New York University
School of Professional Studies
Management and Information Technology

Advanced Python

INFO1-CE9980, Spring 2017
Note that this course focuses on Python 2.7 (3.x can be used as well)

Schedule: 10 Sessions
Section 1: Wednesdays 3/15 - 5/17 6:30 - 9:30PM
Section 2: Fridays 3/17 - 5/19 6:30 - 9:30PM

Instructor: David Blaikie dbb212@nyu.edu 646-469-9970

Course Website: http://www.davidbpython.com/advanced_python

Description: This course—for students who have successfully completed *Introduction to Python Programming/INFO1-CE9990* or the equivalent, and for programmers with a solid command of another programming language—is designed to expand your knowledge with an eye toward eventually acquiring a position as a junior developer. After a whirlwind review of language fundamentals, delve deeply into Python's powerful advanced features, such as user-defined classes, object-oriented design, decorators, and generators. Learn to employ the most widely used algorithms and libraries to solve common problems in the field. Gain a working familiarity with statistical analysis and visualization using Pandas, NumPy, and matplotlib. Write useful web applications using Flask and relational databases. Query and parse HTML, XML, and JSON data using urllib2 and Beautiful Soup, and perform advanced text processing using regular expressions. Learn to apply industry-standard tools and techniques for working within a development team, such as Git for versioning, code review, pydoc and pylint. Benchmark, profile, optimize, and test your programs, and code for memory efficiency. The course concludes with a discussion of common interview questions and pathways for gaining experience and eventually securing a position in the field.

Prerequisites: Introduction to Python Programming (INFO1-CE9990) or equivalent knowledge, or a solid grounding and working experience with another programming language like Java, C, C++, Perl, Ruby, etc. **This course moves *very* quickly through Python basics in the first three sessions.**

Rationale:
- master a language fast growing in popularity around the world and serving IT needs across numerous industries
- develop a deep understanding of the Python standard library, commonly employed techniques and algorithms, and coding for efficiency
- acquire working knowledge of Python for statistical analysis and visualization as well as web programming
- acquire team tools and techniques necessary to become a Junior Developer

Objectives: upon completion of this course, the student will:
- have a deep understanding of Python algorithms, efficiency and professional coding techniques
- have a solid familiarity with the Python standard library
- be able to use Python for data analysis and web development
- be prepared to do professional work in Python and consider employment as a Junior Developer

Structure:

- 10 sessions, 3 hr. lecture in a lab setting
- instructor available (and responsive) via email
- class slides and handouts available online
- homework to be submitted online

Course Requirements: The student is required to demonstrate:

1. familiarity with and ability to use Python language features, and
2. ability to produce professional-quality code - that is, applying proper syntax, usage and code organization

Each week, students are required to:

1. attend an in-class lecture
2. participate in class with questions and by responding to instructor queries
3. complete weekly programming assignments
4. review other students' work

Throughout the course, proper syntax and usage are emphasized in class and required in weekly assignments.

Grading: Please see NYU SCPS' grading and other policies at: http://www.scps.nyu.edu/academic-policies-and-procedures.html

- weekly programming assignments: 70% of grade
- effective/thoughtful code review 20% of grade
- attendance and class participation: 10% of grade

Grading Criteria: in addition to demonstrating the proper use of the code features highlighted in each assignment, you are also required to concientiously review other students' assignments to promote good coding practices among the group.

Deadlines: Homework solutions are due the evening before the next class session. If a due date cannot be met, it can be extended under certain circumstances. You must contact the instructor ahead of time to discuss a late submission.

Special Note on Incompletes:
*Under certain exceptional circumstances and subject to **PRIOR** approval by the Department a student may petition for a grade of "Incomplete." The granting of requests for a grade of Incomplete is not automatic. The student must have completed at least 50 percent of the coursework to be eligible for this grade.*
Special Note on Plagiarism:
New York University takes plagiarism very seriously and regards it as a form of fraud. The definition of plagiarism that has been adopted by the School of Continuing and Professional Studies is as follows: "Plagiarism is presenting someone else's work as though it were one's own. More specifically, plagiarism is to present as one's own words quoted without quotation marks from another writer; a paraphrased passage from another writer's work; or facts or ideas gathered, organized, and reported by someone else, orally and/or in writing. Since plagiarism is a matter of fact, not of the student's intention, it is crucial that acknowledgement of the sources be accurate and complete. Even where there is not a conscious intention to deceive, the failure to make appropriate acknowledgement constitutes plagiarism. Penalties for plagiarism range from failure for a paper or course to dismissal from the University."

Required Materials:
Online course handouts and weekly reading assignments.

Recommended Supplementary Reading:
Alex Martelli, Anna Martelli Ravenscroft and David Ascher, Python Cookbook, 3rd Edition.
Jeff Knupp, Writing Idiomatic Python 2.7.3.

Course Policies:

- Attendance in class is required.
- In cases of absence or extreme lateness the student must contact the instructor.
- All weekly work must be turned in before class meets.
- In some cases (work or personal issues) make-up work may be permitted to fulfill course requirements.
- All work must be original and completed by the student. Team submissions are not permitted. Online resources, whether from the course or outside the course, should be used for reference only and must not be submitted as part of homework assignments.

Course Syllabus:

1. **Objects, Files and Containers**
   a. Objects, Types, Functions, Methods, Operators, Files
      - objects and core data structures: variables, objects, operators
      - functions: arguments and return values: **raw_input(), int()**, **float()**, **str()**, **round(), len(), exit()**
      - debugging: problem solving and attention to detail and error messages
      - methods**: str.upper(), str.format(), str.isdigit(), str.isalpha(), str.format(), str.split(), str.rstrip(), str.splitlines()**
      - blocks and block structure: **if**, **elif** and **else, if not, and**, **or** **while**, **break** and **continue**
      - **str** slicing
      - **file** object, **file.readlines()**, **file.read()**
      - **list** objects, the **for** statement, subscripting, slicing
      - arguments to the program with **sys.argv**

   b. Containers: Lists, Tuples, Sets, Dictionaries
      - summarizing data with containers
        - containers **list** and **set**
        - summary functions **len()**, **in**, **sum()**, **min()**, **max()**
        - boolean objects: the "truth" is everywhere
        - looping through containers
      - introduction to sorting

2. **Dictionaries; Multidimensional Containers; Sorting; Sort Functions; Set Operations; List Comprehensions, Lambdas; OS and File Input/Output**
      - summarizing data with containers: **dict**
      - sorting containers, sort criteria function
      - reading and building complex structures
      - sorting multidimensional structures
      - lambdas for sorting
      - list comprehensions
      - **enumerate()** and **range()**
      - Exceptions
      - writing and appending to files, **with** context
      - **sys.stdin**, **sys.stdout**, **sys.stderr**
      - **sys.argv**
      - **os.listdir()**, **os.path.isfile(), os.path.isdir()**, **os.path.getsize()**
      - trapping Exceptions

3. **Functions and Modules; Classes**
      - functions

    - argument matching modes: positional, keyword, variable
    - function scoping and variable scopes
  - modules
  - instances, classes, instance and class namespaces
  - instance methods and attributes
  - encapsulation
  - object constructor
  - static methods and class methods
  - inheritance and polymorphism

4. **Advanced Classes**
  - data model
  - "magic" attributes
  - implementing operators and statements
  - subclassing builtins
  - iterator protocol
  - object inspection
  - variable naming conventions
  - "internal" types

5. **Decorators, Generators, Closures, Functional Programming**
  - attribute control
    - @properties
    - __getattr__, __setattribute__
  - descriptors
  - __slots__
  - generators
  - recursive functions
  - map(), filter(), reduce()

6. **Testing; Profiling and Benchmarking; Collections**
  - git
    - branching and merging
    - code review
  - testing with **pytest**
  - **pylint**, PEP8
  - benchmarking, profiling and optimizing
  - efficient collections

7. **Internet Programming and Databases**
  - **HTML**, **CSS** and **Bootstrap**
  - **Flask** lightweight web server
  - **Jinja2** Templates
  - **CGI, SimpleHTTPServer**, **CGIHTTPServer (Heroku Web Server)**
  - **sqlite3** and **SQL**
  - **object-relational mapping**
  - **smtplib**, **ftplib**
  - Remote APIs; RESTful interfaces

8. **Web Scraping and Regular Expressions**
   a. **urllib2**, **requests**

b. **Beautiful Soup, scrapy** (HTML parsing and scraping)
c. **Regular Expressions**

9. **Statistical Analysis and Visualization**
   - **pandas** and **numpy**
   - visualization with **matplotlib**

10. **Cracking the code interview; Test Questions; Contributing to Open Source**

    - Big O
    - Whiteboard Solutions
    - Sample Test Questions
    - github and open source

Disclaimer: Syllabus is subject to change due to current events, schedule changes and in particular level and interests of students.