

CS8 – Introduction to Computer Science (Summer 2011)

Brief course description:

Introduction to computer program development for students with little to no programming experience. Basic programming concepts, variables and expressions, data and control structures, algorithms, debugging, program design, and documentation.

Requirements:

There are NO REQUIREMENTS, however, the course is not open for credit to students who have completed Computer Science 10, Computer Science 16, or Engineering 3. Legal repeat for CMPSC 5AA-ZZ.

Instructor: Veljko Pejovic

email: veljko@cs.ucsb.edu (put CS8 in the subject line)

TA1: Qian Liu

email: qian@mat.ucsb.edu

TA2: Daniel Havey

email: dhavey@cs.ucsb.edu

Course meetings:

	Lectures	Lab sessions	Office hours		
			Veljko	Qian	Daniel
Time	Monday, Tuesday and Wednesday 11:00am-12:25pm	Thursdays 11:00am-12:25pm Thursdays 12:30pm-1:55pm	Wednesday 12:30-3:30	Friday 11:00-1:00	Thursday 2:00 - 4:00
Location	Phelps 2524	Miramar lab, Phelps	Harold Frank Hall 2158	Harold Frank Hall 2158	Harold Frank Hall 2158

Resources:

Textbook	<i>Python- Programming in Context</i> , Bradley N. Miller, David L. Ranum, Jones and Bartlett Publishers, 2009. Note: two copies of the book have been placed on reserve at the Davidson library
Course website (lecture notes, homework, lab sessions)	<i>Gauchospace</i> (https://gauchospace.ucsb.edu/courses/course/view.php?id=168) Gauchospace forum is the preferred way of interacting with the instructors, too. Please post all your course related questions on the forum.
Calendar	http://bit.ly/INsi5A

Full course description:

This course introduces fundamentals of computer science to students with little previous programming experience. These fundamentals include algorithms, control structures, data structures, data types, expressions and variables, testing and debugging, and documenting programs.

The programming language taught in this course is **Python 3**. Programming in Python is easier than in many other programming languages such as C or Java, as the syntax is much easier to grasp. Python allows us to concentrate on the algorithmic part of programming and do not get distracted by peculiarities of the language. The resources (especially the textbook and the python doc website) provide everything you need to know about Python in order to write your programs. What they do not provide is thinking about your programs, which gets us to the main goal of this class:

**Learn how to think in terms of computer science:
logical reasoning & algorithmic thinking**

Thanks to Prof. Phill Conrad for the following description of the above:

"Computer Science is the study of algorithms. An algorithm is a well-defined, step-by-step sequence of instructions that can be used to mechanically determine the solution to some well-defined problem. You probably use algorithms every day, for example:

- *If you are looking in the index of a U.S. history textbook for "Gettysburg" you'll probably use an algorithm to find the entry quickly, assuming the index is alphabetized. Here the input to the algorithm is some topic, and the output is a list of pages on which that topic appears.*
- *If you are searching through a parking lot to either (a) find a parking space, or (b) determine that there are no spaces left, you probably use an algorithm to do that - again, without even thinking about what you are doing.*

In the case of using an index, this is probably an algorithm you may have learned in grade school, and it has been so long since you learned it, that now you don't even think about it - you just do it. Finding a space in a parking lot and knowing when to give up and look elsewhere is "just common sense"; this probably isn't something you were ever "taught", or even have to think very much about. You just do it.

Computers don't currently have this capability, i.e. the capability to "pick up things by common sense", and it seems unlikely that they will within our lifetime unless there are major breakthroughs in the field of Artificial Intelligence. Such breakthroughs have been predicted for a while, but they haven't happened yet. (Maybe you'll be the one to figure out how to achieve this!) So, for the time being at least, it falls to humans to design algorithms that computers can use to solve problems. In many cases, these algorithms are "just common sense" - the computer equivalent of looking for an empty parking space in a parking lot (and knowing when to give up). Algorithms like this are easy to design. Many of the algorithms we'll see in this course are like that. In other cases, the algorithms are very complex, or very subtle, and coming up with them is a deep intellectual challenge. Furthermore, the impact of a better algorithm on society can be very large. For example, new algorithms in the field of computational science, modeling chemical and biological reactions with computer simulations, can lead to breakthroughs such as new drugs to fight disease, or renewable sources of energy."

This course is a good resource for you to get introduced to programming and thinking like a computer scientist. However, it's up to you to fulfill your potentials. Some people find computer science hard as it requires a certain level of devotion. Skills that we learn in this class cannot be memorized. You cannot "cram" for a computer science exam. You can only practice a lot. This is another good analogy by prof. Conrad:

"You cannot learn to swim, play guitar, or paint from a textbook or a lecture. You can only

- *learn swimming by spending many hours in the pool,*
- *guitar by spending many hours playing the instrument*
- *painting, by spending many hours putting brush to canvas.*

The same is true of programming."

Course goals:

Once you successfully complete CS8 you should be able to:

- Explain functionalities of main programming concepts such variables, functions, control and data structures.
- Integrate and combine the above concepts in order to compose algorithms for solving numerical, sorting, text-analysis problems.
- Write programs in Python, verify outcomes of your and others' solutions and test your code efficiently.
- Apply concepts and algorithms learned in this class to solve problems in different areas: arts, cryptography, image and data processing.

Course outline:

- **Week one:** History of computer science. Algorithms; UNIX commands; How python works; Data types; Functions; Control structures. - **Chapter 1**
- **Week two:** Solving math problems using Python; Math and Random libraries; Selection statements. - **Chapter 2**
- **Week three:** Strings and character manipulation; Cryptography; Mutable vs immutable data types.- **Chapter 3**
- **Week four:** Handling data with lists and dictionaries; Sorting data, finding minimum, maximum and mean values; Function types. - **Chapter 4**
- **Week five:** Handling files; Basic image processing techniques. – **Chapter 5** and first half of **Chapter 6**
- **Week six:** Recursion and L-Systems - **Chapter 9**

Course components:

- Lectures** Lectures are essential to get a big picture of what you are learning about in this course and why. In the lectures you will be presented clear explanations of many computer science concepts that can be very hard to understand on your own. Moreover, not everything that is covered in the lectures comes from the textbook. You will be able to ask for clarifications and occasionally express your opinion on how the course is progressing, thus directly influence the amount of learning that happens in the class. The attendance is mandatory (more about that in the course policies).
- Homework** Homework assignments serve to help reinforce what you have learned during the lectures. They are tightly connected to the material covered in the lectures. You will be given an assignment after almost every class and the due date for turning it in is the beginning of the next class. The assignments are strictly paper-based. NO electronic submissions are accepted. You have to PERSONALLY hand in your assignment. You may not turn in homework late. You may collaborate with at most ONE other person on homework assignments.
- Lab sessions** The assignments given at the lab sessions are the core of the class. You will be given practical assignments that rely on material covered in the class but explore it in a greater depth. The assignments are given at the beginning of the lab session each Thursday, you can start working on them in the lab with your TAs serving as mentors. You are expected to complete the assignment by the next Friday. Late

submissions are accepted with 10% penalty, only if TA has not started grading yet.

Exams There will be two midterms and one final exam. The reason for the first midterm is that students usually don't assess their previous knowledge well and do not identify areas that they need to work on more. Your first midterm will serve as a quick feedback so that you can concentrate on your weak spots and fix them before the second midterm and the final.

Reading Before each class you should read the textbook and other materials that may be distributed in class. The reading schedule is posted on GauchoSpace.

Grading:

- 30% - Lab Assignments
- 15% - Homework Assignments (each question will be graded all-or-nothing, i.e. no partial credit).
- 15% - Midterm exam 1 (Tues., Aug 9) - closed book, one hand-written page, two sides of notes allowed. (unless announced otherwise)
- 15% - Midterm exam 2 (Tues., Aug 23) - closed book, one hand-written page, two sides of notes allowed. (unless announced otherwise)
- 25% - Final exam (Wed, Sept 7th) - closed book, one hand-written page, two sides of notes allowed.

To guard against relying too much on your partner or other sources, if your average on the exams is well below the class average, you may receive an F in the class

A conventional 10 point scale will be used to map your numeric average into a letter grade, with the lower three and upper three points of each range representing plus/minus.

grade \geq 93	A	73 \leq grade $<$ 77	C
90 \leq grade $<$ 93	A-	70 \leq grade $<$ 73	C-
87 \leq grade $<$ 90	B+	67 \leq grade $<$ 70	D+
83 \leq grade $<$ 87	B	63 \leq grade $<$ 67	D
80 \leq grade $<$ 83	B-	60 \leq grade $<$ 63	D-
77 \leq grade $<$ 80	C+	grade $<$ 60	F

This 10 point scale represents the minimum letter grade you will be assigned at the instructor's discretion, the letter grade scale may be altered in the students' favor if this will be better reflect the students' mastery of the material. Thus, if there is a "curve", it will be applied at the end, not to individual assignments. As a point of reference, the curve will be set so that the approximate median grade is a 'B'.

Pair Programming

Most of the lab assignments in this class will utilize "pair programming". In "pair programming" two students work at the same computer. One is typing code, while the other one is reviewing it. Pair programming is a common practice in industry, and it results in fewer bugs and better solution designs than when programmers work alone. In this class you may choose your partner or ask your TA to assign you one. You will be switching roles frequently, and you are responsible for all the work.

Policies:

Plagiarism and cheating	<p>Cooperative work is an important part of learning; you are encouraged to study together, discuss the lectures, and discuss the software solutions. With the exception of your buddy for pair programming, DO NOT:</p> <ul style="list-style-type: none">• turn in duplicate work (even one line or code or comment)• copy work (even one line) from another student's assignment or file or from a published source other than the textbook or lecture material.• lend another student your assignment or look at someone else's working code to fix your problem• e-mail or transfer any of your files to another student or store your program on a computer to which another student in the class has access <p>In addition, anyone caught cheating on an exam will receive a grade of F for the course. The UCSB policy on academic honesty deals with (Section A.2 from: http://www.sa.ucsb.edu/regulations, Student Conduct, General Standards of Conduct)</p> <p>Finally, if you are struggling with the course and need help, contact the instructor or one of the TAs and we will do all that we can to help, including meeting outside of regular office hours if need be, just DO NOT CHEAT.</p>
Pair programming:	<p>In pair programming you are encouraged to collaborate, because this is what happens in the real world. However, we still need to evaluate your performance. It is your responsibility to make sure that each of the team members puts equal amount of effort into assignments. You will be given a single grade for your programming assignment, and if your partner is slacking off – you will bear consequences, just like in a real company. Note, however, that I mentioned effort – we know that not all of you come with the same previous knowledge of computer science and that will be taken into account. Yet, everyone is expected to try hard. Through pair programming you will also develop skills that are necessary for team-work: managing and motivating your team mates, dividing tasks, etc. Please talk to your instructor or TAs as soon as you spot problems in your team and we will help you sort them out.</p>
Attendance	<p>Attendance is required at all lectures and labs (discussion sections) and is checked via homework submission. Therefore:</p> <ul style="list-style-type: none">• you must hand in homework in class• you may not turn in homework on behalf of another student, or ask someone else to turn in yours <p>In addition, you must take exams. There are no make-up exams.</p>

Students with Disabilities

If you are a student with a disability and would like to discuss special academic accommodations, please contact the instructor. In addition, students with temporary or permanent disabilities are referred to the Disabled Students Program (DSP) at UCSB. DSP will arrange for special services when appropriate (e.g., facilitation of access, note takers, readers, sign language interpreters). Please note that it is the student's responsibility to communicate his or her special needs to the instructor, along with a letter of verification from DSP.