General Information

Instructor: Ben Schafer
Email: schafer@cs.uni.edu
Office: 316 ITTC, phone 273-2187
Office Hours:
- MWF, 9:00-9:50 AM
- MWF, 11:00-11:50 AM
- MWF, 1:00-1:50 PM
- Anytime my office door is open or by appointment

Time and Place: MWF 12:00-12:50, ITTC 328

Class Website: http://www.cs.uni.edu/~schafer/1140/

This is part of my personal web space and will be the main website for the course. It will contain lecture notes, assignments, announcements, and supplemental class materials.

Course Information:

Course Description
I can hear many of you right now:

“I’m going to be a [high school / middle school] [ math / science / communications / … ] teacher, NOT a computer science teacher. Why should I learn how to program?”

Truthfully, that’s a fair question! But I think I have some good reasons why you might want to take this course and learn how to program.

1) To become more computer literate/savvy. More and more computer programs are being created to allow (or even require) “end user programming.” Your use of these tools will depend on your ability to read, modify, and write fairly basic computer programs. This course is designed to give you the skills to understand the structure and logic of programming and some experience in one or more programming environments.

2) To keep up with your students. Kids are being exposed not only to computers, but to simple programming environments at earlier and earlier ages. As someone studying education you are very likely to end up working with kids who know (or at least think they know) how to program.

3) To learn some of the tools you might get to use on the job. We can’t teach you every tool that you will see on the job. But, we can expose you to one or two that you are likely to see and which also require programming from the user. Even if it isn’t your primary job, your ability to help with an after school program might just be the difference that gets you hired at your first teaching job.

4) To make your life easier. The computer is really good at performing repetitive (and often monotonous) tasks – assuming it has been told what to do. Suppose that you have a bunch of scientific data sitting in a file, or available for you to download off of a website. But, unfortunately, it isn’t in the format that you need. Wouldn’t it be great to write your own script that downloads the data and modifies it for you?
CS 1140 is designed to be an introduction to the basic logic and structure of computer programming in environments that a K-12 teacher might use either with their students or for their own personal benefit.

It assumes no previous experience with computer programming. While access to a computer outside of a university computer lab will be beneficial it will not be required for this course – all course materials are available from any computer with internet access and most programming environments are available in most College of Humanities, Arts, and Sciences computer labs.

**Required Materials**
The material covered in this course is very broad in nature. Because of this, no single textbook fits our needs. Instead, all required readings and other materials will be selected from legally available resources on the internet or from instructor produced materials. Everything you will need to complete this course is either directly contained within the class website (see above) or is available on the internet from other sources. In the case of materials in this latter category, links to these materials will be provided from the class website. Thus, it will be to your benefit to become familiar with the class website and pay close attention for changes and additions.

**Course Learning Objectives**
By the end of this semester students taking this course should be able to meet the following course objectives:

**Programming-oriented Outcomes:**
Students should be able to:
- trace a segment of code to determine the result produced or state achieved by given code
- modify a provided piece of code to accomplish a given task
- choose and sequence action statements to accomplish a given task
- develop and use selection statements (if-then, if-then-else, etc.) to control selection between actions
- develop and use iteration statements (for, while) to control repetition of actions
- explain the concepts of sequence, loops, parallelism, events, conditionals, operators, variables, and lists within the context of computer science.

**Teaching-oriented Outcomes:**
Students should be able to:
- discuss resources for learning about several programming environments
- discuss which of several programming environments would be appropriate in a given classroom
- explain the concepts of sequence, loops, parallelism, events, conditionals, operators, variables, and lists within the context of a K-12 classroom.

**Communication and Support**
My preferred method of communication for one-to-one communication, or when I have a non-discussion based announcement for everyone in the class, is email. Furthermore, my preferred email address is schafer@cs.uni.edu. While I monitor my uni.edu account I spend far more time in my cs.uni.edu account and would suggest that you use this account if you want the most prompt response.

Emails sent to the appropriate address between 8:00 AM and 4:00 PM are normally answered that same day. In most cases you should expect to hear back from me within 24 hours. Emails sent outside of this timeframe may not be returned until the following “school” day. In any situation if you feel that you have not heard back from me in a “timely” manner please feel free to contact me again.

**Course Structure and Policies**
This course consists of the following activities and assessments to assist you in achieving the course and instructional objectives. The first fourteen weeks of the semester will be divided up into week-long modules. Each module will consist of several sections:
Practice Labs – where you begin to apply the skills you learned in class (Approximately 20 in-class and 10 out of class. Worth 10 points each)

Practice is important. You can’t get good at something if you don’t get a chance to actually do it. The Practice Labs are “smaller” and very structured activities that give you a chance to take what you have been doing in class and begin to reflect upon and explore this material. This will normally consist of two separate but complimentary activities.

**In-Class Activities** – The goal of in-class, guided practice activities is to give you an opportunity to practice the material you have been working with on a relatively small scale. Each week you will be presented with one or more smaller programming activities – often started for you – that you must complete following a set of guided instructions. These are designed to focus on one or more building block concepts from the week’s material and are intended to help you gain practice with the material before you complete a full blown programming assignment. Think of guided practice as a lab activity in a traditional science classroom.

In almost all situations you will complete in-class activities with a partner (either of your choice or assigned by me, depending on the situation). As a team, you will complete the activity in class and submit any paperwork or code by the end of class. When partners are part of the practice lab, 20% of your grade for that lab is based on my observation of your ability to cooperate with your partner to complete the assignment. If you sit back and let your partner do most of the work you will not receive full credit for the lab. In the event that you are not in class during a practice lab, it is your responsibility to complete the assignment outside of class. In most cases you will not receive points for the 20% of your grade reserved for partner work since you did not work with a partner.

If you do not complete the assignment in class (either because you are absent or because you and your partner do not finish in time) you may continue to work on the assignment. All work is due no later than the START of the next class session unless alternate arrangements were made with me prior to that deadline. If you are working with a partner it is expected that you and your partner will schedule a time to get together outside of class to complete the work together.

**Out-of-class Activities** – The goal of the out-of-class activities is to get you thinking about one or more concepts from the weekly materials. While the exact format may vary they will typically involve having you exploring some existing code or investigating one or more ideas from the preparation materials and considering their application in your future classroom. These activities will normally be completed as an individual.

**Programming Assignments** – where you take the new skills you are developing and “apply” them to one or more tasks (Approximately 12 assignments worth 25 points each).

By the time you have completed all of the previous materials you should have a good understanding of the main topic(s) from that week’s material. Most weeks will “end” with one or more application assignments where you will write some additional code on your own. While the previously discussed activities can be thought of as “informal” practice, this activity is your main “homework” assignment. Application activities will consist of one or more activities designed to allow you to use the materials you have learned for the week on an activity that you might encounter with students.

Discussion of programming assignments with your classmates should be limited. Discussion should be restricted to understanding the specification for the assignment or asking clarifying questions about the structure/format of a particular command. It should NOT involve looking at other student’s solutions to see how they tackled the problem.

**Final Exam (200 points)** – The university scheduled final exam for this course is scheduled on Monday, December 14th from 1:00-2:50. This exam may consist of both a written and a computer based component. The exact format and nature of the exam will be discussed in the last two weeks of the semester.
Grading

The following approximate points/weighting will be used in assigning final grades. This will likely shift slightly as I add or subtract activities.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Estimated Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>100</td>
</tr>
<tr>
<td>In Class Practice Labs</td>
<td>200</td>
</tr>
<tr>
<td>Programming Assignments</td>
<td>300</td>
</tr>
<tr>
<td>Final exam</td>
<td>200</td>
</tr>
</tbody>
</table>

Grading for this course is on an absolute scale. All points earned will be summed and an overall percentage calculated. Final grades will be assigned based on cut off points no “higher” than:

- 90% or above for some sort of A,
- 80% or above for some sort of B,
- 70% or above for some sort of C,
- 60% or above for some sort of D, and
- below 60% for an F.

However, there are a few exceptions to this:

- I reserve the right to adjust these levels if I deem that assigned activities were more difficult than I had expected. That is, a score of 88 is guaranteed to be at least a B+ but it could become an A- under certain situations.
- I reserve the right to lower your final grade in the course by one grade "level" (for example, from a B+ to a B) for each programming assignment for which you fail to submit at least some evidence of effort towards completion. If an assignment has you completely stumped but you gave it some effort, and then show me that you made that effort. I can tolerate confusion; I can't tolerate people too lazy to even attempt the assignment.

Course and University Policies:

I try to accommodate student needs whenever possible, but I can only do so if I know about them. If you ever have to make alternate arrangements for some activity please contact me in advance. The safest way to make such arrangements is by notifying me via e-mail or phone of your circumstances and of how you can be reached.

All assignments are due at their assigned date and time. In order to receive partial credit, always submit your best effort at that time. I may accept late work on a limited basis, but you should not expect this to be the case.

Incompletes are awarded only in very rare instances when an unforeseeable event causes a student who has completed all the coursework to date to be unable to complete a small portion of the work in the last week or two of the semester (typically the final project or exam). Incompletes will not be awarded for foreseeable events including a heavy course load or a poorer-than-expected performance. Verifiable documentation must be provided for the incomplete to be granted, and arrangements for the incomplete should be made as soon as such an unforeseeable event is apparent.

Scholastic Conduct

You are responsible for being familiar with the University’s Academic Ethics Policies:http://www.uni.edu/pres/policies/301.shtml

Copying from other students is expressly forbidden. Doing so on exams or assignments will be penalized every time it is discovered. The penalty can vary from zero credit for the copied items (first offense) up to a failing grade for the course. If an assignment makes you realize you don't understand the material, ask a fellow student a question designed to improve your understanding, not one designed to get the assignment done. Your final submission for assignments should be individual, original work unless otherwise specified. Any substantive contribution to your solution by another person or taken from a publication should be properly acknowledged in writing. Failure to do so is plagiarism and will necessitate disciplinary action. In addition to the activities we can all agree are cheating (plagiarism, bringing notes to a closed book
exam, etc), assisting or collaborating on cheating is cheating. Cheating can result in failing the course and/or more severe disciplinary actions.

Remember: Discussing assignments is good. Copying code or answers is not.

**Class Distractions**
We live in a technological society, and many of you now carry a variety of electronic distractions with you. These include cell phones, laptops, MP3 players, etc. While you are welcome to own and use these, and other, electronic devices, their use in the classroom is forbidden without my explicit permission (This is a University-wide policy).

A few exceptions do exist, and I reserve the right to approve these situations on a case-by-case basis with prior notification. Unless we have discussed it in advance, all electronic devices should be turned off and left out of sight during class time. The inappropriate use of any of these devices will cause you to receive a participation grade of negative one point regardless of your participation otherwise. Multiple infractions may cause more extreme consequences, including removal from the course.

**Technology Requirements**
Students in this course will rely heavily on the use of the computer. Fortunately all of the preparation materials and most of the programming environments for this course are available from any computer with a web browser and internet access. Furthermore, all of the required assignments can be completed using software available in most CHAS computer labs or available for free download to a personally owned machine.

The following information has been provided to assist you in preparing to use technology successfully in this course.

If you do not own a computer than you can find appropriate machines (with all the correct software) in several different labs on campus. While many will work, I suggest the following:

- Wright 339. This is a public lab and is arguable your best bet.
- Wright 112. This is a teaching lab used for several classes and may not always be available.
- ITTC 335 is a small general purpose lounge. It may be crowded, but it’s advantage is that it is close to my office if you have questions.

I realize that many of you have access to machines at "home." If you would like to work on this course from “home” than you should make sure to have the following:

- Internet access/connection – most of the materials are available to be read online. Therefore, you will need access to a machine with adequate internet access. During weeks where there are video/screen captures used as part of these materials, a high speed internet connection may be required.
- A modern browser – this may seem silly, but several components for this course require that you are accessing the materials from a “modern” (that is, an html5 compliant) web-browser. To know if your browser is sufficiently modern visit [http://detectmybrowser.com/](http://detectmybrowser.com/) Look at the three sections labeled HTML5 and make sure that you have green lights for most of those sections. If not, you should update your webrowser.
- Email – I will send occasional announcements and clarifications via email. By default this will be sent to your UNI provided/assigned email address. You should pay regular attention to this account throughout the semester.
- Word Processor – A typical Reflection Activity will involve downloading a question/response sheet which will be distributed in rtf format. This format is readable from a wide variety of word processors such as Microsoft Word or Apache Open Office. If responses are to be submitted electronically in a word processed format you should use and return this rtf formatted file.
- Silverlight – Many of the lectures prepared by me are delivered using a free Microsoft plugin called Silverlight. In order to play these lecture videos you must install this plugin.
- Scratch – The first unit in this course uses a programming environment called Scratch. This is a web based programming environment so you should have relatively easy access from any web enabled computer. Simply visit http://scratch.mit.edu/. You should have no problem accessing Scratch from any of the CHAS computer labs such as those in ITTC and Wright Hall. If you use your own machine you may program from there as well. However, be aware that Scratch requires a fairly recent version of Adobe Flash. You may need to update the flash plugin to your browser. Do so by visiting http://get.adobe.com/flashplayer/.

- Python – The second and third units in this course will use a programming language called Python. This too should be installed in most of the CHAS computer labs. It too is freely available for download from http://www.python.org/getit/. If you use your own machine you should download version 3.2.x and install this software.

Accessibility
The Americans with Disabilities Act of 1990 (ADA) provides protection from discrimination for qualified individuals with disabilities. Students with a disability, who require assistance, will need to contact the Office of Disability Services (ODS) for coordination of academic accommodations. The ODS is located at 213 Student Services Center. Their phone number is 319/273-2676. Additionally, please contact me immediately if you have a learning or physical disability requiring accommodation

Academic Learning Center Services
You are encouraged you to utilize the Academic Learning Center's free assistance with writing, reading, and learning strategies. UNI’s Academic Learning Center is located in 008 ITTC. Visit the website at http://www.uni.edu/unialc or phone 319-273-2361 for more information.