FRE6831
COMPUTATIONAL FINANCE LABORATORY (PYTHON)

Edward D. Weinberger, Ph.D., F.R.M
Adjunct Professor
Dept. of Finance and Risk Engineering
edw2026@nyu.edu
Office Hours by appointment
(6 46) 436-6174 (cell)

This half-semester course introduces the Python programming language . Interest in Python is growing faster than any other major programming language, according to a survey conducted by Stack Overflow, a widely consulted programming website. While there is much general interest due to Python's many extensions (NumPy, Pandas, SymPy, and a variety of AI tools, for example), Python is of particular interest in finance because a version of it is being used as the "glue" that holds together the computing infrastructure of several major financial institutions (Bank of America, JP Morgan Chase, and Goldman Sachs have working systems, and I'm told that Barclays is working on it.).

Course Overview and Goals:

Python is sufficiently quirky that a half-semester (six lecture + project) course must necessarily focus on the language itself, as opposed to specific financial applications. Python basics, just like the basics of English, are reasonably easy to learn, given the prerequisites below. However, just as there is a world of difference between a native English speaker and one that is merely fluent, there is a world of difference between mastering the basics and becoming a true Pythonista. The substantial project required by this course will get students started (but only started!) on becoming the latter.

Prerequisites:

Students will be expected to have fluency in an object-oriented language, such as C++, Java, or C#, as this course is intended to introduce students to object oriented Python, not programming in general, nor to the object-oriented paradigm in general.

Required text:

The Quick Python Book by Naomi Cedar, 3rd Ed., Manning, 2007, ISBN-10: 1617294039.

Recommended Reading

Just about everything there is to know about Python can be found somewhere on the web by Googling "Python <name of feature>". Often, the answers can be found on

`stackoverflow.com` or in the standard documentation maintained by the Python Software Foundation, https://docs.python.org/3/, which is surprisingly readable.

Course Materials and Resources

- **Access your course materials**: NYU Classes (nyu.edu/its/classes)
- **Databases, journal articles, and more**: Bern Dibner Library (library.nyu.edu)
  NYU Virtual Business Library (guides.nyu.edu/vbl)
- **Obtain 24/7 technology assistance:** Tandon IT Help Desk (soehelpdesk@nyu.edu, 646.997.3123)
  NYU IT Service Desk (AskIT@nyu.edu, 212-998-3333)

Course Requirements and Grading

To keep students engaged, I will occasionally ask them questions about such things as the likely output of a program, etc. This kind of class participation will be ungraded. Grades will be assigned based on the above-mentioned project, which is the preparation of a Python program that computes USD LIBOR discount factors and forward rates from text files containing the standard inputs, namely, rates on LIBOR cash deposits, Eurodollar futures prices, and USD LIBOR swap prices (More details provided during the first lecture and in a project description on the course website. Further details can be inferred from spreadsheets, posted on the website for this class, that implement samples of these calculations.). Lectures will include a discussion of how to build pieces of this program.

My overall philosophy of grading is that students who are able to demonstrate mastery of everything I taught them deserve an "A", partial mastery deserves a "B", and students who "aren't in the room" deserve a "C". Projects that cannot reproduce the calculations displayed by the posted spreadsheets are in the "not in the room" category. Projects that can not only reproduce these calculations but also pass additional tests that I do not post on the website earn higher grades, depending on how many they pass.

In the past, most students' projects are unable to pass all of the tests, as there are a considerable number of special cases to be dealt with.

Detailed Course Outline

Note: Placement of topics in lectures is only approximate

Lecture 1

Topic 1: Introduction to the Course and to Python

I. Course "Mechanics"
II. Observations on the FinTech eco-system
    a. 50 years of coding:

i. what has and what hasn't changed
ii. Pictorial Programming

b. A few insights from Computer Science 101
i. Interpreted vs compiled languages
ii. Objects
iii. $O(N)$ vs $O(\log N)$ vs $O(1)$ implementations
iv. Hashing
v. Sorting: an example of efficiency

c. Data, data, everywhere!
i. Input sources
ii. Databases
iii. Need to process disparate data elements

d. Industrial strength programming

e. Need well known language to interface with machine learning, symbolic calculation

III. How Python fits in

a. Why Python?
i. Elementary Python easy to learn, but also "expert friendly"
1. Addresses some annoying things in other languages
2. Can do a lot in a few lines
a. "batteries included" libraries
b. Expressive syntax, most notably lists and dictionaries
ii. Intended to be readable
iii. Free, but very well supported
iv. Many, many extensions (SciPy, NumPy, SymPy, AI libraries, etc.)
v. Multi-platform (no platform dependencies)
vi. Full support for object-oriented programming, including operator overloading, but without
1. explicit garbage collection
2. explicit pointers

b. Problems with Python (primarily because Python is interpreted)
i. Slower than C/C++

IV. Characteristics and Quirks of Python

a. Readability is key; hence indents used as block identifiers
b. Python 2.x vs Python 3.x (to be discussed more fully later on)
c. Python is interpreted, but …
d. Python uses "duck typing" and automated garbage collection
e. Everything is an object; object oriented programming fully supported
f. Lots of introspection

V. Installing Python

a. "Hello, world!"
b. Libraries

VI. The very beginnings

a. Numbers: `int float complex`
b. Strings
c. Booleans

d. None
e. Built-in functions: https://docs.python.org/3/library/functions.html
f. Dates via `datetime`

Reading: Cedar, Introduction and Chapters 1 – 4
Assignments: Write interpolation function and discount factor function for CD's

Topic 2: Lists, Tuples, and Sets

I. Lists
   a. "Declaring" a list
   b. Arrays, but with a twist!
      i. Length unspecified beforehand; entries added at end
      ii. List operators (append, indicies/slices, etc.)
      iii. List operations
      iv. Lists as queues and stacks
      v. Nested lists and deep copies
II. Tuples
   a. Mutability vs Immutability
   b. Declaration
   c. List-tuple conversion
   d. Packing/unpacking tuples
III. Sets
   a. Uniqueness of elements
   b. Set operations

Reading: Cedar, Chapter 5

Lecture 2

Topic 3: Strings

I. Strings as immutable sequences of characters, including special characters
II. `str` vs `repr`
III. String methods
   a. `split` and `join`
   b. Conversions
   c. Other string methods
IV. The many ways of formatting and printing strings
V. The `bytes` data type
VI. Unicode basics

Reading: Cedar, Chapter 6

Topic 4: Dictionaries

I. Review: Hashing
II. Definition as an associative array with immutable
III. Dictionary operations
IV. Some applications

Reading: Cedar, Chapter 7
Assignment: Trial implementation of discount factor storage

Lecture 3

Topic 5: Control Flow

I. Statements, blocks, and indentation
II. Boolean values and expressions
III. Standard stuff: `if` and `while`
IV. Loops over sets
    a. The `range` function
    b. `break` and `continue`
    c. tuple unpacking
    d. `enumerate` and `zip`
    e. list comprehensions
    f. generators

Reading: Cedar, Chapter 8

Topic 6: Python Functions

I. Definition and scoping
II. Function parameter options
III. Lambda expressions
IV. Functions assignment to "pointer" variables
V. Decorators
VI. Generator functions
VII. The `dir` function
VIII. Comments and doc strings

Reading: Cedar, Chapter 9

Topic 7: Input and output

I. Variants of the `print` statement
II. File objects
III. Reading command line parameters

Reading: Cedar, Chapter 13 (optionally Chapter 12)
Assignment: Reading market data from flat files

Lecture 4

Topic 8: Basics of Objects in Python

I. Basics of object definitions
    a. Attributes and methods
    b. The __init__() method
II. Member vs class variables
III. Static and class methods
IV. Inheritance
V. Private variables and methods

Reading: Cedar, Chapter 15 and 17

Topic 9: Modules

I. Setting up a module
II. Local and global variables
III. The `import` statement
IV. The `main` statement
V. Scoping rules

Reading: Cedar, Chapter 10
Assignment: Trial design of USDYieldCurve class

Lecture 5

Topic 9: More About Classes

I. Multiple inheritance
II. Operator overloading
III. Making a class callable
IV. Get/set attrib
V. @property

Reading: Cedar, Chapter 17

Topic 10: Exceptions

Reading: Cedar, Chapter 14

Topic 11: Regular expressions (if time permits)

Reading: Cedar, Chapter 16

Topic 12: NumPy

Reading: https://numpy.org/

Lecture 6

Various advanced topics, chosen from the following:

I. Pandas
II. Multi-threading
III. Unit testing and the mock library
IV. New features of Python 3.7
    a. Sorted dictionaries
    b. Ways of declaring variables with a given type
V. Functional programming
VI. Python and SQL databases
VII. SymPy
VIII. Other topics, to be determined

Reading: Cedar, chapters to be determined; other sources to be determined

Departmental/School-Wide Policies (Comments specific to the project **in bold**, below)

Academic Misconduct

A. Introduction: The School of Engineering encourages academic excellence in an environment that promotes honesty, integrity, and fairness, and students at the School of Engineering are expected to exhibit those qualities in their academic work. It is through the process of submitting their own work and receiving honest feedback on that work that students may progress academically. Any act of academic dishonesty is seen as an attack upon the School and will not be tolerated. Furthermore, those who breach the School's rules on academic integrity will be sanctioned under this Policy. Students are responsible for familiarizing themselves with the School's Policy on Academic Misconduct.
B. Definition: Academic dishonesty may include misrepresentation, deception, dishonesty, or any act of falsification committed by a student to influence a grade or other academic evaluation. Academic dishonesty also includes intentionally damaging the academic work of others or assisting other students in acts of dishonesty. Common examples of academically dishonest behavior include, but are not limited to, the following:
    1. Cheating: intentionally using or attempting to use unauthorized notes, books, electronic media, or electronic communications in an exam; talking with fellow students or looking

at another person's work during an exam; submitting work prepared in advance for an in-class examination; having someone take an exam for you or taking an exam for someone else; violating other rules governing the administration of examinations.

2. Fabrication: including but not limited to, falsifying experimental data and/or citations.
3. Plagiarism: Intentionally or knowingly representing the words or ideas of another as one's own in any academic exercise; failure to attribute direct quotations, paraphrases, or borrowed facts or information. **Submitting code that implements the same methodology as another student is not plagiarism; submitting the same code is plagiarism. It is surprisingly easy to tell the difference.**
4. Unauthorized collaboration: working together on work that was meant to be done individually. **You are encouraged to discuss the project with others. However, since I need to assign grades individually, I hold each of you individually responsible for the quality of the project you submit. Submitting code "borrowed" from another student that is not fully understood therefore runs two risks: First, that I will punish you for plagiarism, and, second, that you will not detect problems in the code because you don't understand it!**
5. Duplicating work: presenting for grading the same work for more than one project or in more than one class, unless express and prior permission have been received from the course instructor(s) or research adviser involved.
6. Forgery: altering any academic document, including, but not limited to, academic records, admissions materials, or medical excuses.

Disability Disclosure Statement

Academic accommodations are available for students with disabilities. Please contact the **Moses Center for Students with Disabilities** (212-998-4980 or mosescsd@nyu.edu) for further information. Students who are requesting academic accommodations are advised to reach out to the Moses Center as early as possible in the semester for assistance.

Inclusion Statement

The NYU Tandon School values an inclusive and equitable environment for all our students. I hope to foster a sense of community in this class and consider it a place where individuals of all backgrounds, beliefs, ethnicities, national origins, gender identities, sexual orientations, religious and political affiliations, and abilities will be treated with respect. It is my intent that all students' learning needs be addressed both in and out of class, and that the diversity that students bring to this class be viewed resource, strength and benefit. If this standard is not being upheld, please feel free to speak with me.

One of the ways that I try to maintain an equitable environment is by devising grading standards that are fair to all students. I therefore cannot arbitrarily raise a student's grade simply because failure to do so will "spoil their GPA" or cause them to lose a scholarship.