Course Philosophy

This course is a natural progression of 6.00.1x and if you took that course, you will remember that it was an intense but rewarding experience. We are glad you are back with us for 6.00.2x! This document is a repeat of the course philosophy of 6.00.1x and we hope you will take the time to refresh yourself on the following points. If you are just joining us for 6.00.2x, we hope that this document will give you a general feel for what you should expect in terms of workload, the course structure, and most importantly, what we hope you will get out of it.

This is a hard course. It is easy to get discouraged, but don't! As you will noticed from the first assignment, there are many components and in order to successfully complete all parts, you have to:

- understand the relevant subtleties of the Python language
- critically think about approaches to solve the problem
- have good time management skills
- pay close attention to deadlines

You have to put in many hours, *especially if you have never coded before*. The course description does say that no coding experience is necessary, but as with anything else, prior experience will make it easier for you. We am sure that students with no coding experience have completed the first assignment perfectly, but they probably put in a lot of work, asked questions, and after some frustrating but rewarding hours, managed to get the correct code. It is our intent that over the entire course, students should be able to get the work done in about 10 hours a week. *If, on average, you are spending lots more than that, let us know.*

We encourage research and independent learning. You may have noticed this now that we are two weeks into the course. This goes back to keeping this course on par with the MIT campus one. Independent research comes in many forms. *But ultimately, it comes down to paying attention, repetition of the concepts, and practice practice practice!* Your research can be done in many forms:

- try out code in your Python development environment (Canopy, IDLE, or whatever)
- read the Python documentation or other Python tutorials
- use a paper and pencil to sketch out solutions
- if you have the textbook, read the relevant sections
- post to the forums

This course does not aim to just teach programming. *We also teach computational methods.* As such, we may not cover every detail of the Python language. Otherwise, this would become a course on "Introduction to Python", which it is not. If you are unfamiliar with Python, we have posted many Python resources available online. The forum participants, staff, and community TAs are also willing to help you if you have a question about certain Python constructs.

The staff and community TAs are here to help you. *We cannot stress this enough!*

- We are here to offer suggestions to put you on the right path.
- We are here to clarify Python language subtleties.
- We are here to fix bugs in the course content.
- We are here to have interesting discussions.
- We are NOT here to give away answers.

If it seems that we are curt at times, please try to understand why. While we are happy to post an explanation a couple of times, it becomes tedious and counterproductive to post the same explanation many times. We would rather be helping a student understand logic of their code rather than, for example, telling yet another student to check the Full Output button when their code does not run correctly with the grader. *We expect students to search the forums for similar problems and read the instructions carefully.*

We are all here to learn computer science be it from the videos, the exercises, or each other – we distinguish this course from other introductory computer science courses because of the *accelerated pace and focus on students critically thinking about problems*. We will stress practice as a very important part of learning. Start early and do little bits of exercises every day (during lunch breaks, 10 mins of goofing around on the Python shell every now and then, taking a piece of paper and a pen to your kid's soccer game, whatever), the material starts to seep in. *This type of learning is highly encouraged.*

This course will be offered every term. If you think you can't keep up this time:

- Do your best!
- Keep asking questions!
- Keep trying out code in your Python shell!
- Try to learn as much as you can at your own pace!

Then if you try the next offering again, with the knowledge you gained this time, you will conquer this course with more ease.

We know we can't please everyone but we thank you for your understanding. If you have concerns about the course, please raise them in the forum. We can discuss them but let's not argue. *We will do our best to address them or pass them on to the course admins.* We value all suggestions but please keep in mind that the course cannot be drastically changed this offering. Students and staff alike, we all want to make this an AWESOME computer science course! By the n-th iteration, we will surely have reached a good balance of tough and rewarding!

We hope you will keep at it and we wish you continued success in this course!