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

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 much of this interest is due to Python's many extensions (NumPy, 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.).

Python is sufficiently quirky that a half-semester course must necessarily focus on the language itself, as opposed to specific financial applications. A half-semester course must also choose one of the two incompatible versions of Python, *i.e.* Python 2 and Python 3; this course will focus on Python 3. Python basics, just like the basics of English, are reasonably easy to learn. 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. Hopefully, there will be time to present some of this advanced material towards the end of the course

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 Python, not programming in general, or to the object oriented paradigm in particular.

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.

Grading:

Grades will be assigned based on 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.). Lectures will include a discussion of how to build pieces of this program.

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

<u>Topic 2: Lists, Tuples, and Sets</u>

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

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