Draft Syllabus COMP-10 Introduction to Computer Science

Synopsis

These days computers are indispensable tools for research. This does not only hold for "technical" fields such as physics or chemistry but also for the Humanities and the Social Sciences. While most students are competent users of standard software such as word processing or spreadsheets, the real power of the computer is unleashed when we are able to program it ourselves to perform useful tasks that are tailored to what we want it to do for us.

This course is aimed at people who have no (or very little) previous experience in computer programming, and who not *necessarily* want to major in computer science. It will cover some elementary principles of computer science, and writing basic programs in the computer language Python.

Course objectives

This one credit course is teaching computer programming and computer science at an introductory level, without implicitly or explicitly requiring any previous background in Computer Science or related fields.

The objectives of this course are that after completion, the students are able to:

a) understand basic concepts of Computer Science such as information, representation, and computation.
b) understand what computers can do for us, and what they can't do for us.
c) achieve a basic understanding about how computers work "under the hood"
d) know how to write elementary programs in the computer language Python
e) know how to design and write computer programs in Python to perform computational, text manipulation, and data manipulation tasks.
f) *enjoy* solving problems by programming a computer.

Textbooks

Title:	The cartoon guide to the computer
Author:	Larry Gonick
Publisher:	Harper Paperbacks
ISBN13:	978-0062730978

Title:	Starting out with Python, 3rd edition.
Author:	Tony Gaddis
Publisher:	Pearson, published 2015
ISBN13:	9780133582734

Evaluation (grading)

At about 2/3 of the course there will be a small exam in which you are required to write a few small Python functions that solve elementary problems. This exam **should be passed** in order to get credit for the course. The difficulty level of this exam will be **lower** than that of some of the homework assignments that you have already done before the small exam takes place. The function of this exam is solely to establish whether you can apply basic programming techniques individually, without help from others. If you have made sure you were able to solve and understand the homework problems by yourself, you should be able to pass this exam with ease. You can use books, notes, homework exercises and the python documentation during this exam, but not the internet. An example exam will be provided.

Your final grade will be computed from the number of points out of a total of 100 points. Of these points,

Max. 50 pts will be for the homework assignments (max 10 pts for each)

Max. 50 pts will be for the final project.

Homework grading is roughly based on: 3 points for demonstrating understanding of the problem, 5 points for solving it correctly, 2 points for clarity/readability of program.

The final project is a mre comprehensive computer program in Python, written by yourself, that accomplishes some nontrivial task or solves a nontrivial problem. If at all possible, your project should do something useful and interesting that is relevant to the field that you are majoring in.

The grading criteria for the final project are:

- Max 10 points for clarity, structure, and readability.
- Max 10 points for the program doing what it was supposed to be doing.
- Max 10 points for the *elegance* of the program (this aspect will be demonstrated and explained during the lectures).
- Max 10 points for the documentation of the program (5 pts for the programmer documentation and 5 points for the user documentation)
- A difficulty multiplier that multiplies your points for the project.
  The multiplyer is M = 1 + (DF * 0.05), where DF is the difficulty factor:
  1 = very easy
  2 = easy
  3 = between easy and hard
  4 = hard
  5 = very hard

Final course grading scale:

A+	97-100
A	93-96.9
A-	90-92.9
B+	87-89.9
B	83-86.9
B-	80-82.9
C+	77-79.9
C	70-72.9
D+	67-69.9
D	64-66.9
D-	60-62.9
F	59.9 and below

Homework & collaboration policy

The homework assignments are to guide you through the various aspects of programming, and enable both you and the teacher to assess and monitor your progress during the course.

Collaboration and teamwork are important to advancing in academia and learning new things. In programming courses, you should collaborate but there is a grey area between collaboration and copying.

So please carfully read Tufts' Academic Integrity Policy (available at: http://uss.tufts.edu/studentAffairs/documents/HandbookAcademicIntegrity.pdf). In addition to the text there, the context of a programming course requires more precise language.

1. You may work with others, but make sure all of the code you submit is your own, and that you can explain how it works and why you coded it this way.
2. If someone helps you with a program, cite their help in your program code. It is better to err on the side of caution.

3. In some programming courses, you may be prohibited from sharing or evaluating code. In this course, you can ask others for help and help others in reviewing their code. However, you and only you should add or subtract code from your program. Generally, you can think of this as “Do discuss the program. Do discuss the problem. Don’t talk code.”
    a) When helping others, don’t tell them what code to write, but tell them how to approach the problem.
    b) When getting help, don’t let others tell you what to type or let them type.
4. Do not post homework questions on any online forum. You can ask generic questions that can help with subparts of the problem. Again, all the code must be your own.

Cases of academic dishonesty are required to be reported to the appropriate Dean and will be handled formally.

Late Assignment Policy.

All assignments should be submitted to Trunk on time. Late assignments will result in losing the points for that assignment. In cases of late assignments due to circumstances beyond the students control, accomodations can be discussed, provided the circumstances are formally documented.

Students with disabilities.

Tufts University is committed to full inclusion of all students. Students who require academic accommodations for a disability should take initiative to ensure that we receive the appropriate documentation. Students may speak with the Coordinator of Services for Students with Disabilities to discuss the process for requesting accommodations. See https://students.tufts.edu/student-accessibility-services/how-we-help/academic-accommodations

Tentative Schedule

NOTE: This schedule is *tentative* (hence the section title). This means that depending on how fast or slow we can cover the materials, and on what additional issues come up during the lectures/discussions, the schedule is subject to changes.

Session #	General topic
1	General introduction.
2	What is information?
3	How to represent information?
4	What is computation? What can and can't be computed?
5	The Python environment
6	Numerical and boolean values. Arithmetical and logical expressions.
7	Variables and assignments. Left hand sides and right hand sides.
8	Strings, if/then/else.
9	Lists and tuples, while loops
10	Sets and dictionaries
11	For loops and iteration over data structures.
12	Idem
13	Functions and return values.
14	Functions and recursion
15	Idem
16	File I/O, reading
17	File I/O, writing
18	Error handling
19	Exceptions
20	EXAM
21	Simulations with numbers
22	Simulations with text
23	Discussion of project plans
24	Discussion of project plans
25	Final discussion and feedback session.