Course 1: Introduction to Programming

General Course Information

Instructor: Ian McLaughlin
Email: im60@nyu.com
Semester of the course: Summer 2016
Dates of the course: June 27, 2016 – July 9, 2016

Course Description

An introduction into the world of coding from the ground up. Begin by learning to think in the way that computers operate and programs think and to work your way up to writing actual functioning Python code. Learn how to write, read, and debug code. Work with variables, discover how functions, methods and operators work with different data types, and learn how to work with loops. Learn procedural Python. Apply debugging techniques including understanding error message types, statement insertion and the python debugger. Mastery of the concepts in this course provides a firm foundation for further study in any language.

Course Prerequisites

This course assumes no prior knowledge of Linux, Python, or programming in general. Comfort with algebraic and geometric problem solving techniques will be helpful when solving exercises that may be mathematical in nature.

Course Structure/Method

Lecture / Lab. This class meets in person from 6:00 pm to 9:00 pm on 6/27, 6/29, 6/30, 7/5, 7/6, and 7/7, and from 10 am to 5 pm on 7/9.

Course Learning Outcomes

Upon successful completion of this course, students will be able to:

1. Perform essential operating system tasks within the Unix command line interface

2. Describe a problem solution in programming terms
3. Write and run Python modules using a text editor and a command prompt
4. Create basic programs in Python using core features such as variables, operators, functions, methods and program flow

Communication Policy

The best way to contact me is via email. I will do my best to respond within 24 hours. Communication and participation in class is not only encouraged, but required.

Course Expectations

Class Preparation

Each session will consist of two components: discussion and lab. Discussion consists of a mix of lectures, programming examples, and question-driven group analysis of one or more large programming problems. Lab will consist of either group or individual work on exercises or projects. Questions arising during lab may be used to fuel additional discussion as time permits.

Attendance

Success as a student begins with attendance. Class time is not only for learning new skills, but also for practicing what you have learned. Most assignments and demos are completed in class. Students are expected to attend classes regularly, arrive on time, and participate. The Introduction to Programming with Python Diploma does NOT have excused absences.

Instructors take attendance during every session. Students are responsible for remembering or keeping track of when they miss class. The final grade of any students who miss 5 or more hours will suffer and may result in dismissal from the course.

- Absence: not attending class
  - Hour-cost: 2 hours
- Tardy: arriving to class 15 minutes to 1 hour late OR leaving class 15 minutes to 1 hour early
  - Hour-cost: 1 hour

Students are encouraged to e-mail their instructors when they are absent. It is polite to inform instructors so that they can let students know what they missed. Students are responsible for all academic work missed as a result of absences. It is at the instructor's discretion to work with students outside of class time in order to make-up any missed work.

Required and Recommended Material

Computer Hardware Requirements

- 8GB of RAM
- 100 GB of free disk space

Software

All software will be provided in class.

Assessment Strategy and Grading Policy

Given the introductory nature of Course 1, assignments are completed individually and are centered around problem solving techniques. All assignments must be completed by the end of Course 1 in order to advance to Course 2. Note: all exercises should be committed to version control and backed up to GitHub.

Assignment	**Title**	**Points**	**Due Date**
1	Complete vimtutor	10	06/27/2016
2	Shell challenges	10	06/29/2016
3	Python interpreter exercises	10	06/29/2016
4	File system scripting challenges	10	06/30/2016
5	Procedural logic with Python, Bash, and C	10	07/05/2016
6	Debugging challenges	10	07/09/2016
7	Numerical problem solving challenges	40	07/09/2016

Course Outline

Session 1: 06/27/2016, 6pm-9:30pm
Description: Basic environment setup. File systems and their navigation with the command line. Editing text with Vim. Source control with Git and GitHub.
Assignments: Complete Vimtutor. Shell challenges.

Session 2: 06/29/2016, 6pm-9:30pm
Description: Python environment management with virtualenv. Installing packages with pip. Using the Python interpreter, and improving with bpython. Where to find help.
Assignments: Vimtutor revisited. Shell challenges revisited. Python interpreter exercises.

Session 3: 06/30/2016, 6pm-9:30pm
Description: What is a script? What is a scripting language? Reading, writing, and editing Bash scripts. Python as a Bash alternative. File system manipulation with Python.
Assignments: File system scripting challenges. Debugging challenges.

Session 4: 07/05/2016, 6pm-9:30pm
Description: What is a program? Hello World and numerical computation in C. Python as a C alternative. Integers, floats, strings, and the meaning of type.
Assignments: Complete same procedural logic in Python, C, and Bash, printing the same message to the terminal for all three programs.

Session 5: 07/06/2016, 6pm-9:30pm
Description: Style guide: the basics. Writing comments. Introduction to numerical and string methods. Using Python as a calculator and string parser.
Assignments: Critiquing program style. Numerical problem solving challenges.

Session 6: 07/07/2016, 6pm-9:30pm
Description: Data organization with Lists, Tuples, and Dictionaries. Program flow with looping constructs.
Assignments: Continue numerical problem solving challenges.

Session 7: 07/09/2016, 10am-5pm
Description: Reading error messages and debugging. Handling errors. Introduction to algorithmic thinking.
Assignments: Complete numerical problem solving challenges. Complete debugging challenges.

At the discretion of the instructor, the syllabus may be modified to better meet the needs of the students and to achieve the learning outcomes established in the syllabus.