CS 8 – Introduction to Computer Science

Syllabus – Winter 2019

Class Time:	**M & W 9:30 AM – 10:45 AM**	**Location:**	**CHEM 1171**
Instructor:	**Ziad Matni**	**Email:**	zmatni@ucsb.edu
Office Hours:	**Wednesdays** 1:00 PM – 2:30 PM in **SMSS 4409**		
T.A. Info:	*See page 8.*		
Lab Times:	**Tuesdays** 1 PM, 2 PM, 3 PM, 4 PM in **PHELP 3525** (50 minutes long)		
Class Main Website:	**bit.ly/cs8Matni** (Note, it's NOT GauchoSpace!)		
Class Piazza Site:	**https://piazza.com/ucsb/winter2019/cs8**		
Catalog Description:	https://www.cs.ucsb.edu/education/courses/cmpsc-8		

Class Overview

Students will learn fundamental principles and concepts of computers and programming, including structured programming techniques. Students learn to use the Python interpreter and write Python programs. We will be using **Python 3** in this class (don't use anything older than version 3.4.3). Computer Science is the study of **abstractions** and **algorithms**. In Computer Science, an abstraction is *a useful representation of something from the real world that allows us to work with it more easily or efficiently*. 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*. Computers are machines that can "process" algorithms with the help of *code* or *programs*.

Learning how to program requires time, perseverance, and consistent practice: exactly like practicing a musical instrument, a field sport, or cooking a gourmet meal. There's a *science* behind programming, but it is also about *technique* – and that requires you to "*get your hands dirty*" and practice, practice, and then practice some more! You are bound to make mistakes – and that's ok because you will learn from them. Making mistakes means **you are learning!** Do not be afraid of trying something that you initially have *no clue* about! Remember that practicing early and often will make you a better programmer in the end. This means that you should not procrastinate and wait until the last moment to do your assignments and homework… but you knew that already, didn't you!? ;)

What you need to know BEFORE you take this course

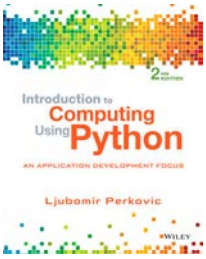
Little or no prior exposure to computer science is assumed. This is a **beginner** class in CS.

What you will learn by the end of this course

You'll have the opportunity to learn all of these things (though not necessarily in this order):

- The "Why do we care?" question regarding computer science and its role in society.
- Problem solving and abstraction
- Memory concepts: variables, names, types, values, and assignment statements
- Strings of characters
- Control structures (for loops, if/else, while loops)
- Functions
- Processing text files
- Lists and dictionaries
- Program style: How to write code that other people can read and understand

Required Textbook



Introduction to Computing Using Python: An Application Development Focus, 2nd Edition
By Ljubomir Perkovic
ISBN: 978-1-118-89094-3

Where Do I Find Stuff?

- All information you need about what today's lecture is, to what homework or lab is due today, is found in **THIS SYLLABUS!**
- All material (class slides, example codes, assignments, etc…) will be on our **main website** at: **bit.ly/cs8Matni**.
- All questions you have should be posted on our class' **Piazza** site. I encourage EVERYONE to participate on Piazza and answer each other's questions to the best of your ability. The TAs and I will observe Piazza postings and chime-in when needed.
- All graded labs and homework can be reviewed on **Gradescope**.
- All grades will be eventually posted on **GauchoSpace**.

Algorithms and Programs

You probably use abstractions and algorithms every day—for example:
If you pick up any textbook, you'll probably find an index in the back of the book. The index is an abstraction—whether the book is about biology, modern art, political science, or computers, the "way the index works" is the always the same. It is composed of the same pieces (topics and page numbers), and organized in the same way (alphabetically by topic, then lists of page numbers in numerical order from smallest to largest.)

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. Here the input to the algorithm is some topic, and the output is a list of pages on which that topic appears.

Algorithms have to be both *designed*, and "*coded*" so the computer can carry them out.
Computers don't currently have the capability to "pick up things by common sense", so we humans have 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 "Gettysburg" in the index (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.

And often, what goes along with finding a good algorithm is finding a good abstraction of the real world concepts we are interested in: cells, molecules, oil fields, words, sentences, students, courses, GPAs, etc. Algorithms and abstractions really go hand-in-hand.

Once an algorithm is designed, we need to *code* it. Coding is expressing algorithms in a programming language. Human languages such as English and Spanish are not very well suited for expressing algorithms. So, special languages are used. In this course, we'll learn the **Python** programming language. We choose Python rather than Java or C++ because:

a) If you are learning your first programming language, Python is easier to learn than the others, and
b) Learning Python provides a good foundation for learning C++, Java, or any other computer language.

If you only learn one programming language, Python is a good choice. It is widely used by scientists and web application developers. For example, many internal systems at Google are based on Python code. This course provides you with the opportunity to become a pretty good beginning programmer, and be well prepared for an intermediate programming course such as CS16 (the first course that counts towards the CS major at UCSB, and which requires at least one quarter of prior programming experience.)

Note that you will ***only*** become a good beginning-level programmer if you put ***a lot of time and effort*** into this course—that is true no matter how much thought and attention I put in my lectures, assignments, and exams.

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

- learn to swim by spending many hours in the pool
- learn to play guitar by spending many hours playing the instrument
- learn to paint by spending many hours putting brush to canvas

The same is true of programming. Programming is not a series of facts to be memorized—you cannot "cram" for a computer science exam. You must practice, practice, practice.

Class Format

This is a large lecture class that meets twice a week and is accompanied by a lab. Attending lectures and your lab is **mandatory**. Attendance will be taken in labs and missing too many will result in reducing your lab grade.

This course has **multiple readings**, **8-9 homework assignments**, **7-8 lab assignments**, **1-2 projects**, **2 midterms** and **one final exam**. You will submit homework as a **hardcopy** in class, submit lab and projects assignments **online**, and do all the exams in the same classroom. It is really important to do the class readings ahead of time. Class participation is vital and highly encouraged (and recognized too!)

Just as in a math classes, everything we do in this class (and almost all classes in CS) builds on all the work that came before. So, ***everything is cumulative***—meaning that you can't afford to miss any classes unless absolutely necessary. Miss two lectures in a 10 week two-lecture per week course, and you've

already skipped 10% of the course—it wouldn't be surprising if your performance (i.e. final grade) in the course dipped by a similar amount!

You may find the workload heavy. It may even feel unreasonable compared to your other courses. However, I assure you that it is not unreasonable, given the goal of making you a skilled beginning programmer. Programming is a skill, and the only way to get good at it is lots and lots of practice, which takes lots and lots of time. The usual "folklore" rule of thumb is 8–12 hours per week for a normal college class. That means you should expect, at a minimum to put in 5–9 hours per week on this course, on top of the 3 hours 20 minutes you spend in lecture and lab each week.

Lectures

The purpose of the lectures in this course is to guide you through the readings, homework, and labs:

- To provide an overview of how everything fits together.
- To provide hands-on demonstrations of Python programming and other things that you'll do on your own later.
- To provide additional information that is not in the textbook (and to sometimes clarify the textbook).
- To provide an opportunity to ask questions, and hear answers to questions asked by others.

This course moves quickly. So attendance is very important.

Homework

In every Monday class, you'll be given a homework assignment that is **due in the following Monday class.** There may be exceptions around or before exams, or to accommodate university holiday schedules.

These are typically pencil/paper type problems, though sometimes you'll need access to a computer to solve them. If you don't have reliable access to a computer at home (or in your dorm), please plan your schedule so that you can spend time in the CSIL computer lab between classes.

Homework assignments are completed on paper—they may **NOT** be submitted electronically—and may **ONLY** be submitted in person, in the class in which they are due. You may type your homework and print it out, or you may print out a blank homework and fill it in with pen/pencil.

You may **NOT** turn in a homework assignment "on behalf of" an absent classmate, or have someone else turn in your homework for you—doing so in this course is a form of academic dishonesty. You can *work* with a “homework-buddy”, but you each have to turn in your own work *and* you have to disclose who you worked with (there’s a place to do that on the homework form).
Again, please do **NOT**:

- Turn in homework on a day other than when it is due. No late submissions accepted (see late policy).
- Have someone else turn in your homework for you (that will be considered academic dishonesty).
- Leave homework in a mailbox or slide it under a door.
- Email your homework or upload it anywhere online.
- Copy answers directly from other students or (heaven forbid!) website. Do your own work!
- Forget to cite (i.e. give credit to) your sources, if you consult your textbook, a website, or person.

Labs

The labs meet on Tuesdays at **PHELP 3525** and are run by the T.As.
Attendance is taken and is mandatory.

Please do not switch your registered lab sections before clearing it with **all** TAs involved (space is tight in these labs). You will likely have to switch sections with someone in order to get this to happen.
You will be given **lab assignments** every week on **Tuesday morning** (lab will be posted the evening before). The lab assignments have to be turned in **by 5:00 PM on Friday**, by uploading them using the **Gradescope** service. You can **ONLY** turn in your lab assignments on **Gradescope**. You will learn how to use the **Gradescope** service in your first lab (Lab 00), which is on Tuesday, Jan. 8th.

In some labs, you will be asked to pair up and work with one other partner in the lab. This "pair programming" concept is explained further in another section in this syllabus.
Again, please do **NOT**:

- Use anything other than **Gradescope** to submit your lab. So don't email them, for example.
- Turn in labs late. They are due by **5:00 PM Friday** (except when indicated otherwise).
- Copy answers directly from other students or websites. **Do your own work**!
- Forget to cite (i.e. give credit to) your sources, if you consult your textbook, a website, or person. This can be done in the program with comments.

Projects

You will be given 1 or 2 programming projects to do. These will be done in groups of 2 (i.e. pair programming) and will involve slightly more complex and inventive programming than your labs. You will be given at least 2 weeks to complete each of these projects. More detailed information will be given in class by the instructor. Projects, like labs, will be submitted using **Gradescope**.

Exams

Both the midterm and final exam are closed book. **The final exam is cumulative**.

The midterm and final exam dates are **set** – see the schedule on **the last page of this syllabus**. I will not accommodate different dates for exams (see my make up policy).

Make Up Policy

If you miss a class, you miss the opportunity for the points on that in-class assignment, or homework that was due. Period. Generally speaking, I do not allow for makeups in this class, with few exceptions.

There is no makeup for homework or lab assignments, except for excused absences arranged and agreed to by the instructor **in advance**. If you don't turn in an assignment by the due date and time, you will get a **zero grade** for that assignment.

There is no makeup for exams. This is a stricter policy than with assignments. The midterm and the final exam dates are announced in this syllabus and are fixed. If you believe you cannot attend any of these dates, especially the final exam, please consider dropping the class.

In rare cases, if there is a documented family emergency, documented extended illness, documented required court appearance, or other situation beyond the students' control (**with documentation**) the instructor may grant additional make up days entirely at the instructor's discretion—but this is **not** a guarantee or a right. Asking for accommodation because "I already bought my plane ticket" or "I have out of town guests that week" is a futile exercise that will get you nowhere…

Late Submission Policy

Late submission means *within* **24 hours after deadline (for homework, labs, and projects)**. *Anything submitted after that is graded with a zero*. Late submissions will result in a **20% penalty**. Recall: homework is due at the **start of class** and you can **only** submit homework in class.

In summary: homework is due every Monday in class; if you turn one in after that before Tuesday at 9:30 AM, you'll get a 20% grade penalty. After that, it's a zero grade.

Labs are due on Friday by 5:00 PM via *Gradescope*; if you turn them in on Saturday at 5:00 PM, you will get a 20% late penalty. After that, you will get *a zero grade*. The same policy applies to projects and their due dates.

The Use of Laptops and Smartphones in Class

I will allow students to use laptops/tablets in class to take notes and try out programming code. However, I am extremely aware of the distractions afforded by laptops/tablets.

If I notice your laptop/tablet activities are completely off-task and distracting to students around you, **you will be asked to leave class and the class will be counted as an unexcused absence**. In general, I expect that you will all behave like the adults that you are, recognize that you are paying for the course you are taking, and treat that time with the respect it deserves.

I do not allow the use of cell phones in class. Please turn them off or put them on vibrate mode before you enter the classroom. If your phone causes a distraction in class, or if I (or one of the TAs) notice you using your phone in class or lab, **you will be asked to leave class**. Additionally, I reserve the right to ban laptops from lecture at any given time if I sense that they detract from the learning outcomes described for the class.

About Pair Programming

Some (but not all) of the programming work in this course will be done using a style of programming known as ***"pair programming"***. This is where two people work together at the same terminal, as "lab partners", to solve a programming problem.

For the assignments where pair programming *is* used, **it is required**, not optional. Here's why:

- Pair programming is a real-world skill that is highly valued by employers.
    - Many companies use pair programming extensively, including several local area employers of UCSB CS graduates.
- Companies that employ UCSB CS and CE grads tell us that our graduates have good technical skills but need better skills and working in pairs and groups to solve problems.
    - Incorporating pair programming into our curriculum is part of our response to this "real-world" feedback.

- Most students find it helpful and enjoyable—UCSB CS students that were surveyed about their pair programming experiences overwhelmingly reported positive results.
- There is also evidence in the scientific literature that it improves student learning, and helps you get better grades.

To learn more about pair programming, watch the following video (it takes less than 10 minutes). http://bit.ly/pair-programming-video.
We also realize that working in groups has another, potentially less positive, side to it: namely the problem of "freeloaders". So, please:

- Do NOT "just copy" homework or code from others and claim it as your own work. That is called plagiarism and is subject to harsh consequences from the instructor, the department, and the university.
- Do NOT work together on assignments **unless you've been specifically told that it is allowed**.

The bottom line:

- The instructor will try to be very specific about what kinds of collaboration are permitted, and what kinds of collaboration are not permitted, and are considered a form of academic dishonesty.
- If you are not sure about whether some kind of collaboration is permitted or not, or if dishonesty is taking place, **it is your responsibility to ask questions**.

Grading and Grade Distributions

Item	Grade %
Homework	10%
Labs and Projects	30%
Midterm Exams (2)	30%
Final Exam (cumulative)	30%
TOTAL	**100 %**

Range	Grade
[93 – 100]	A
[90 – 93)	A-
[87 – 90)	B+
[83 – 87)	B
[80 – 83)	B-

Range	Grade
[77 – 80)	C+
[73 – 77)	C
[70 – 73)	C-
[60 – 70)	D
< 60	F

[X – Y) means ***"X to Y inclusive of X (but not Y)"***

Grades are calculated to 2 decimal places and strictly assigned.

A+ grades: These may be awarded to the *very* best performing students in the class—but the cutoff for A+ grades will be determined at the end of the course at the discretion of the instructor (there is no pre-determined cutoff---an average of 97 or more doesn't guarantee you an A+ grade.)
If I decide to curve the grade (it's not guaranteed that I will), I will do so on the final class scores and not on any individual item.

F grades: ***If you miss your final exam, or your midterm exam, you will receive an F, regardless of your running score in the class***. If you feel that I or the TAs have made a mistake (like adding up a grade incorrectly), then you should certainly bring that to my attention in an expedient fashion (within one week's time), ***but*** engaging in **grade-grubbing** (for example, asking me to round up your final class grade at the end of the quarter) is something you should avoid doing – please know that I will not engage you in these requests (i.e. I typically ignore them). If you have any questions about how grades are computed, please feel free to ask, and I would be happy to explain further.

CS8 (Winter 2019) TEACHING ASSISTANTS

The teaching assistants (TAs) aid the instructor in multiple ways and are responsible to lead the labs, do the grading, proctor classes and exams, and help out students through their office hours. Office hours will be held in "open labs", that is, there might be more than one class' T.As in the lab, which is nice for the students especially when/if it gets crowded. Please bring your laptop computers to office hours, especially if you have specific programming questions. Note that some office hours will be held in Phelps 2150, while others may be held in Phelps 3525.

Name	Email	Office Hours and Location	Lab Time
Alex Ermakov	aermakov@ucsb.edu	Fri. 11 AM – 12 PM in PHELP 2510	Tue. 1 PM
Muqsit Nawaz	mmnawaz@umail.ucsb.edu	Mon. 6 PM – 9 PM in PHELP 2510	Tue. 2 PM
Zexi Huang	zexi_huang@ucsb.edu	Tue. 5 PM – 8 PM in PHELP 3525	Tue. 3 PM
Keqian Li	keqianli@ucsb.edu	Thu. 5 PM – 8 PM in PHELP 2510	Tue. 4 PM

UCSB Policies on Academic Integrity and Honesty

I adhere strictly to the University's academic integrity policy. Please cite other people's work if you are going to refer to it in any of your work.
http://judicialaffairs.sa.ucsb.edu/CMSMedia/Documents/Academic%20Integrity%20at%20UCSB%20edited%20version.pdf
It is expected that students attending the University of California understand and subscribe to the ideal of **academic integrity**, and are willing to bear **individual responsibility for their work**. Any work (written or otherwise) submitted to fulfill an academic requirement must represent a student's original work. **Any act of academic dishonesty, such as cheating or plagiarism, will subject a person to University disciplinary action**. Using or attempting to use materials, information, study aids, or commercial "research" services not authorized by the instructor of the course constitutes cheating. Representing the words, ideas, or concepts of another person without appropriate attribution is plagiarism. Whenever another person's written work is utilized, whether it is a single phrase or longer, quotation marks must be used and sources cited. Paraphrasing another's work, i.e., borrowing the ideas or concepts and putting them into one's "own" words, must also be acknowledged. Although a person's state of mind and intention will be considered in determining the University response to an act of academic dishonesty, this in no way lessens the responsibility of the student.
(Section A.2 from: http://www.sa.ucsb.edu/Regulations/student_conduct.aspx, Student Conduct, General Standards of Conduct)

Disabled Students Program (DSP)

UCSB provides academic accommodations to students with disabilities. **Students with disabilities are responsible for ensuring that the Disabled Students Program (DSP) is aware of their disabilities and for providing DSP with appropriate documentation**. DSP is located at 2120 Student Resource Building and serves as the campus liaison regarding issues and regulations related to students with disabilities. The DSP staff works in an advisory capacity with a variety of campus departments to ensure that equal access is provided to all disabled students.
If you have a disability that requires accommodation in this class, please go see the DSP very early on in the quarter. I will only honor these types of requests for accommodation via the DSP.
More information about the DSP is found here: http://dsp.sa.ucsb.edu

Class Schedule (Mondays and Wednesdays)

The lecture topics are subject to change or re-arrangement. These changes will be announced in lecture.

W #	L #	Date	Topics	Textbook Readings	Hwk # Assigned	Lab # Assigned	Proj #
1	1	1/7	Intro / overview of the class	-	Hw 1	IC0	
	2	1/9	Intro to Python: The basics	Ch. 1, 2		Lab 0	
2	3	1/14	Strings, Lists and Tuples, Intro to Functions	Ch. 2.1 – 2.5	Hw 2	Lab 1	
	4	1/16	Objects and Classes; Turtle Graphics				
3	-	**1/21**	**University holiday – no class**	Ch. 3.1 – 3.5	Hw 3	Lab 2	
	5	1/23	Functions in Python				
4	6	1/28	Conditionals and Loops 1	Ch. 5.1 – 5.6	Hw 4	Lab 3	Proj 1
	7	1/30	Conditionals and Loops 2				
5	8	2/4	Conditionals and Loops 3	Ch. 5.1 – 5.6	Hw 5	Lab 4	
	-	**2/6**	**Midterm Exam #1 (Lectures 1-8)**				
6	9	2/11	String formatting ; File I/O	Ch. 4.1 – 4.4	Hw 6	Lab 5	Proj 2
	10	2/13	File I/O ; Errors and exceptions				
7	-	**2/18**	**University holiday – no class**	Ch. 6.3 – 6.4	-	Lab 6	
	11	2/20	Character encodings				
8	12	2/25	Random and other math functions	Ch. 6.3 – 6.4	Hw 7	Lab 7	
	-	**2/27**	**Midterm Exam #2 (Lectures 7-12)**				
9	13	3/4	Dictionaries and sets	Ch. 6.1 – 6.2 10.1 – 10.2	Hw 8	Lab 8	
	14	3/6	Recursion				
10	15	3/11	*TBD*				
	16	3/13	Review for Final Exam				
Wednesday, March 20th, 2019: 8:00 AM – 11:00 AM FINAL EXAM (cumulative)							

NOTES:

- All assignments will be posted on the class Main website.
- Students must read the textbook chapters before attending class.
- Students must submit all homework in *printed form* **by start of class** on **Mondays**.
- Students must submit all lab and project assignments in *electronic form* on **Gradescope** by that week's **Friday at 5:00 PM**.
- The midterm and final exams will be taken in class. The dates are fixed and will not change.