INTRODUCTION TO COMPUTER SCIENCE I
SPRING 2019
COURSE INFORMATION
PROF. KAPLAN

Last updated: 2019-Jan-08

Be sure to read **all** of this document!

1 The topics

A computer can carry out operations that you request at staggering speeds. The trick is learning how to instruct the machine to do your bidding. Learning a *programming language* (in this case, Java) that allows you to direct the machine is necessary and interesting, but it is **not** the sole focus of this course. The real focus, and the greater challenge, is determining **which** operations you want the machine to perform, and **in what order**. Learning a language in which to express these operations is merely a side-effect of this course, albeit a useful one.

Given a particular problem that you want to solve, you first need to devise a solution. Then, you need to devise a sequence of operations that will carry out that solution—that is, an *algorithm*. Only then can you express that algorithm in some programming language, thus creating a *program*. In this course, we will tackle increasingly difficult problems for which we can create programmable solutions. Although we will have time only to introduce you to the concepts of algorithmic problem solving and computer programming, you will have seen and used all of the basic building blocks necessary to devise and express the solution to **any** computational problem.

The basic topics are given below, roughly in the order that will we cover them. If you have no idea what some (or many) of them are, don't be alarmed—you will find out soon enough. **This course assumes no previous knowledge of programming, computing, or computers**.

- **Storing, moving, and combining data:** *Variables* and *data types*; arithmetic operators.
- **Choosing which and how many operations to perform:** *Conditional statements* and *loop statements*.
- **Making and using small, self-contained units of code:** *Functions*, *function calls*, and *encapsulation*.
- **Using simple collections of data:** *Lists*, single- and multi-dimensional.
- **Basic algorithmic strategy and analysis:** *Searching*, *sorting*, *shuffling*, and how long they take.
- **Basic file I/O:** *Reading* and *writing* sequences of data.
- **Basic object orientation:** Using *objects* and defining new *classes*.

This course will be project-intensive. Much of the material will seem easy enough to comprehend when presented in class, but the only way to understand this material thoroughly is to use it. That is, to truly understand a problem in depth, you must formulate an algorithm to solve the problem and then write that algorithm in a programming language. In this manner, our projects will require you to address these problems in detail.

In this respect, **computer science is special**: the ultimate arbiter of whether your solutions and algorithms are correct is the wonderful machine that can carry out your operations and show you the result. If your work is at all flawed, you are likely to observe the consequences of that flaw; you will be able to diagnose your errors by repeatedly running your program with small variations in the *input*—the data that you feed into the program; you will be able to verify the correctness of your solutions by testing it many ways. The process of *debugging*—finding and fixing incorrectness in your programs—is likely to change how your brain works, requiring a clear understanding of Python's capabilities and rigorous and organized testing to insure that you find the problem.

2 Lectures, labs, and help

Lectures and labs: The lectures for this class are on **Monday and Wednesday** of each week, from **11:00 am to 11:50 am** in **Chapin 201**. The labs occur on **Fridays** in **Webster 102**, with half of you in the lab section from **11:00 am to 11:50 am**, and the other half from **12:00 pm to 12:50 pm**.

You are expected to be present for **all of the lectures and labs**, and missing either is strongly discouraged. I will not teach material twice, so if you miss a lecture or a lab, then you're on your own. If you must miss lecture or lab due to an illness, a curricular conflict (e.g., a Geology field trip), or an emergency situation, contact me and I will arrange to handle the problem. **If you have a extra-curricular conflict** with a lecture or lab—for an athletic event, for a (non-curricular) musical or theatrical performance, to depart early for or arrive late from a vacation, or for any other non-emergency—then **the choice is yours to miss or to attend**. If you choose to miss the class meeting, I do **not** need to know **why** nor even **that** you will be absent. You have elected, voluntarily, not to attend, and you must be prepared to obtain and learn on your own the material that you missed. I recommend that you choose to attend the class meeting when these conflicts arise. Do not underestimate the willingness of those who run extra-curricular programs to make accommodations for your academic priorities.

I expect you not only to attend lectures and labs, but also to be attentive for them. The time will be best spent if it is interactive, and that requires that you be up-to-date on the class material, and that you be alert and prepared to participate.

Office hours and meetings: If you seek assistance, reinforcement, review, or other opportunities to discuss the course material or assignments, you should see me. There is a link on the course web page for scheduling times for meetings. Please use them; chatting with me outside of class is one of the reasons you came to a small college.

Email: Many questions simply do not need an in-person meeting, at least not initially. You should certainly feel free to send email me with your questions or concerns.

Tutors and TAs: If you are struggling with some aspect of the class, your first line of defense should be to **see me**. However, we may find it best to spend additional time with a peer tutor, going over the material from lectures and working on the projects.[1] If you find yourself in that situation, go to the college's page on how to obtain a peer tutor.

Additionally, each lab section will have two or three *student teaching assistants (TAs)* to help answer questions and provide guidance in the labs. I will also schedule *TA help sessions* during two evenings of the week. I will announce and post when these help sessions will be held.

3 Texts and materials

The text for this course is online, and you may download it as a PDF. It will serve as a reference and reinforcement of the material covered in class. However, your primary source of material for this course is our lectures. You may also obtain this book, as well as another text that we've selected as an optional reference, from the Reserve Desk at the Keefe Science Library in the Science Center.

All other tools for this course—all of the software and documentation—will be provided. We will see, during our first lab, how to access the computer system on which you will do your programming. If you wish to use these tools (or other tools) on your own computer, you are welcome to do so.

4 Assignments, deadlines, and extensions

There will be a number of programming projects. The deadline for each will be stated clearly on the assignment, as will the manner of submission. **Late submissions may receive failing grades**. Turn in what you have, and do so on time.

An extension for any assignment **must be requested, in writing** (email counts as *writing*), **at least 48 hours prior to the deadline**. The determination as to whether or not a particular situation merits an extension will be made on a case-by-case basis. Scheduled events are **not** sufficient reason to warrant an extension. Rather, extensions are intended for unusual circumstances that prevent you from planning your time well in order to meet the deadline. Note that a sudden onset of illness or other emergency situation that occurs less than 48 hours before a deadline will be treated as a special case.

[1] Be cautious about project work with tutors! It is critical that they provide only conceptual feedback or hints about structuring or debugging your code. To see actual code written by the peer tutor to solve a part (or all) of a project is *plagiarism*, and thus to be stringently avoided.

5 Exams

There will be **one mid-term exam** given during a regular class lecture hour; there will also be a **comprehensive final exam** given during the final exam period at the semester's end. The mid-term exam will be given during week 7 of the semester (with the specific day of that week to be announced); the final exam will be a 3-hour, scheduled exam. Its time and location will be announced when the Registrar's office posts the final exam schedule.

6 Grading

For most students, this course contains a great deal of mystery. While computers are likely familiar to you, their inner workings are not. Worse, it's not clear what *computer science* **is**: is it applied mathematics? theoretical mathematics? engineering? empirical science? The answer is that it is all of these things. Unfortunately, it is unlikely that knowing how to categorize computer science is of much help to you as you consider and (later) reflect upon this course.

Because grades matter to you for good reasons—they may affect your search for a job, or your applications to professional or graduate programs—you may therefore treat this course as a risk. So much is unknown about the course material, the projects, the exams, and our expectations for your work. However, I believe that the type of thinking on which this course focuses is a major intellectual asset, and I do not want you to avoid it for fear that your GPA may suffer.

A special grading policy: Therefore, this course employs a **special grading policy** that is intended to minimize that risk. It is a policy that requires only your effort and organization; in exchange, the risk is largely removed. Specifically, this special policy is:

> If you complete all of the course work, submitting all assignments on time, and demonstrating a sincere effort in all submitted work (including exams), then your recorded grade for this course will be no lower than a `B`.[2]

The projects and exams for this course will demand some effort from you. However, if you struggle to master the material, submitting work that is complete but flawed, then you do not risk a damagingly low grade. I require only that the work be submitted by its deadline, and that all work demonstrate real effort to produce correct solutions. Clearly, *a sincere effort* is a subjective standard. However, if you put forth the effort that I expect, that effort should be unambiguously apparent to me when I grade your work. If you are uncertain whether your work will meet this standard, then simply **ask me**.

In short, if you try, and if you are organized, then you will receive at least a middling grade for this course. Our hope is that this rule will leave you free to struggle with the material for its own sake, enjoying the challenges and puzzles with less distraction.

[2]It is important to recognize that this special rule is intended to remove the risk of a poor grade **for those who put reasonable effort into the coursework**. If you fail to put in that reasonable effort, then any final grade may be possible.

Calculation of your final grade: There is no formula to determine your final grade. I will assign a final grade based on the evidence you have presented of your mastery of the material at the course's end. While that is (intentionally) vague, I will provide the following guidance:

- **Major programming projects** are modestly important. They show the result of your effort on more complex problems over a loosely bounded amount of time and with broad resources at your disposal.
- **Minor programming projects (labs)** are not very important. They are practice. You should do them, but mostly to be sure that you are on top of the course's progression.
- **Exams** are very important. The final exam is most critical.

7 Academic dishonesty

You will be expected to do your own work on all assignments and exams in this course. While I encourage you to interact with your classmates and discuss the material and assignments, there is a limit to the specificity of such discussions. I seek to make that limit clear here.

It is acceptable to discuss any assignment for the class with a classmate. You may even discuss your approach to a particular problem, or review relevant material for a problem with another person. However, you **may not show another student your work, nor see another student's work. If in doubt, *ask me***. If you are unusure whether or not a particular kind of communication would rise to the level of academic dishonesty, then you should contact me immediately and find out.

8 The big picture

It will be easy to get lost in your code while you write and debug each of the assignments. From time to time, remember the deeper goal of what you're trying to do: take a problem, break down its solution into small, simple steps, and try to express those steps in a rigorous language that the computer can check and then use. Like any kind of writing, learning to devise or understand an algorithm, and then to write, test, and debug it all take practice. Put in the time, and your brain will change how it sees and evaluates all kinds of problems.