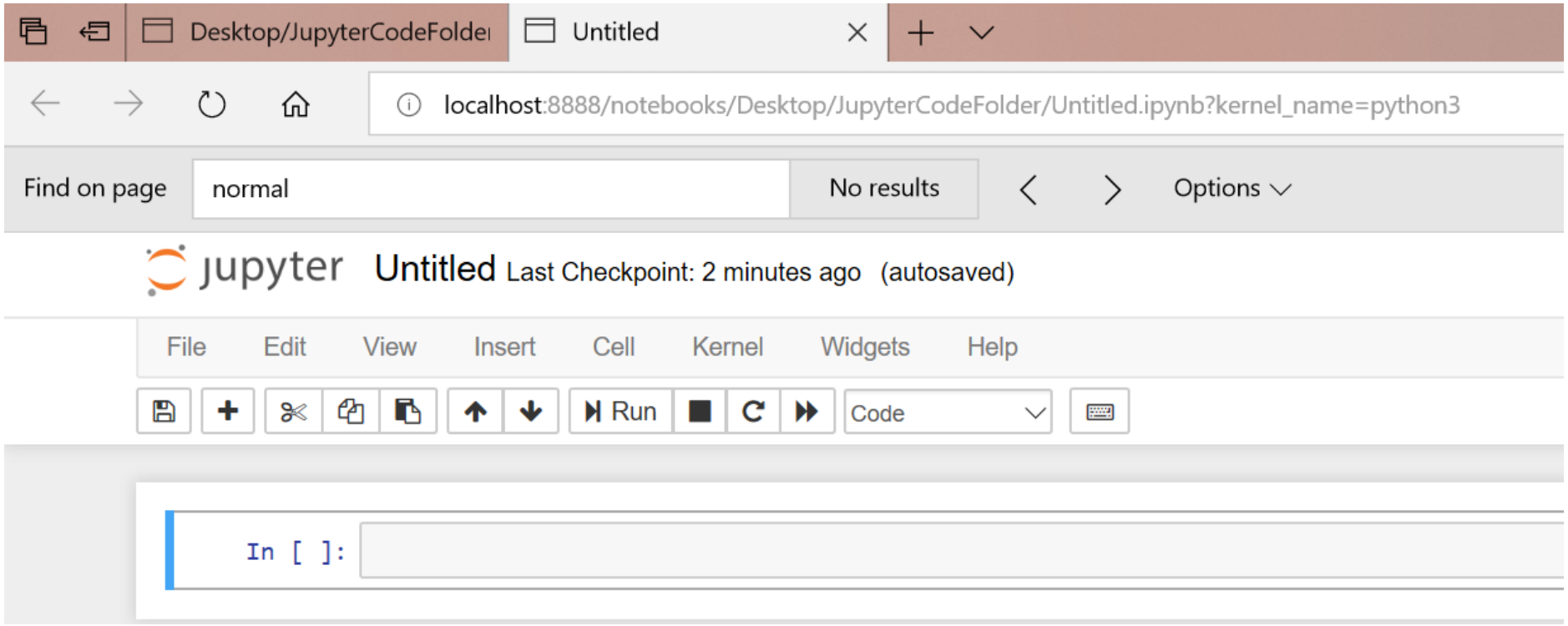
Jupyter Notebook Introduction

- Then click New, select Python 3



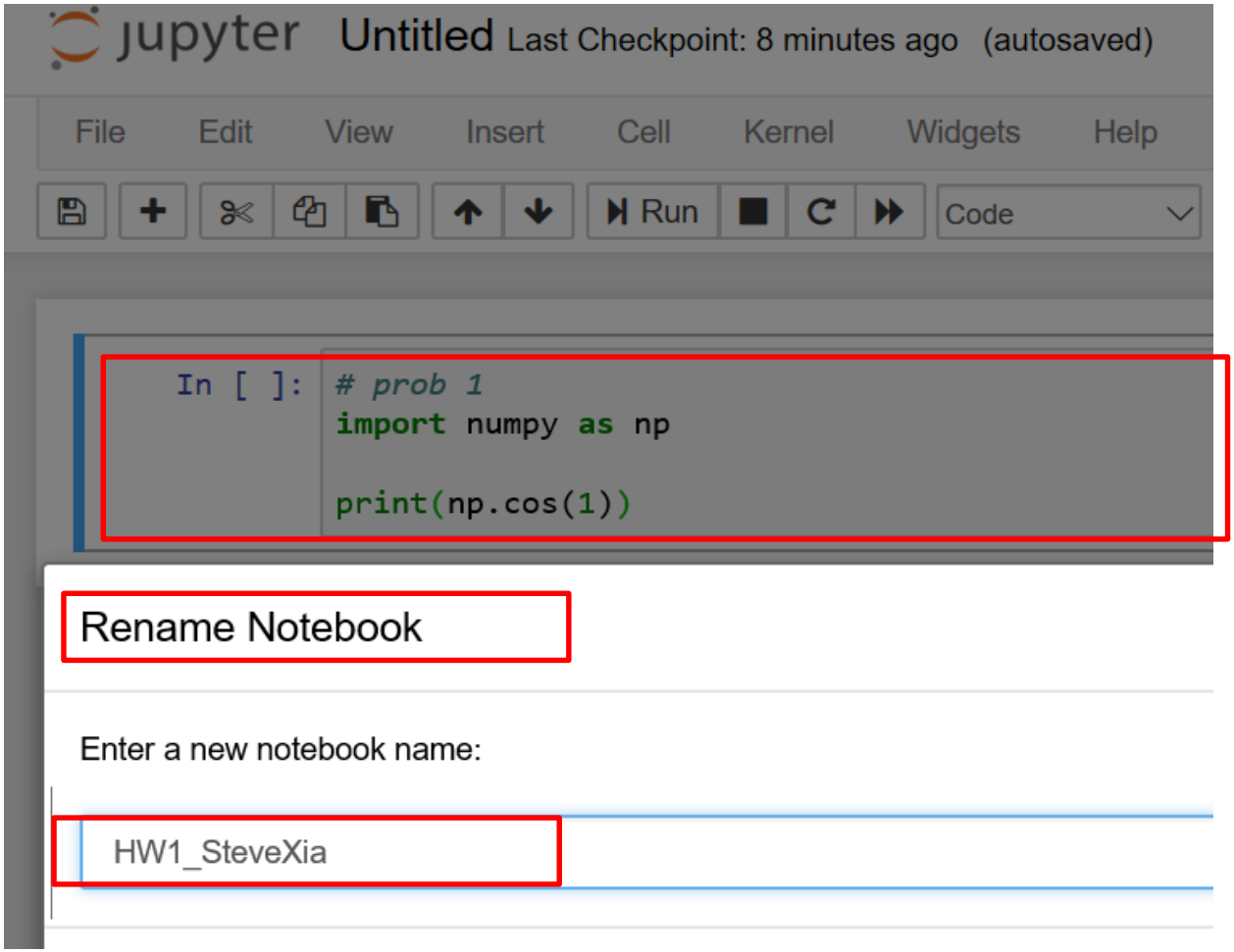
Jupyter Notebook Introduction

- A new window with name Untitled will pop up



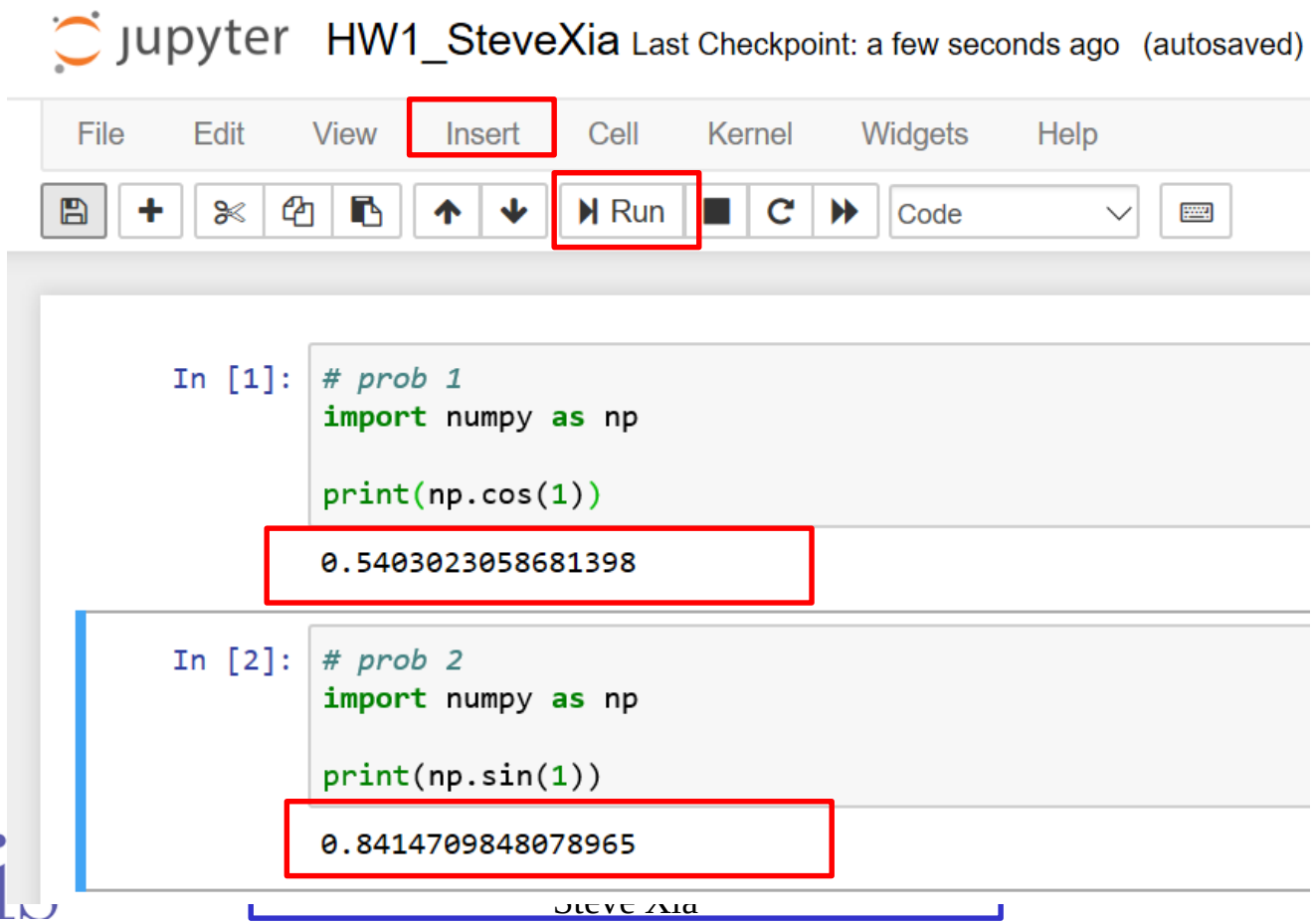
Jupyter Notebook Introduction

- Copy the first portion of the code (e.g., hw1, problem 1) from your Spyder editor here as the first cell, then rename it HW1_YourName



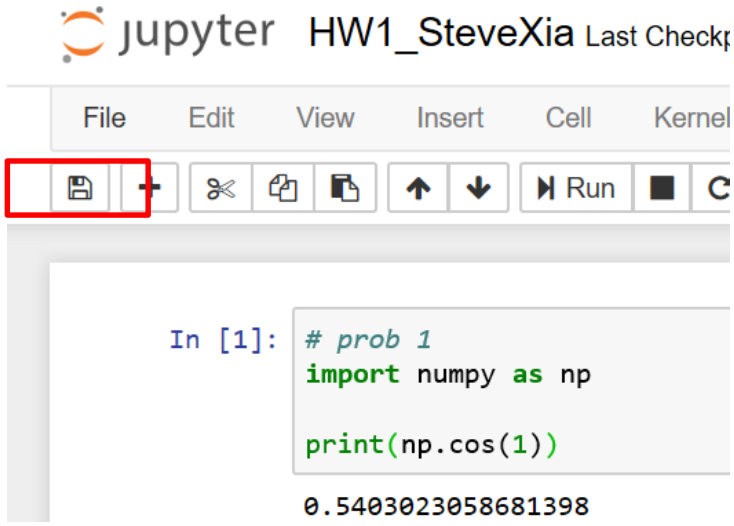
Jupyter Notebook Introduction

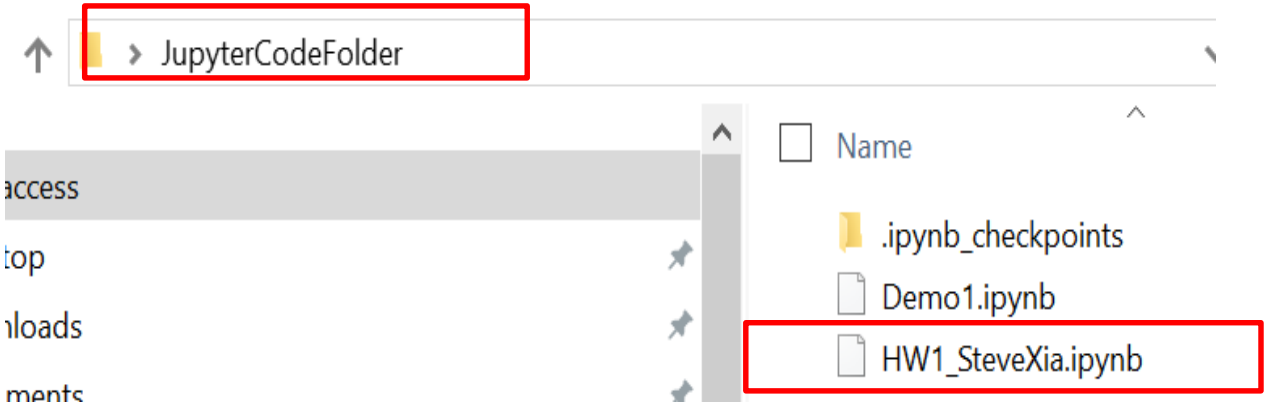
- Insert a cell below and copy the second portion of the code (e.g., hw1 problem 2) from your Spyder here as the second cell, then repeat the process until you are done with copying all of your homework solutions here. Click run to show results for code in each cell



Jupyter Notebook Introduction

- Click save and you should find the file HW1_SteveXia.ipynb at the folder where you started the jupyter session. Submit this file to the TA





Python Packages

- Python's power comes from a lot of the 'specialized' packages you can download to work with the main program
  - Numpy — for matrix and numerical operations
  - Pandas — for data analysis
  - SciPy — for scientific calculations such as integration
  - scikit-learn —for machine learning (sponsored by Google)
- A package is just a directory containing
  - files with Python code — called modules in Python speak
  - possibly some compiled code that can be accessed by Python (e.g., functions compiled from C or FORTRAN code)
  - a file called __init__.py that specifies what will be executed when we type import package_name (e.g., import numpy)
- use the import command to load the package into Python
  - import numpy as np # Load the numpy package and name it np
  - b = np.cos(a) # calling the cosine function contained in the numpy (np) package

Python Packages

- To find what packages you have installed with Python:
  - Type in help("modules") in your Spyder console

```
In [1]: help("modules")
```

```
Crypto              cProfile            mistune             spyder_io_dcm
Cython              cachecontrol        mkl                 spyder_io_hdf5
IPython             calendar            mmap                spyder_profiler
OleFileIO_PL        certifi             mmapfile            spyder_pylint
OpenSSL             cffi                mmsystem            sqlalchemy
PIL                 cgi                 modulefinder        sqlite3
PyQt5               cgitb               mpmath              sre_compile
__future__          chardet             msgpack             sre_constants
_ast                chunk               msilib              sre_parse
_asyncio            click               msvcrt              ssl
_bisect             cloudpickle         multipledispatch    sspi
_blake2             clyent              multiprocessing     sspicon
_bootlocale         cmath               navigator_updater   stat
_bz2                cmd                 nbconvert           statistics
_cffi_backend       code                nbformat            statsmodels
_codecs             codecs              netbios             storemagic
_codecs_cn          codeop              netrc               string
_codecs_hk          collections         networkx            stringprep
_codecs_iso2022     colorama            nltk                struct
_codecs_jp          colorsys            nntplib             subprocess
_codecs_kr          commctrl            nose                sunau
_codecs_tw          compileall          notebook            symbol
_collections        comtypes            nt                  sympy
_collections_abc    concurrent          ntpath              sympyprinting
_compat_pickle      conda               ntsecuritycon       symtable
_compression        conda_build         nturl2path          sys
_csv                conda_env           numba               sysconfig
_ctypes             conda_verify        numbers             tables
_ctypes_test        configparser        numexpr             tabnanny
_datetime           contextlib          numpy               tarfile
```

- you can find and explore the directory for NumPy on your computer easily - C:\ProgramData\Anaconda3\pkgs\numpy

Python Packages - SciPy

- The SciPy library is built on top of NumPy and provides additional functionality
- SciPy includes many of the standard routines used in
  - linear algebra
  - integration
  - interpolation
  - optimization
  - distributions and random number generation
  - signal processing
- For example, let's calculate $\int_{-2}^{2} \phi(z)dz$ where ϕ is the standard normal density:

```
from scipy.stats import norm
from scipy.integrate import quad

fai = norm()
value, error = quad(fai.pdf, -2, 2)  # Integrate using Gaussian quadrature
```

Find what version of Packages you have & update

- import scipy

- scipy.__version__
  Out[30]: '1.1.0'

- To update your packages:
  - In Anaconda prompt, type in: "*conda update scipy*"

Python Packages - NumPy

- NumPy is the fundamental package for scientific computing with Python.
- It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities
- For example, create simple matrix using Numpy:

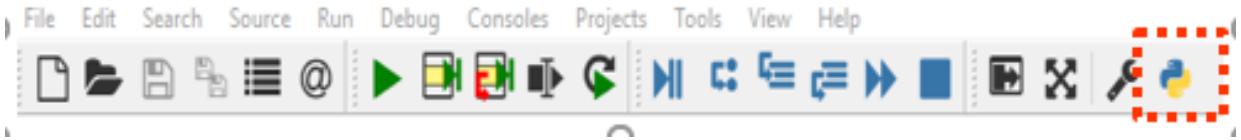
```
import numpy as np
# create a 3x3 matrix
X = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# use built in functions to initiate matrices
# create a matrix of zeros with three rows and two columns
Y = np.zeros((3,2))
# create a matrix of ones with three rows and two columns
Y1 = np.ones((3,2))
```

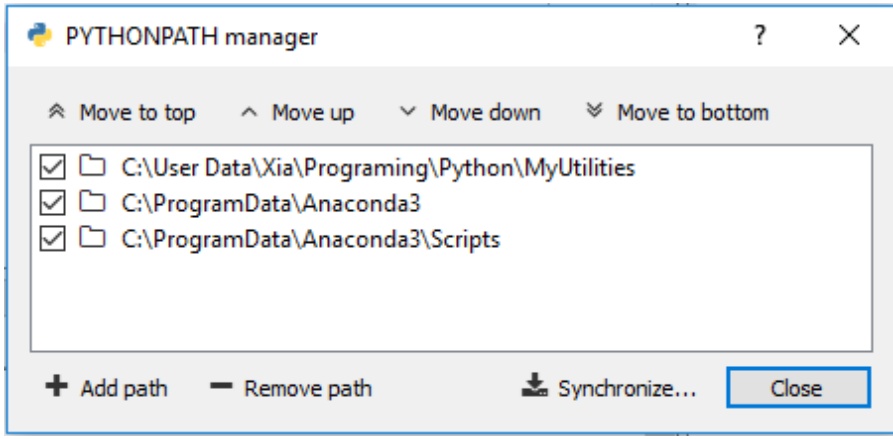
A few basics

- Python is a case-sensitive language
- How to add comment lines: use the # symbol
- How to comment out a whole block of code: begin with """, and end the block with """
- Line break: Simply type enter to break a long line into multiple lines
- del X, Y – clear variable X and Y from the computer memory
- Click the ‘x’ button under the variable explorer window in Spyder to clear all the variables in memory
- Type in %clear to clear the Ipython console or ‘control L’ from the keyboard

Setup: Defining file/function search path



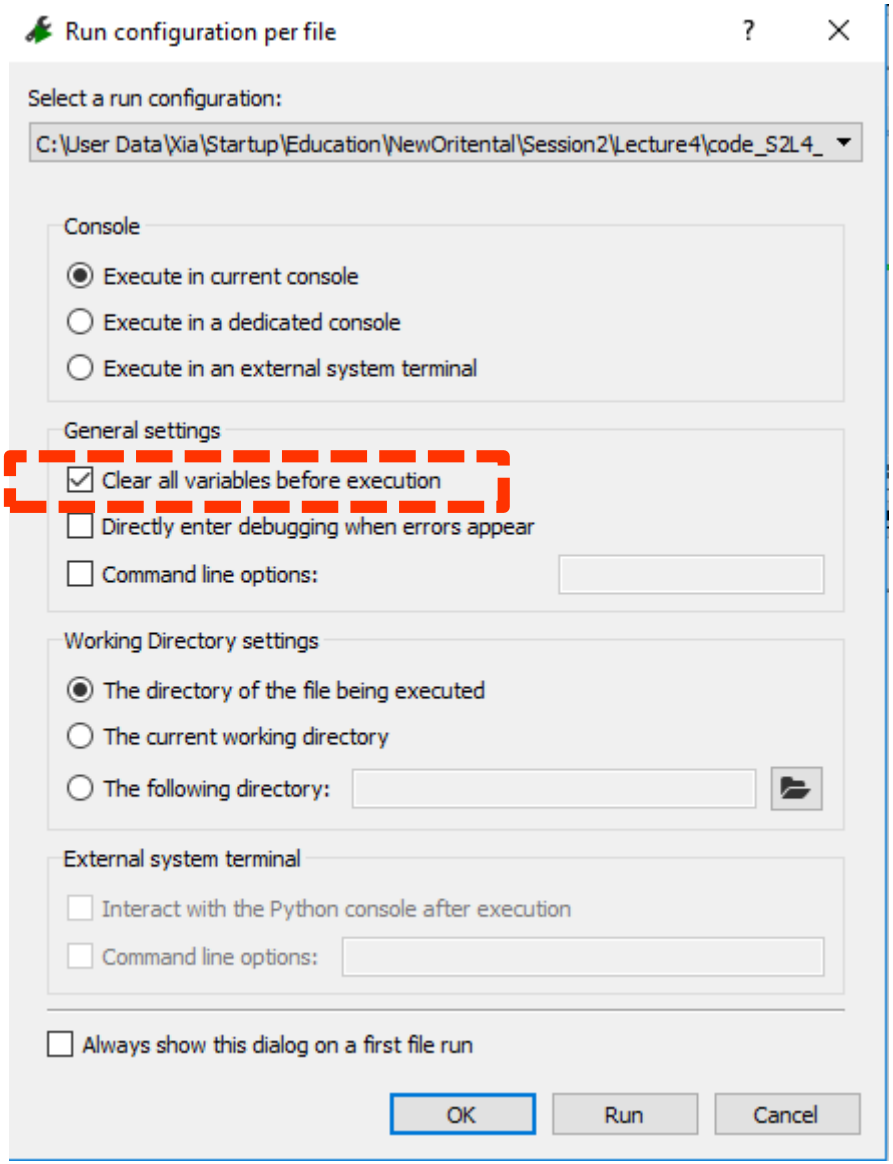
- Where does Python search for Python script files or more importantly, Python functions?
  - It always start with the current working folder (how do you find out what is the current working folder?)
  - Then the path defined in the path setup
  - Functions in packages loaded in using the import call is automatically available



- The way to change the path file:
  1. In Spyder, click the Python path manager icon and add the folder you want to the list of folders
  2. In a python script:
     import sys
     sys.path.append("C:/User Data/Xia/Programing/Python/MyUtilities")

Setup: Clear all variables before running a python script

- You might want to automatically clear all the variables in Python before you run a script.
- Easiest way to do this is to change the Spyder setting
- going to the Spyder menu > Run > Configuration per file > General settings and selecting the option called Clear all variables before execution



Python Basics – function call and object method

- Python's powerful in that there are a lot of computing algorithms are built-in through available functions and methods associated with a class coming with the software
- Basic form of a function call:
  - A, B = funcname(X,Y)
  - A and B are the output variables; X and Y are the input variables
- Basic form a method call associated with an class/object

```
>> X = np.array([1, 2, 3])
>> X.mean
```

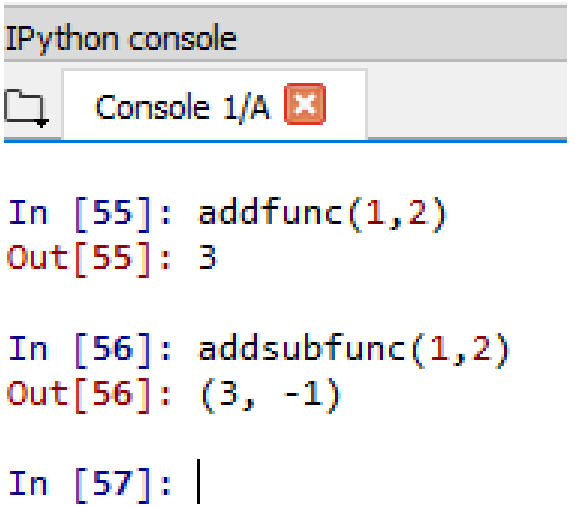
Example: Creating a function and calling a function

- Creating a function and calling a function

```
def addfunc(x,y):
    return x+y

def addsubfunc(x,y):
    return x+y, x-y

a=addfunc(1,2)
c,d = addsubfunc(1,2)
```



- One thing to remember:
  - Location of the function, need to either at the top of the script, or, use the import command to import from packages or saved python files where the functions are defined

Example: Creating a class and calling methods associated with the class

- Creating a class called Point and define a method print associated with it
- Then create an object belongs to the class Point and calls the method print

```
# defining a class called Point, with two methods, assign and print
class Point:
    def assign(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def print(self):
        print(self.x, self.y, self.z)

# define a point object
P1 = Point()
P1.assign(1,2,3)
P1.print()
```

- One thing to remember:
  - Location of the class definition, need to either at the top of the script, or,
  - use the import command to import from packages or saved python files where the classes are defined

Python: Lambda Functions

- Python supports the creation of anonymous functions (i.e. functions that are not bound to a name) at runtime, using a construct called "lambda":
- Here the function created by lambda is said to be anonymous, because it was never given a name. This is a very nice feature for applications where we don't want to create a separate named stand-alone function
- The benefit is that you have the flexibility to change the content of the function each time you use it.
- a) A code shows the difference between a normal function definition (f) and a Lambda function definition, and b) how it is used in our optimization code is shown below

quick and dirty way

```
>>> def f (x): return x**2
...
>>> print f(8)
64
>>>
>>> g = lambda x: x**2
>>>
>>> print g(8)
64
```

Modules: importing user created functions

- A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended
- Modules are a good way to organize user defined functions

WriteFiles.py | L3_DownloadData.py | L2_PlottingScatter.py | L1_ModuleTest.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar  7 11:29:49 2018

@author: Steve Xia
"""
import AddNPointModule as AP

a = AP.addfunc(1,2)
c,d = AP.addsubfunc(1,2)

# define a point object
P1 = AP.Point()
P1.assign(1,2,3)
P1.print()
```

nloadData.py | L2_PlottingScatter.py | L1_ModuleTest.py | AddNPointModule.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 23 15:42:24 2017

@author: Steve Xia
"""

def addfunc(x,y):
    return x+y

def addsubfunc(x,y):
    return x+y, x-y

# defining a class called Point, with two methods, assign and print
class Point:
    def assign(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def print(self):
        print(self.x, self.y, self.z)
```

Matrix Manipulation - basics

- Use the numpy package to help construct and manipulate matrices
  - import numpy as np
- Use ‘,’ to add new columns or new row elements.
  - X=np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
- Or use sequencing: np.linspace(2, 4, 5) to create an 1d array
- Matrix indexing – starts at zero, not one
  - X[0,2] – the first row, third column element of matrix X
  - X[0,:] – the first row of the matrix X
  - X[:,1] – the second column of the matrix X
  - b1=X[1:3,0:2]– the 2nd and 3rd rows and 1st and 2nd column of the matrix X
  - Note: the python indexing convention is that 0:N means index of 1,2,…**N-1(not N)**
  - X[0,-1] – first row, last column element of X. Index -1 can be used to reference the last element of an array

Matrix Initiation using built in functions

- Create a matrix of zeros with three rows and two columns
  - *Y = np.zeros((3,2))*
- create a matrix of ones with three rows and two columns
  - *Y1 = np.ones((3,2))*
- create a matrix of nan with one rows and three columns
  - *Y3 = np.full((1,3), np.nan)*

Matrix Manipulation - more

- How to append a matrix to the right, or below another matrix

```
>> X = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>> Z = np.ones((1,3))
```

X = 1 2 3 Z = 1 1 1
4 5 6
7 8 9

>>Z1 = **np.vstack**((X,Z))

Z1 = 1 2 3
4 5 6
7 8 9
1 1 1

>>Z2 = **np.hstack**((X, np.transpose(Z)))

Z1 = 1 2 3 1
4 5 6 1
7 8 9 1

- How to take out a row/column of a matrix

>>

Z1 = 1 2 3
4 5 6
7 8 9
1 1 1

- Take out the first row of Z1

```
>> W = np.delete(Z1, (0), axis=0)
```

W =
4 5 6
7 8 9
1 1 1

- Delete the second column of Z1

```
>> W1 = np.delete(Z1, (1), axis=1)
```

Z1 = 1 3
4 6
7 9
1 1

Matrix Operation & Boolean Indexing

- The algebraic operators +, -, *, / and ** all act element-wise on arrays
  - a = np.array([1, 2, 3, 4]) b = np.array([5, 6, 7, 8])
  - a + b -> array([6, 8, 10, 12])
- Sample matrix multiplication
  - A = np.ones((2, 2)) B = np.ones((2, 2))
  - A @ B -> [2., 2.
    2., 2.]
  - Or np.dot(A,B) gives the same results
- Using Boolean outputs to select only a portion of a matrix
- Example 1 – selecting only elements that are greater than 3
  - Z2[Z2 > 3]
- Example 2 - select only the rows of y where the first element of each row is greater than 0
  - # y = np.arange(35).reshape(5,7)
  - # y1 = y[y[:,0]>0]

Logical and Comparison Operations

- Boolean data type – True or False
- <, <=, >, >=, ==, != are operators that compare the values of two objects and returns True of False

  >> X=2

  >> X!= True
- 'and' for logical and; 'or' for logical or; 'not' for logical not

operators	descriptions
not x	Returns True if x is True, False otherwise
x and y	Returns x if x is False, y otherwise
x or y	Returns y if x is False, x otherwise

- '&' for bitwise and; '|' for bitwise or; '~' for bitwise not
- Membership operations
  - 'in' evaluates to True if it finds a variable in a specified sequence and False otherwise. 'not in' does the opposite
  - >> 'good' in 'this is a great show' → False
  - >> 'good' not in 'this is a great show' → True

Basic Control Structures

- For loops

```
for x in range(3, 8, 2):
    print(x) # Prints out 3,5,7
    y=x+1
    print(y) # Prints out 4, 6, 8
```

- While loops

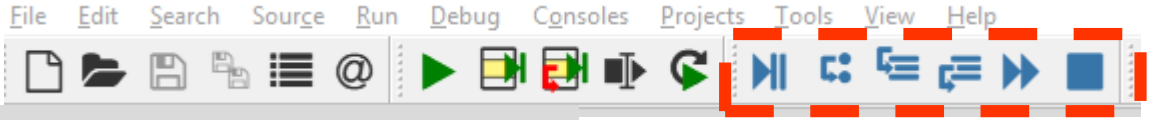
```
# Prints out 0,1,2,3,4
count = 0
while count < 5:
    print(count)
    count += 1
```

- If statements

```
 var1 = 100
if var1>100:
   print("Var Value {0:8.4f} is greater than 100".format(var1))
else:
   print("Var Value {0:8.4f} is equal or less than 100".format(var1))
```

Debugging - Example

- The Spyder IDE has ok debugging capability
- The key basic debugging function is done by setting break points in your code so you can step through certain parts of your code to find out what is happening



Editor

py | L2_DebugDemo.py | L2_matrixNIndexing.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Sep  5 23:38:00 2018
4
5  @author: Steve Xia
6  """
7  import numpy as np
8  def demo(x):
9      for i in range(5):
10         print("i={}, x={}".format(i, x))
11         x = x + 1
12     return x
13 a = np.ones((3,1))
14 a1=3
15 b=demo(0)
16 c=2
```

Python – how to get help information

1. Highlight the function you need help on, then press 'control+ i' in Spyder Editor

```
# read in data from
df3 = pd.read_excel(
```

read_excel

Definition : read_excel(*args, **kwargs)
Type : Function of pandas.io.excel module

Read an Excel table into a pandas DataFrame

Parameters

io : string, path object (pathlib.Path or py._path.lc
file-like object, pandas ExcelFile, or xlrd worl

2. Use the help() function from IPython console

```
In [14]: help(pd.read_excel)
Help on function read_excel in module pandas.io.excel:

read_excel(io, sheet_name=0, header=0, names=None, inde
squeeze=False, dtype=None, engine=None, converters=None
false_values=None, skiprows=None, nrows=None, na_values
date_parser=None, thousands=None, comment=None, skipfoo
    Read an Excel table into a pandas DataFrame

    Parameters
    ----------
    io : string, path object (pathlib.Path or py._path.
        file-like object, pandas ExcelFile, or xlrd wor
        The string could be a URL. Valid URL schemes in
        and file. For file URLs, a host is expected. Fo
```

Python – how to get help information

3. Just search the internet. You will find a lot of answers. This is my favorite way.
4. To see content of a method associated predefined classes: ??np.mean() or ??X.mean() with X a numpy array

Lcoation of souce code: c:\programdata\anaconda3\lib\site-packages\numpy\core\fromnumeric.py

```
In [15]: ??np.mean()
Signature: np.mean(a, axis=None, dtype=None, out=None, keepdims=<class 'numpy._globals._NoValue'>
Source:
def mean(a, axis=None, dtype=None, out=None, keepdims=np._NoValue):
    """
    Compute the arithmetic mean along the specified axis.

    Returns the average of the array elements.  The average is taken over
    the flattened array by default, otherwise over the specified axis.
    `float64` intermediate and return values are used for integer inputs.

    Parameters
    ----------
    a : array_like
        Array containing numbers whose mean is desired. If `a` is not an
        array, a conversion is attempted.
    axis : None or int or tuple of ints, optional
        Axis or axes along which the means are computed. The default is to
        compute the mean of the flattened array.

        .. versionadded:: 1.7.0

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

File:      c:\programdata\anaconda3\lib\site-packages\numpy\core\fromnumeric.py
Type:      function
```

Homework Assignment

See separate assignment doc on Latte – assignment #1

Open and save data

- Read from a csv file

  df = pd.read_csv('InputDataExample.csv', index_col=0, parse_dates=True)

- Save a dataframe to a csv file

  df.to_csv('OutputDataExample2.csv')

- Read from from an xlsx file

  df3 = pd.read_excel('InputDataExampleXlsx.xlsx', index_col=0, sheetname='Sheet1')

- Save to an xlsx file

  writer = pd.ExcelWriter('OutputDataExampleXlsx.xlsx', engine='xlsxwriter')
  # Write each dataframe to a different worksheet.
  df3.to_excel(writer, sheet_name='OriginalData', startrow=1, startcol=0, header=True, index=True)

- Open Matlab data file

  import scipy.io as spio
  mat = spio.loadmat('EF_MonteCarlo_InputData.mat', squeeze_me=True)
  stock = mat['port']

Open and save data

- Open and write a text file

```
file1 = open("myfile.txt","w")
L = ["This is Delhi \n","This is Paris \n","This is London \n"]

# \n is placed to indicate EOL (End of Line)
file1.write("Hello \n")
file1.writelines(L)
file1.close() #to change file access modes

file1 = open("myfile.txt","r+")
content_1st = file1.readline()
content_2nd = file1.readline()
content_rest = file1.readlines()
```

Python Graphics – line plots and bar charts

```
# line plots
figure_count = 1
plt.figure(figure_count)

plt.plot(ret1)
plt.ylabel('R(t)')

# bar chart
figure_count = figure_count+1
plt.figure(figure_count)
plt.hist( ret1, normed=True, bins=50,
histtype='stepfilled', alpha=0.5, label='ret')
plt.legend(loc='upper left',  bbox_to_anchor=(0.05,
0.9), shadow=True, ncol=1)
xmin, xmax = -0.1, 0.1
plt.xlim( (xmin, xmax) )
plt.xlabel('returns')
plt.ylabel('frequency')
```

Plotting methods associated with Series/Dataframes

- Example: simple line plot of all data in a dataframe/Series

```
Ret_Dow.plot(kind='hist', bins=bins, normed=True,
             alpha=0.5, color='blue')
```

- Example: areaplot of data in a dataframe

```
pw = pd.DataFrame(portfolios)
pw.columns = df.columns.values
pw.index = returns
pw.plot.area()
```

Useful Python Charting Resources

https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python

https://python-graph-gallery.com/

https://matplotlib.org/users/pyplot_tutorial.html

https://plot.ly/python/

Extra Resources for Learning Python

- https://www.w3schools.com/python/default.asp
- https://developers.google.com/edu/python/introduction
- https://docs.python.org/3/tutorial/index.html
- https://lectures.quantecon.org/py/index_learning_python.html
- https://pandas.pydata.org/pandas-docs/stable/dsintro.html
- http://www.scipy-lectures.org/intro/numpy/numpy.html

- Book
  - Python for Data Analysis, Wes McKinney, O'Reilly Media, Inc., 2018.