New York University
School of Professional Studies
Management and Information Technology

Introduction to Python

INFO1-CE9990, Spring 2017

Schedule:

Section 1: Tuesdays from 6:30pm - 9:30pm from 1/24 – 3/28 - 10 sessions
Section 2: Thursdays from 6:30pm - 9:30pm from 3/16 – 5/18 - 10 sessions

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

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

Description: Master the foundations of software development with one of the fastest growing and most in-demand programming languages in the world -- Python. Write powerful applications to solve the most common programming tasks that are encountered by engineers in the field. Learn fundamentals that can be applied to further study in any language. This course covers objects and object types, functions and methods, looping and conditionals, text processing, sorting, and multidimensional structures, as well as how to set up a development environment and debug and troubleshoot your programs. No prior knowledge is assumed -- this course is designed specifically for beginners aiming to get started in the world of software development.

Rationale:

- master core language concepts, including simple algorithms (data types, statements, method and function calls, looping), data parsing and structuring, code analysis and debugging
- learn the basics of a language fast growing in popularity around the world and serving IT needs across numerous industries
- become familiar with a useful, easy-to-use scripting language that can accomplish diverse tasks with minimum effort

Objectives: upon completion of this course, the student will:

- have a firm grounding in core programming concepts and be able to "think like a computer scientist"
- be able to use Python to accomplish myriad programming tasks in diverse application domains
- be familiar with common practices and techniques used to get things done in Python
- be ready to expand knowledge of Python to an intermediate/advanced level

Structure:

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

Course Requirements: The student is required to demonstrate:

1. familiarity with and ability to identify code parts of speech, features and object types and values in any sample of code
2. compose logic in Python to process data
3. 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. take an in-class quiz,
3. participate in class with questions and by responding to instructor queries, and
4. complete weekly programming assignments.
5. Students may also be asked to review their colleagues' work as part of a group process.

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/academics/noncredit-offerings/academic-noncredit-policies-and-procedures.html

- weekly programming assignments: 45% of grade
- weekly quizzes: 45% of grade
- attendance and "continuity" (see below): 10% of grade

Grading Criteria: in addition to demonstrating the proper use of the code features highlighted in the assignment, you are also required to adhere to basic quality standards as outlined in the "Code Quality Criteria" document (on the handouts page). Solutions cannot be reviewed until they adhere to this criteria. "Continuity" means keeping in touch with the instructor regarding continuing progress, especially when student's personal or professional conflicts cause an absence or delay.

Deadlines: Homework solutions are due by Wednesday morning (Section 1) or Monday morning (Section 2) of the week of class so the instructor / other students will have time to review and comment. 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

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 by Wednesday morning.

- 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. Use of online resources is discouraged (principally because they may offer a shortcut to completing assignments, often without complete understanding, also because they introduce new topics that may be confusing). Under no circumstances can code found on the web be used in assignments.

Course Syllabus:

1. **Introduction to Python and Setting Up**

   - objects and core data structures
     - variables
     - objects
     - objects with math operators
     - **print** statement
   - functions: arguments and return values
     - **raw_input()**
     - **int()**, **float()**, **str()**
     - **round(), len()**
   - debugging: problem solving and attention to detail and error messages
   - getting started
     - working from the command line, command line tools
     - getting Python on your platform
     - writing and running Python programs
   - course requirements and course resources
     - your accounts on **home.nyu.edu**

2. **Built-in Functions and Object Methods; Conditionals and Blocks**

   - objects and object behavior
   - methods
     - **str.upper()**
     - **str.format()**
   - the difference between functions and methods
   - arguments and return values with methods
   - combining expressions
   - **if**
   - blocks and block structure
   - **elif** and **else**
   - **if not**
   - **and** and **or**
   - **while**, **break** and **continue**
   - method tests: **str.isdigit()**, **str.isalpha()**
   - string formatting with **str.format()**
   - **exit()**
   - debugging loops: the "fog of code"

3. **Looping and Selecting Tabular Data; file**, **list** and **str** conversions

    - **file** object
    - the **for** statement
    - the newline character and **str.rstrip()**
    - **str** slicing
    - **str.split()**
    - **list** object: subscripting and slicing
    - arguments to the program with **sys.argv**
    - using a "summary" variable to count or sum inside a loop
    - **file.readlines()**
    - **str.splitlines()**
    - **len()**
    - **print** revisited
    - identifying type
    - debugging: **pdb**

4. **Summarizing Data with Containers**

    - 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 functions
    - introduction to sorting
    - debugging: striving not to guess

5. **Dictionaries and Sorting**

    - summarizing data with containers: **dict**
    - sorting containers

6. **Structuring and Sorting Complex Data**

    - sort criteria function
    - looping through and reading a complex structure

7. **Complex Data, Part 2; Power Tools for Filtering, Transforming and Sorting**

    - looping through and building a complex structure

- sorting multidimensional structures
- lambdas for sorting
- list comprehensions
- **enumerate()**
- **range()**
- Exceptions
- converting **dict.items()** back into a **dict**

8. File and Directory Input / Output

- writing and appending to files
- opening a file in **with** context
- **sys.stdin** and **sys.stdout**
- **sys.stderr**
- **sys.argv**
- **os.listdr()**
- **os.path.isfile(), os.path.isdir()**, **os.path.getsize()**
- trapping Exceptions

9. Functions and Modules

- functions
  - function basics
  - argument matching modes: positional, keyword
  - argument matching modes: variable positional, variable keyword
  - function scoping and variable scopes
- modules
  - creating reusable modules
  - using modules
  - modules as programs and modules as libraries
  - installing modules

10. Classes

- the case for Object Oriented Programming
- class as an instance factory
- instance methods
- instance attributes
- setter and getter methods
- encapsulation
- instance vs. class namespace
- object constructor: **__init__**
- static methods and class methods
- inheritance and polymorphism

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