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:**


**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.).

**Detailed Course Outline**

Note: Placement of topics in lectures is only approximate

**Lecture 1**

**Topic 1: Introduction to the Course and to Python**

I. The class project  
   a. Statement  
   b. Discussion of what is required to do it  

II. Course “Mechanics”

III. Observations on the FinTech eco-system  
   a. Data, data, everywhere!  
      i. Input sources  
      ii. Databases  
      iii. Need to process disparate data elements  
   b. Industrial strength programming  
   c. Need well known language to interface with machine learning, symbolic calculation  

IV. 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  
      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++

V. 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. $O(N)$ vs $O(\log N)$ vs $O(1)$ implementations  
   e. Python uses “duck typing” and automated garbage collection
f. Everything is an object; object oriented programming fully supported

g. Lots of introspection

VI. Installing Python
   a. “Hello, world!”
   b. Libraries

Reading: Cedar, Introduction and Chapters 1 - 4

**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, indices/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

IV. Dates via `datetime`

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

*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: Modules**

I. Setting up a module  
II. Local and global variables  
III. The import statement  
IV. The main statement  
   V. Scoping rules
Lecture 4

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)

Topic 8: Exceptions

Reading: Cedar, Chapter 14

Topic 9: Regular Expressions

Reading: Cedar, Chapter 14

Lecture 5

Topic 10: Objects in Python

I. Basics of object definitions
II. Member vs class variables
III. Static and class methods
IV. Inheritance
V. Private variables and methods
   VI. Operator overloading

Reading: Cedar, Chapter 15 and 17

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. Other topics, to be determined

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