CISC 5352 FINANCIAL DATA ANALYTICS LECTURE NOTE (1)

Henry Han Ph.D.
Department of Computer and Information Science
Fordham University, New York NY 10023

The goal of this class

Foster and enhance students' data analytics and software development capabilities in finance.

What makes financial/business data analytics so popular in the market?

High-frequency trading (HFT) is changing the whole business world

HFT: A computerized trading: unlike traditional trading, HFT relies on **a high-frequency trading system** to trade automatically.

High-frequency trading (HFT) is changing the whole business world

HFT: A computerized trading: unlike traditional trading, HFT relies on **a high-frequency trading system** to trade automatically.

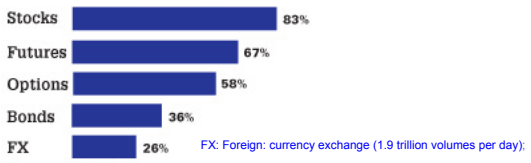
The high-frequency trading system is actually a financial information system that conducts trading in a **high-frequency** mode.

A transaction can be done in **a few milliseconds!**

It uses computer algorithms to do trading in a high-frequency mode automatically

It has started to dominate finance industry since 2000

2008: 71.3% trading in finance done via HFT
2010: ~75% trading finance done via HFT
2016: >75% trading done via HFT



FX: Foreign: currency exchange (1.9 trillion volumes per day);

Asset classes traded by HFT in 2010

HFT belongs to algorithmic trading (algo-trading)

Algo-trading : use computer algorithms to do trading: trading is automated by computer programs.

It makes trading more rigorous/systematic and faster by removing human related factors in trading (e.g. special preference for some assets).

It increases financial market liquidity by making assets to be bought and sold (at their stable prices) more quickly.

HFT: high-frequency algo-trading

What are possible risks brought by HFT to market?

It greatly increases trading risks for its nature of high-frequency trading

It wiped 998.5 points (9.2%) off the DOW in 7 minutes on May 6 2010

Officially called 2010 Flash Crash: started 2:41 pm and finished 2:47 pm. At 3:07 pm market regained 600 points).

It greatly increases trading risks for its nature of high-frequency trading

HFTs traded over 27,000 contracts in the 14 seconds (2:45:13 pm and 2:45:27 pm): 49% of whole trading volume on that day!

It also wiped 3% off S&P 500 in 4 minutes on that day (2:41 pm-2:44 pm)

It was blamed as one of reasons for 2008 financial crisis!

It challenges business analytics in an unprecedented way

HFT makes market tendency, security prices and investment return more unpredictable in a long run.

HFT brings a huge amount of data or even big data to market!

The availability of the large amount of data requires business analytics to move from **model-driven analytics to data-driven analytics.**

Model-driven analytics

- We only have relatively small data available
- Data analytics relies on traditional mathematical/ statistical models, which usually have theoretical assumptions about market (e.g. BSM model).
- Decision making is based on the theoretical models
  - Use limited data to fit models
  - Models' prediction capability is relatively low
  - No serious programming skills needed

Model-driven analytics: Black–Scholes (BS) model for option pricing

- **BS model is a model published in 1973 for option pricing. Black and Scholes were awarded 1997 Nobel economics prize for this model!**
- **It assumes**
  - **The stock pays no dividends**
  - **stock returns are normally distributed**
  - **Interest rates are constant and always known**
  - **There are no transaction costs (no commission)..**
  - **Options are only European options**

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf = 0$$

- f(S,t) is a function to model the value of an option
- S: S: stock price (known)
- σ : implied volatility (unknown)
- t: interest rate (known)

Data-driven analytics: listen to the data!

- We have a large amount of data available
- Data analytics models are driven from a large amount of data instead of theory only
- Decision making is based on data
  - Use models to fit data: derive suitable models from data
  - Models are more specific and have better predictive capability
  - **Serious programming skills are essential**

How to answer the following most important trading questions via modern business analytics approaches?

- This stock (e.g. Google) is a less risky security or not?
- The stock will be risky in the future according to its option performance?

The first question can be answered by computing stock's volatility

Volatility can be viewed as the 'standard deviation' of a stock though its computing is more complicate

It is computed from known historical pricing data

The higher the volatility, the more risky the security.

It is usually between the range of 15% to 60%.

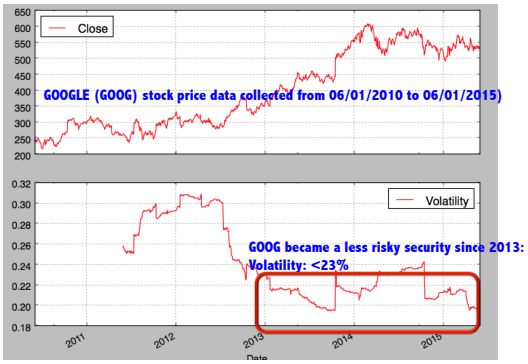
A stock with ~25% volatility→ less risky stock.

Is GOOG (GOOGLE stock) a risky security (suppose we asked this question on June 01, 2015)?

We collected all GOOG stock from June 01, 2010 to June 01 2015 to compute its volatility

---

Is GOOG (GOOGLE stock) a risky security?

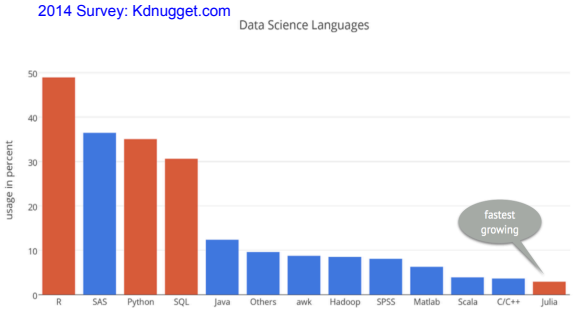


Data collection and analysis done by using Python's analytics package: Pandas by H Han, 2015

Which languages are good for business analytics?

R
Python
Hadoop/Spark
SQL
SAS
SPSS
C++

2014 Survey: Kdnugget.com



In Financial data analytics field

Python/R
C++
C#
Java
matlab

Which languages are good for business analytics?

R
Python (our focus)
Hadoop/Spark
SQL
SAS
SPSS
C++

We start/review python intensively

Our focus is to use python conduct **financial data analytics**

This class is definitely NOT 'only a python programming class'!

The structures of this class

① Component 1: python programming intensive (introduce python and basic knowledge in finance)

② Component 2: state-of-the-art financial models and high frequency trading (financial data analytics by python)

③ Component 3: advanced topics in high frequency trading (e.g. big financial data analytics)

Python: an interpreted language like R/ Matlab

1. **C/C++ syntax**
2. A major language in big data age
3. An OOP language
4. One course in Python makes you a capable (though not expert) programmer
5. Can use widely available packages and your new skill to solve problems (e.g. yahoo-finance: a python package to download yahoo finnace data).

PyPI - the Python Package Index: https://pypi.python.org/pypi

Python packages we will use

1. Numpy (Numerical Python)
2. pandas (**Python Data Analysis Library**)
   1. http://pandas.pydata.org/
3. matplotlib: (python plot library)
   1. http://matplotlib.org/
4. Ipython (Interactive computing environment)
5. SciPy (a numerical computing library)
6. Scipy+Numpy =Matlab

They are all contained in **Anaconda** package

https://www.continuum.io/downloads

ANACONDA
Powered by Continuum Analytics

Python is an *interpreted* language

Why?

① **interpreted means that Python looks at each instruction, one at a time, and turns that instruction into something that can be run.**

② **That means that you can simply open the Python interpreter and enter instructions one-at-a-time.**

③ **You can also *import* a program which causes the instructions in the program to be executed, as if you had typed them in.**

④ **To rerun an imported program you *reload* it.**

Python versions:

1. Almost all Linux systems have "built-in" python (default 2.7)
   1. Latest version: 2.7.12
2. So it is Mac OS (default 2.7)
3. You need to download python-3.52 (latest version)
4. https://www.python.org/downloads/release/python-352/

Warning: OS X El Capitan or other versions may take forever time to verify the package…

Under this case, you need to launch

/System/Library/CoreServices/Installer.app in finder

and run installer manually to finish the verification

Want to know your python version

Type

python –v

python3 –v

How to run Python under Windows?

Download python from https://www.python.org/

https://www.python.org/downloads/windows/

You can also download cygwin to do Python

Windows users should better download cygwin

Python 2 or Python 3

- Python 3 will take the market finally, though Python 2 is still in the Market.
- Python 2.7 is the highest version for python 2
- It seems to stop update
- Python 3.4/5 is a mature version for python 3
- We mainly focus on Python 3

print a (python 2) can't pass python 3!

print(a)!!! (python 3!)

We will introduce more differences between python 2 and 3

To use print function from Python 3 in Python 2, you need to input the following Command at the beginning of your python program

```
from __future__ import print_function
```

Division is different

3/2 is 1 in python 2
3/2 is 1.5 in python 3

To use division from Python 3 in Python 2, you need to input the following Command at the beginning of your python program

```
from __future__ import division
```

Python file's extension: .py

.py is a module (a python file: a set of python 'commands')

Each python program is called a module

IDE makes coding life easy

Python integrated development environments (IDE)

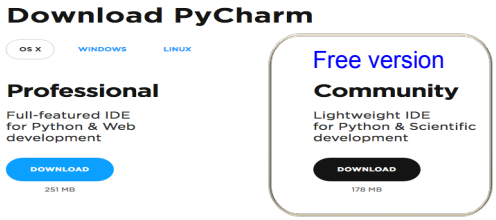
PyCharm

This is a commercial IDE!

Professional version has a good support for web development

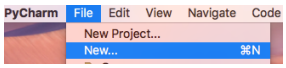
https://www.jetbrains.com/pycharm/download/#

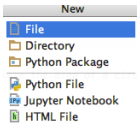
https://www.jetbrains.com/pycharm/download/

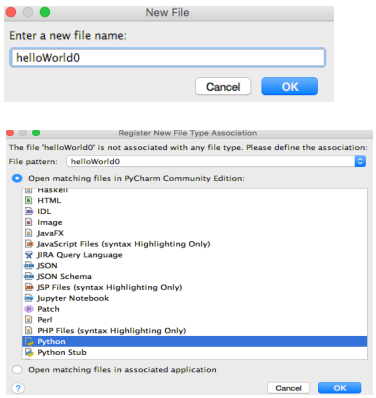


Get started with basic python with PyCharm IDE

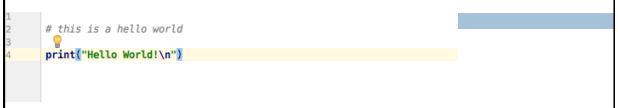
Suppose you want to write a helloWorld0.py and run it



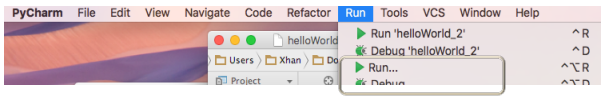




Start your coding…



Run…



Runing results

```
Hello World!

Process finished with exit code 0
```

Get started with basic python under linux (it works same in Mac OS)

In Mac OS, you need to get terminal at first
go→utilities→terminal

The easiest way

① Type python or python3
② print("Hello World\n")
③ Ctrl-D to move out of python

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more informatior
>>> print("Hello World\n")
Hello World

>>> 
```

Write a script file called helloworld0.py and run it.

Review basic linux commands in case you are not familiar with them…

① ls
② pwd
③ mkdir
④ cd
⑤ cd..(NEW)

Basic linux commands

1. pwd → display current working directory
2. ls → list items in the current directory
3. cd → change to a directory (folder)
   cd lab1 (go to directory "lab1")
4. cd..→ exit a directory (folder)
5. mkdir → create a directory

Basic linux commands

cp filename1 filename2 --- copies a file
rm filename --- removes a file

More: http://mally.stanford.edu/~sr/computing/basic-unix.html

Using emacs to write a python file hellowworld0.py

- Emacs mini-tutorial

emacs *filename*	
ctrl-x ctrl-c	exit emacs
ctrl-x ctrl-s	save current file
ctrl-v	scroll down one screen
ESC v	scroll up one screen

More information about emacs

http://doors.stanford.edu/~sr/computing/emacs.html

```
# This is my hello world
print("Hello World\n");
```

Save it as helloworld0.py

How to run it?

python helloWorld.py

```
$ python helloWorld0.py
Hello World
```

python2

```
Fordhams-MBP:Lecture Xhan$ python3 helloWorld0.py
Hello World
```

python3

All these linux-version steps work for Windows also!

You only need to download Cygwin

https://www.cygwin.com/

Using PyCHarm is recommended!

First Python

You can write Python and run it line by line
But it's good to write it as a module (.py)-- a program or a package of programs)

Modules are files that can be imported into your Python program.

Python's input: input()

Type the following codes in your interface

```
>>> x=input()
3.1415
>>> print(x)
```

What's x' type: float (real number) or a string?
Python 2 and 3 have different results!

In python 3, input function always returns a string: a type conversion is must

```
>>> x=float(input())
3.1415
>>> print(float(x))
```

 # a safer version
x=float(input())

Python's output: print

```
① print("Hellow World!\n")
② print(" Do you want to know how to calculate
  the area of a circle?\n")
③ radius   = float(input("Enter your radius\n"))
④ print(" You enter radius: ", radius)
```

Continuation: break a long line

python is sensitive to end of line stuff. To make a line continue, use the \

```
print("this is a test", \
" of continuation")
```

prints

```
this is a test of continuation
```

Continuation in R, matlab, C++

R: “,”

Matlab: “,…,”

C++: you can use “\” or just go to next line by hit Enter

Python comments (same as R)

- A comment begins with a # (pound sign)
- This means that from the # to the end of that line, nothing will be interpreted by Python.
- You can write information that will help the reader with the code
- Matlab comments : %
- C++//Java comments: //
- Ccomments: /* */

import of math module

import the math module with `import math`

if you want to use math functions

`math.pi`, for example, is pi.

`math.pow(x,y)` raises x to the yth power

math is class in Python

Code the following first python module

```
# this is my Hello World python!
# I am going to calculate something!

import math

print("Hellow World!\n")
print(" Do you want to know how to calculate the area of a
circle?\n")
radius   = float(input("Enter your radius\n"))
print(" You enter radius: ", radius)

area = math.pi*(radius**2)
print(" The area is:", area)
```

Python HelloWorld_2.py

Hellow World!

Do you want to know how to calculate the area of a circle?

Enter your radius

32324
(' You enter radius: ', 32324)
(' The area is:', 3282464734.371189)

```
Hellow World!

 Do you want to know how to calculate the area of a circle?

Enter your radius
23
(' You enter radius: ', 23)
(' The area is:', 1661.9025137490005)

Process finished with exit code 0
```

Lab assignment 1:

Write a python program to calculate the the circumference of a circle such that
1. input the radius
2. output the circumference

Keywords of python!

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	
class	exec	in	raise	
continue	finally	is	return	
def	for	lambda	try	

`elseif`→`elif`

They are prevented from using in a variable name

Variables: a word in a programming language

There are different types of variables → they have different storage in memory

Python variables names (same as C/C++)

- must begin with a letter or underscore _
  - A2
  - 2A (no!)

Python variables names (same as C/C++)

- must begin with a letter or underscore _
  - A2
  - 2A (no!)
- may contain letters, digits, and underscores
- may be of any length

Python variables names (same as C/C++)

- must begin with a letter or underscore _
  - A2
  - 2A (no!)
- may contain letters, digits, and underscores
- may be of any length
- upper and lower case letters are different
  - X is not x!

Python variables names (same as C/C++)

- must begin with a letter or underscore _
  - A2
  - 2A (no!)
- may contain letters, digits, and underscores
- may be of any length
- upper and lower case letters are different
  - X is not x!
- names starting with _ (underline) have special meaning. Be careful!

Python "primitive types"

1. integers: **5**
2. floats: **1.2: all real numbers**
3. booleans: **True/Flase**
4. strings: (C/C++ string): a sequence of characters
   1. a_str="this is a python string!"

Python "primitive types"

1. integers: **5**
2. floats: **1.2: all real numbers**
3. booleans: **True/Flase**
4. strings: (C/C++ string): a sequence of characters
   1. a_str="this is a python string!"
5. lists: [,] ['a',1,1.3] → cell in matlab, list in R
   1. It is an extension of array: an special array that include all types of elements
   2. rm(list=ls()) #Almost all R programming's beginning sentence
6. others we will see more …

Python is not a strongly typed language

You don't need to declare a variable type before you use it!

This is the key point between Python and other languages like C/C++/Java

Fundamental Types

- Integers
  - `1, -27 ( to +/- 2^64 - 1)`
  - `The range may rely on your computer systems (32-bit or 64-bit)`

How do we know the max integer/float you can have in python?

How do we know the max integer/float you can have in python?

import sys

sys.maxint

sys.float_info.max

9223372036854775807

1.7976931348623157e+308

Fundamental Types cont'd

- Floating Point (Real numbers)
  - `3.14, 10., .001, 3.14e-10, 0e0`
- Booleans (True or False values)
  - `True, False` note the capital

Converting types

You need to convert the value returned by the input command (characters) into an integer

`int("123")` yields the integer 123

It is not only important for old version python in Windows! Now it is required in python 3

Operators: glue words (variables) to expressions

Expressions are meaningful in python

An expression = variables + operators

In C++/Java an expression may not be meaningful!

- 2+3→ expression
- int x=m%2→ A statement (only statement is meaningful in C++/Java

Expression is meaningful in python! In C++ expression is not meaninful

Expression: x+2 ➔ return a value

Y=x+2→ a statement → describe an action → no 'return values'

Python Operators

Reserved operators in Python (expressions)

+	-	*	**	/	//	%
<<	>>	&	|	^	~	
<	>	<=	>=	==	!=	<>

= is assignment (not equal)

```
my_int = my_int + 7
lhs = rhs
my_var = 2 + 3 * 5
```

Operators

- Integer
  - addition and subtraction: `+, -`
  - multiplication: `*`
  - division
    - quotient: `/`
    - **integer quotient: //**
    - remainder: `%`
- Floating point
  - add, subtract, multiply, divide: `+, -, *, /`

Two types of divisions

The standard division operator (/) yields a floating point result no matter the type of its operands:

- `2/3` → yields `0.6666666666666666`
- `4.0/2` → yields `2.0`

- `2/3 (C++)` → `0 (also in python2)`

Two types of division, cont'd

Integer division (//) yields only the integer part of the divide (its type depends on its operands):

```
2//3     → 0
4.0//2   → 2.0
```

```
It means we don't need to do type
conversion as we did in C/C++
```

Mixed Types

What is the difference between `42` and `42.0` ?

- their types: the first is an integer, the second is a float

What happens when you mix types:

- done so no information is lost

```
42 * 3 → 126
42.0 * 3 → 126.0
```

When mixed types in calculation

The result will be the high version type (int + float→ float)

Order of operations and parentheses

- Precedence of *,/ over +,- is the same, but there precedents for other operators as well

Operator	Description
()	Parenthesis (grouping)
**	Exponentiation
+x, -x	Positive, Negative
*,/,%, //	Multiplication, Division, Remainder, Quotient
+,-	Addition, Subtraction

Don't need to remember the orders

Just use parentheses

Parentheses always takes precedence

math module

```
import math
print(math.pi)          # constant in math module
print(math.sin(1.0))    # a function in math
help(math.pow)          # help info on pow

math.log10(x)
math.sqrt(x)
math.ceil(x)            # ceilling function
math.floot(x)           # floor function
math.e
```

Lab 2 write a small financial calculator

Such that, principal, interest rate and number of years are entered from keyboard

We can assume we are using the yearly compounding model!

- **P0: present value (PV)→ current $**
- **P(t): $ after t years: future value (FV)**
  FV=PV(1+r)^t

Python's control structures

```
if boolean expression:
    expressions
```

The special is elseif→elif

```
if boolean expression:
    expressions

if boolean expression:
    expressions

else:
   expressions
```

```
if boolean expression1:
        suite1
elif boolean expression2:
        suite2
(as many elif's as you want)
else:
        suite_last
```

Run the following program: demoelif.py

```
# determine a letter grade from a percentage input

print(" This is a grade calculator \n");

percent_float = float(input("Please enter  your percentage (e.g. 20) ? "))
print("\n");
print(" You enter: ", float(percent_float))
print(" \n Your grade is as follows:\n")

if      percent_float >100:
      print("wow, too good to be true!")
if 90 <= percent_float <= 100:
   print("you received an A")

elif 80 <= percent_float < 90:
   print("you received a B")

elif 70 <= percent_float < 80:
   print("you received a C")

elif 60 <= percent_float < 70:
   print("you received a D")

elif percent_float<=60:
   print("oops, no that good")
```

Lab assignment 3: a detailed grade calculator

Rewrite the program such that

1. A+: >=95
2. A: >=90
3. A-: >=87
4. B+: >=85
5. B: >=80
6. B-: >=78

…

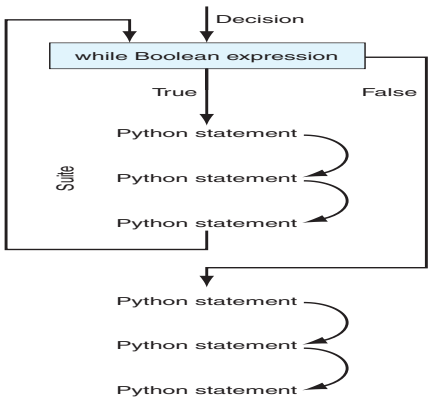
for –loop: for x in range(0, 10):

```
for x in range(0, 10):
print(x)
```

While-loop (while… else is a special case in Python)

- Top-tested loop (pretest)
  - test the boolean before running
  - test the boolean before each iteration of the loop

```
While boolean expression:
    suite
```




```
n=100
x  =1
while x < n:
  x = x + 1

print("Final value of x is: ", x)
```

```
n=100
sum  =0.0
x=1
while x < n:
  sum = sum + 1.0/x
  x=x+1

print("sum: ", sum)
```

Lab assignment 4: use while loop to compute

1. Sum of 1+2+3+….+10000
2. Mean of 1+2+3+….+10000
3. Sum of 1+1/2+1/3+….+1/10000