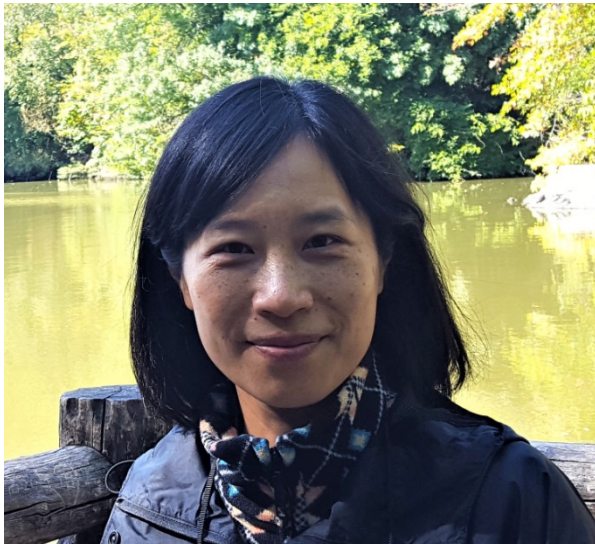


GR5260 Programming for Quant & Computational Finance

About the instructor



Ka Yi Ng, Ph.D.

22+ years of R&D experience in FinTech industry
specialized in Treasury and Capital Markets

- Financial engineering team at Calypso Technology
- Managed a dev team at ION Group, providing trading software to financial institutions and corporates
- Quant developer at Wall Street Systems
- Received Ph.D. in Mathematics at Columbia

Email: kyn@math.columbia.edu

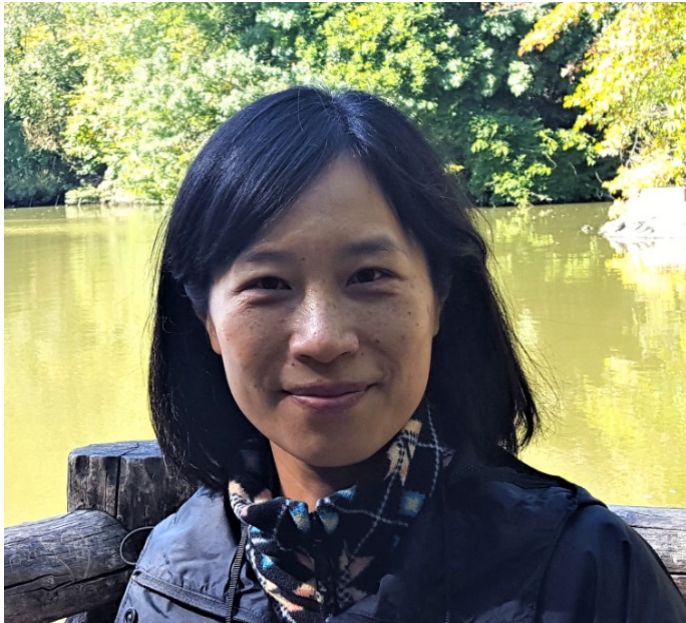
Office hours: by appointment

Web: www.math.columbia.edu/~kyn

<https://www.linkedin.com/in/kayi-ng/>

GR5260 Programming for Quant & Computational Finance

Instructor: Ka Yi Ng



Email: kyn@math.columbia.edu
Office hours: by appointment

TA: Ying Wu



Email: yw3521@columbia.edu
Office hours: TBA

About this course

- Theory and practices in quantitative finance
- Python in implementation and analysis
- Hands-on experience

Python 3.7+

Basic programming features

Data structures and types

Classes and objects

Error and exception handling

Data handling and visualization

Data I/O operations

Standard Python libraries:

numpy,pandas,scikit-learn,etc

Use Cases

Yield curve construction

Valuation of financial instruments

Monte Carlo simulation

Portfolio risk measurements

Historical volatilities

Value-at-Risk

Time series forecasting

and more...

Pre-requisites

Quantitative finance

GR5010 Introduction to Math Finance or equivalent

Mathematics

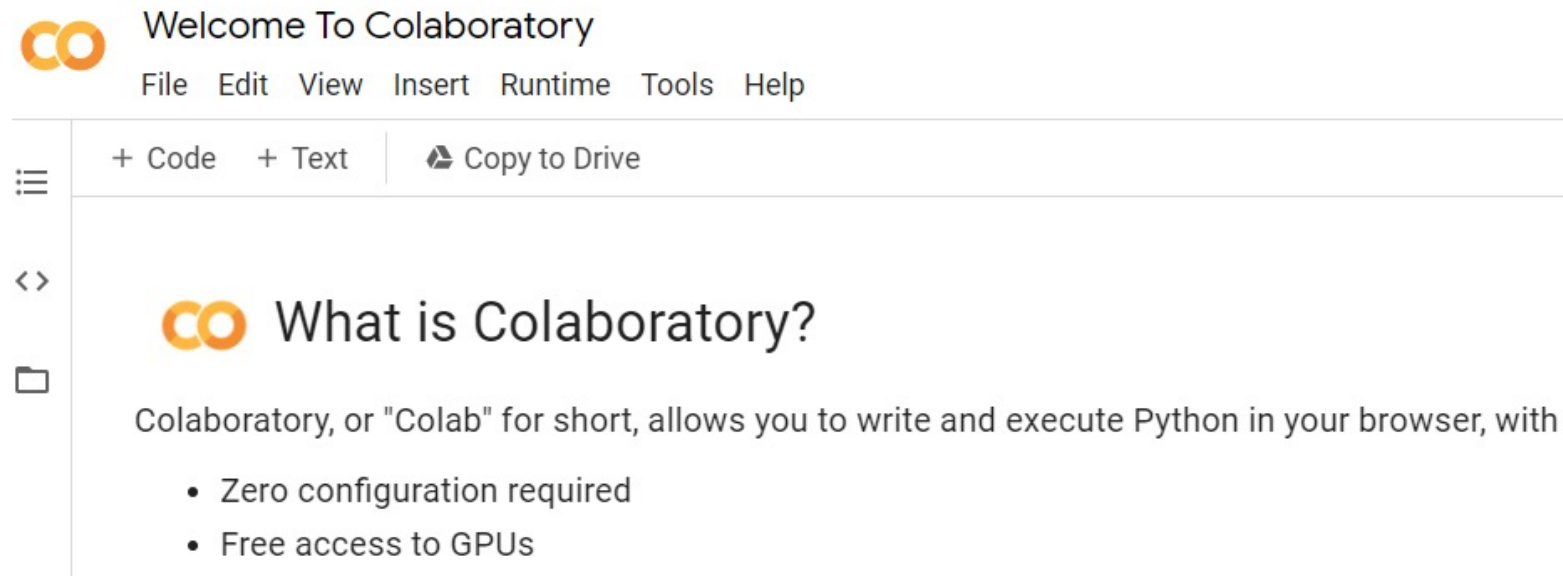
Linear algebra (matrices, eigenvalues and eigenvectors), Calculus (Taylor's expansion, integration and multivariable calculus), elementary analysis (e.g. limits), probability theory (distribution, conditional probabilities), stochastic calculus (e.g. Ito's lemma) and statistics (mean, variance, moment generating functions, linear regression)

Computing skills

Familiarity with programming concepts. Prior programming experience is preferred but not required

Technical Preparations

- Python installation
 - <https://www.anaconda.com/products/individual>
 - Jupyter Notebook, Spyder IDE
- Google Colab (main tool to be used in classes)
 - <https://colab.research.google.com/notebooks/intro.ipynb>



- Accessible via your Google Drive

References

Name	Author	Publisher
Python for Finance: Mastering Data-Driven Finance	Yves Hilpisch	O'Reilly Media
Options, Futures and Other Derivatives	John Hull	Prentice Hall
Monte Carlo Methods in Financial Engineering	Paul Glasserman	Springer
Machine Learning in Finance: From Theory to Practice	M. Dixon, I. Halperin, P.Bilokon	Springer

No standard textbook for this course

Course work & Grading scheme

Requirements	Percentage
Homework	40%
Exam 1	20%
Exam 2	30%
Class participation	10%

Homework

- 4-5 assignments in total
- Include theory and practice
- Submit homework via Courseworks by due date and time
- Late homework: no points

Written Exams

- Open book, in-class
- Theory (~30%) and practice (~70%)

Class participation

- Stimulating discussions and insights

Academic integrity in a virtual learning environment

Please be aware of the academic integrity for virtual learning environment guidelines (subject to standard penalties from Columbia University):

- **Class materials are intellectual properties of the instructor.** They must not be broadly shared (including on social media) without explicit consent of the instructor
- Contents and discussions of the class must not be circulated outside the classroom. Audio and Visual recording of the class (if available) must not be circulated outside the classroom. This is especially critical to protect the identity of speakers. Failure to do so could be **a form of bullying and endanger class participants**
- **Students are not permitted to record any portion of the class sessions without the explicit consent of the instructor**

<https://gsas.columbia.edu/student-guide/research/academic-integrity-and-responsible-conduct-research>

<http://bulletin.columbia.edu/columbia-college/standards/>

<https://studentconduct.columbia.edu/content/policies>

Students should be aware that academic dishonesty (for example, plagiarism, cheating on an examination, or dishonesty in dealing with a faculty member or other university official) or the threat of violence or harassment are particularly serious offenses and will be dealt with severely under Dean's Discipline.

- Be careful of “collaboration” on assignments
- Do your own work

Course policies

Homework

- If you have any questions about homework, please ask your TA in advance, eg. 24 hours before the due time
- Late homework: no points
- If students are caught copying, all participants would receive zero for that assignment

Exams

- Unless you have a good reason, make up means 20% off, and you need to get a TA to proctor

Emails

- Subject line starts with “GR5260”
- Email will typically be replied by the end of the next business day
- If you don't get a reply, please come to see me before the next class

Assignment: Practice

Homework 1: yield curve construction

For a given time t where $t_i \leq t < t_{i+1}$, the log-linearly interpolated discount factor for time t is:

$$DF(t) = \exp\{\ln(DF_i) + (t - t_i)(\ln(DF_{i+1}) - \ln(DF_i))/(t_{i+1} - t_i)\}$$

- a) Define a function to return the log-linearly interpolated discount factor for a given time t . For $t > t_n$, this function shall return the discount factor computed from the zero rate of the last discount factor DF_n in the curve. That is: $DF(t) = \exp(-zt)$ and $z = -\ln(DF_n)/t_n$

- i) Use the curve generated above to compute the daily forward rates for the next 2000 calendar days. Plot the computed daily forward rates. Comment on the shape of the curve and explain why this is the case.

Important dates

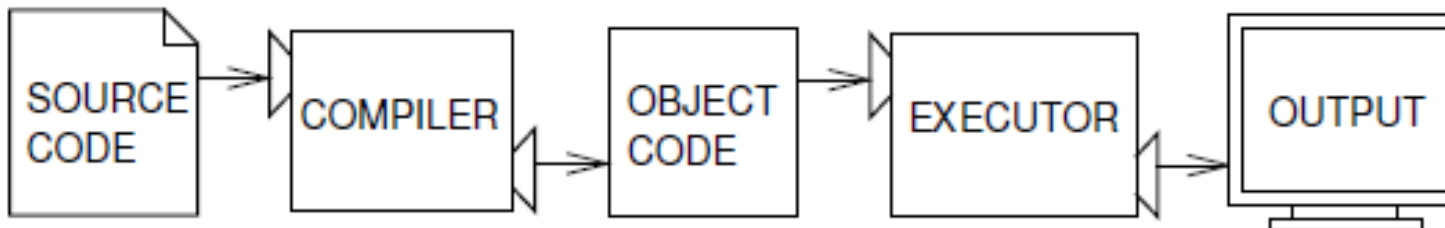
Event	Date
Change of program ends	Jan 27
Last day to drop class	Feb 21
Exam 1	Mar 10
Spring break	Mar 13 - 17
Exam 2	May 5

Python vs C++

- Python as an interpreted language
 - An interpreter reads the python program and executes it. It executes the program instructions as it reads them.



- C++ as a compiled language
 - A compiler first reads the entire program and translates it into a different form before running the compiled form. The compiled code can be run repeatedly without repeated translation



Installing Python, Jupyter and Spyder


<https://www.anaconda.com/products/individual>

Anaconda Installers

Windows 

Python 3.8

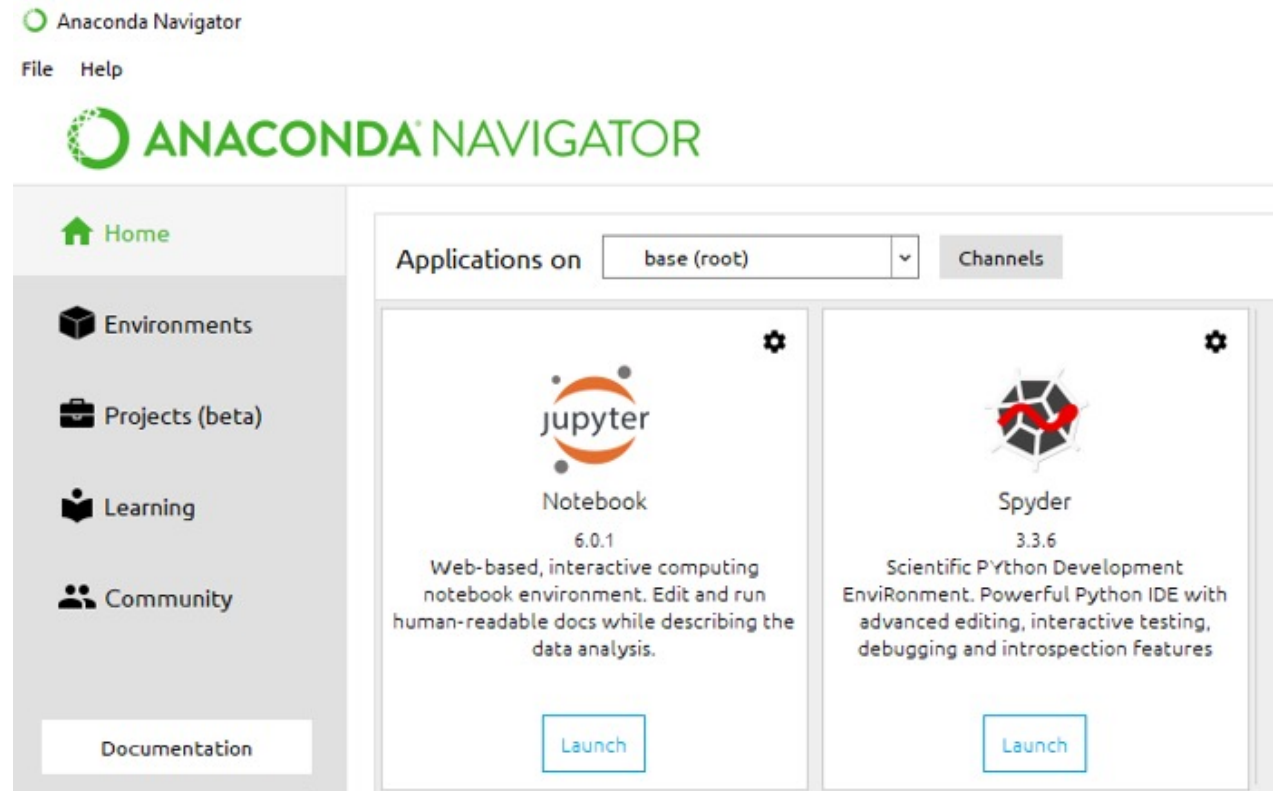
64-Bit Graphical Installer (457 MB)

MacOS 

Python 3.8

64-Bit Graphical Installer (435 MB)

- Python 3.7 or higher



The screenshot shows the Anaconda Navigator desktop application. At the top, there's a title bar with the Anaconda Navigator logo and menu items 'File' and 'Help'. Below the title bar is a sidebar with navigation options: 'Home' (selected), 'Environments', 'Projects (beta)', 'Learning', and 'Community'. At the bottom of the sidebar is a 'Documentation' button. The main area displays 'Applications on base (root)' with a dropdown menu and a 'Channels' button. Two application cards are visible: 'Jupyter Notebook' (version 6.0.1) and 'Spyder' (version 3.3.6). Each card includes a brief description and a 'Launch' button.

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Home

Environments

Projects (beta)

Learning

Community

Documentation

Applications on base (root) Channels

Jupyter Notebook 6.0.1

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch

Spyder 3.3.6

Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection Features

Launch