UNI CS 1510 (Spring 2014)
Introduction to Computing, Section 1

Course Syllabus (Version 2.0)

Lecture:	MWF	11:00am-11:50am	328 ITTC
Lab:	Tu	10:00am-11:50am	112 Wright

Contact Information

Instructor

Sarah Diesburg - diesburg@cs.uni.edu
Office: 311 ITTC Building
Office hours: MWF 10:00-11:50am, 1:00-1:50pm, and by appointments
Class website: http://www.cs.uni.edu/~diesburg/courses/cs1510_sp14/index.htm and UNI eLearning

Teaching Assistants (TAs)

Levi Bostian- bostianl@uni.edu
Grader, General Assistant

Colby Easterday – easterdc@uni.edu
Lab Assistant

Course Description

As the name implies, CS 1510 is the computer science department's introductory course. While it is the first course in the programming sequence for majors it is appropriate for non-majors and it does NOT assume that you have programming experience.

This course has two primary goals:

- First, to introduce the general field of computer science. We hope that you will leave this course with a sense of what computer science is and what computer scientists do. We also hope that you leave with a sense of just how exciting and intellectually powerful the discipline is.
- Second, to introduce the concept of programming. Programming is the way that computer scientists express their ideas and implement solutions to problems. Even if you never "program for a living", you will need to know how to program in order to appreciate the ideas you learn and to work in the industry. We hope that you leave this course with a sense of what programs can do and of how you can use programs to express ideas.

Outcomes

While a major goal of this course is to provide a good start to the development of programming skills, the course is not solely about programming. Upon successful completion of the course students should have gained the following skills and proficiencies:

- general computer & OS usage;
- computer operation;
- a mental model of how programs are executed
- machine capabilities and functions
- a variety of incidental knowledge/understanding.
- general program design;
- standard approaches to common (simple) tasks;
- abstraction (data, procedural, thinking);
- data & problem representation;
- elementary data structures;

Additionally, you should develop skills and understanding that will ultimately allow you to analyze complex problems and apply your knowledge and experience to developing good solutions to them. Programming is a creative process. However, to exercise that creativity, one must learn basic tools and principles. That is the purpose of this course.

Course Material

- Lecture notes (posted on the class Web site)
- Required textbooks: *The Practice of Computing Using Python (2nd edition)*, by William Punch and Richard Enbody

Tentative Schedule (Subject to Change)

Date	Readings and pre-class assignments	Sessions
1/13		Session One - Course Introductions
1/14	Read section 0.8	Lab One - Getting Started in the lab
1/15	Read sections 0.1-0.7	Session Two - Understanding the history and basics of computers
1/17		Session Three - Problem solving and computational thinking
1/20	MLK, Jr. Day - NO CLASSES	
1/21	Read 1.1-1.8	Lab Two – IDLE and Basics of Python Programming
1/22	Read 1.9	Session Four - Lab Debrief and Chapter 1 review Homework 1- First Program
1/24		Session Five - Basic problem solving in python
1/27		Snow Day
1/28	Start section 2.1 (pp 81-90)	Lab Three - Conditionals and Selection Statements
1/29		Session Six - More with data and types Session Seven - Debrief Lab Three Homework 2- Selection Statements- Responding to Customer Requests
1/31	Read section 2.2 (pp 103- top 113, and 114-120)	Session Eight - Multi-way conditionals
2/3	Finish section 2.1 (pp 90-103) Finish section 2.2 (pp113 and 121-137)	Session Nine - Introducing Repetition
2/4		Lab Four - Exploring Repetition
2/5		Session Ten - Debrief and Practice Homework 3- Iteration
2/7		Session Eleven - More loop practice
2/10		Session Twelve - Nested looping
2/11		Lab Five- Stars and Triangles
2/12	Sections 3.1-3.5	Session Thirteen - Lab Debrief Homework 4: Additive and Multiplicative Persistence
2/14		Session Fourteen- Algorithms and Program Development
2/17	Read 4.1 and 4.2	Session Fifteen - Introduction to Strings
2/18	Read 4.3 and 4.4	Lab Six - Strings
2/19	Read 4.5 and 4.6	Session Sixteen - Working with Strings Homework 5- A Game!
2/21	Read 4.7	Session Seventeen - More with Strings - Methods
2/24		Session Eighteen - More with Strings - Penny Math
2/25		Lab Seven - Caesar Cypher- Strings, Conditionals, Loops

2/26		Session Nineteen - Lab Debrief
2/28		Session Twenty- Exam prep, various examples, and wrap up
3/3		In-class Exam #1 (Chapter 1-4)
3/4		In-lab Exam #1 (Chapters 1-4)
3/5	Read 5.1-5.6	Session Twenty One- Introduction to Files
3/7	Read 5.7	Session Twenty Two - More with Files
3/10	Read Chapter 6	Session Twenty Three - Defining your own functions Homework 6- Examining Olympic Medal Counts
3/11		Lab Nine- Functions
3/12		Session Twenty Four - Lab09 Debrief, more on functions
3/14		Session Twenty Five - Function wrap up, new design docs and comments
3/24	Read 7.1-7.5	Session Twenty Six - Introducing Lists Homework 7- Pig Latin
3/25	Read 7.6-7.9	Lab Ten - List Lab
3/26	Finish Chapter 7	Session Twenty Seven - Lists and Tuples
3/28	Chapter 8	Session Twenty Eight - More Lists and Tuples
3/31	See class handouts	Session Twenty Nine – Intro to Functional Decomposition Homework 8- Tracking the Greats of the NBA (Design Version)
4/1		Lab Eleven – Design Lab
4/2		Session Thirty – More Design
4/4		Session Thirty One – Design Wrap-Up
4/7	Read Chapter 9.1-9.3	Session Thirty Two - Introduction to Dictionaries Homework 9- Tracking the Greats of the NBA
4/8		Lab Twelve – "Green Eggs and Ham"- Using Lists for Text Analysis
4/9		Session Thirty Three - Debrief Lab and Memory
4/11	Read 9.4-9.6	Session Thirty Four - Memory
4/14		Session Thirty Five - Sets
4/15		Lab Thirteen – Sets and Dictionaries to Analyze Movies
4/16	Read 11.5-11.9	Review of Lab Thirteen Homework 10: Analyzing Debates- Creating Word Tags/ Clouds of a Speech
4/18		Session Thirty Six - Intro to Searching
4/21		Session Thirty Seven - Intro to Sorting
4/22		Lab Fourteen - Analyze Customer Data
4/23		Session Thirty Eight - More with Sorting
4/25	Read Chapter 16	Session Thirty Nine - Intro to Recursion
4/28		Session Forty - More with Recursion
4/29	Study Guide	In-Lab Exam #2
4/30		Session Forty One - Finishing up Recursion
5/2		Session Forty Two – Final Review
5/6	Final Exam, Tuesday 10:00-11:50am	

Computing Environment

Class Website: Most course materials will be made available on the course web page during the semester. You are responsible for checking this site frequently for reading assignments, prep activities, lecture notes, announcements and supplemental class materials.

Computer Access: We will be using Windows as our primary operating system this semester. You will do your lab exercises and programming assignments using IDLE, the default editor distributed with Python. You have two choices for access to Python and IDLE.

1. By signing up for this class you will be provided Windows access on the CHAS computing network. While there are multiple places where you can log on and use this account, there are three of primary interest.
    a. Wright 112 – This is the main lab used during laboratory sessions. This is a public lab part of the week but it also used by other classes at other times of the day/week and may not always be available.
    b. Wright 339 -- This is mostly a public lab and is the lab in which the TAs will hold assistance hours.
    c. ITTC 335 – This is a small general purpose lounge available to students in the CS department. This is a good place to get a quick printout or check your email between classes.
2. You may work from home and you may use any OS of your choice including Mac or Linux. Python and IDLE are easily downloaded from www.python.org. You should download the latest edition of version 3 (NOT version 2).

Whether you work in the labs or from home, you will need to have Internet access to submit your assignments.

Course Structure and Grading Policies

Grade Determination

The final grade you earn in this course will be based on the points accumulated over five activities as described below.

Activity	Quantity	Points
Class Participation / Attendance	~15 at 2 pts each	30
In-lab work (Tuesday)	13 @ 15 pts each	195
Programming Assignments	11 @ 25 pts each	275
Written Exams	100 and 150 points	250
Programming Exams	2 @ 125 pts each	250
Total		***1000***

To receive a passing grade for the overall course, you must earn a passing grade on the final exam and a passing grade on the projects.

To continue on to the next class in the computer science major, you must earn at least a C.

The grading scale is as follows:

100 – 92	A	69.9 – 68	D+
91.9 – 90	A-	67.9 – 62	D
89.9 – 88	B+	61.9 – 60	D-
87.9 – 82	B	59.9 – 0	F
81.9 – 80	B-		
79.9 – 78	C+		
77.9 – 72	C		
71.9 – 70	C-		

Class Attendance and Participation

Class attendance is required. If you miss a class, it is your responsibility to find out what was covered. I have often found that class attendance is directly correlated to grade. Since each topic in computer science builds on the last, missing classes and material can be severely detrimental to you understanding future material.

In-lab work

Lab is designed to be a time to allow you to learn new skills, apply and practice existing skills, and prepare yourself for the upcoming lectures and programming assignment. Points for these activities will be assigned based on level of difficulty for each activity and will be awarded for successful completion. If you are unable to complete each of the activities in lab, you will be allowed to complete them on your own. However, you must complete and submit the lab sheet by the start of the next class period following each lab unless announced otherwise (normally Wednesday).

Attendance to lab sessions is required - you will receive credit only for labs you attend. The activities for lab are designed to encourage interaction between students and instructors (either me or the TAs) and sometimes between students themselves. Failure to be in lab robs you of some of the learning objectives for that lab. I expect you to be in lab, *on time*, and prepared to work. Students who arrive late will be docked points regardless of the quality of their lab work. In general, students who do not show up at all will not receive credit for making up the lab (although you should still complete the activities so you do not fall behind).

Programming Assignments

Programming assignments are designed to take what you have learned in lab and during lecture, and apply these skills to a program on a scale larger than that explored in-lab. It is expected that you will complete all assignments **as an individual** unless otherwise instructed (see section on scholastic conduct). If you have questions concerning an assignment, feel free to consult your instructor, ask a TA, or post a question on the discussion board.

Programming assignments will typically be assigned over a time period of one week. That is, programs assigned on Wednesday will typically be due the following Wednesday. This is done so that you have plenty of time to think about the assignment and work through any difficulties. You should start assignments as soon as they are made and not wait until the last minute to get started. This is often a recipe for disaster. In order to encourage this, each assignment will consist of three deadlines.

- The "normal" deadline is the due date announced on the assignment (typically the START of class on Wednesday).
    - Electronic submission of your code and submission of a paper printout of the assignment by this due date will earn you up to 100% of the points – normally 25 points.
- The "early" deadline is one class session prior to the "normal" deadline (typically the START of class on Monday).
    - Students who electronically submit their code by the "early deadline" will earn a bonus of 10% of the points they earned on the assignment. This will be automatically added to your grade. Thus, if you earned all 25 points you would receive a bonus of 2.5 points. If you only earned 20 of the 25 points you would earn a bonus of 2 points. Students should still submit the paper printout of their assignment on the normal due date but should include the phrase "bonus eligible" in the header of the assignment.
- The "late" deadline is one class session after the "official" deadline (typically the START of class on Friday).
    - Students who complete and **submit (to eLearning)** the electronic version of their code, and submit a paper printout by this "late deadline," can earn a maximum of 50% of the points possible on the assignment. That is, the most you can earn on an assignment submitted late is 12.5 points.

Exams

There are a total of four exams this semester.

- Two will be "traditional", written, exams offered in the class room – one at approximately midterms and one during final exam week.
- Two will be programming exams offered during the lab portion of the course.

By default these exams are closed-book/closed-notes exams. Under certain circumstances I may allow you bring in certain reference materials. If so, specific instructions regarding what will be allowed during the exam will be published to the course website prior to the exam and will be clearly indicated in the instructions for each exam. It is your responsibility to be aware of what is and is not allowed on a given exam.

The dates of these exams are listed on the class schedule. You are expected to be present for these exams unless you have made prior arrangements. Make-up exams will be offered under very limited circumstances. If you are aware of conflicts prior to the exam, please bring these to my attention as early as possible.

Incompletes

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.

Scholastic Conduct

You are responsible for being familiar with UNI's Academic Ethics Policies (http://www.uni.edu/pres/policies/301.shtml).

It would be easy if I could just state "**copying and collaboration is wrong.**" But, in fact, that statement is partially inaccurate in a programming environment. There will be many times this semester when I will explicitly ask you to copy some starter code, and there are times when you may (and even should) discuss code with other students. In other words, under certain circumstances, copy and collaboration is not only ok, it is expected.

General guidelines regarding "copying"

- If I provide you with starter code, or if your book provides code appropriate to a problem, than "copying" is more than acceptable. In these cases, you should acknowledge the original author of the code in the header block for the code (how to do this will be discussed several times during the first couple weeks of class).
- Copying from other students is **expressly forbidden**. This includes direct copying where one person gives another person their code, but also includes indirect copying – where two students program separate deliverables while sitting right next to each other. Doing so on exams and programming assignments will be penalized every time it is discovered. The penalty can vary from zero credit for the copied items up to a failing grade for the course.
- From time to time you may discover an outside source that provides some information/code which helps you tackle a given problem. 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.

General guidelines regarding collaboration

- First and foremost, your final submission for any assignment (whether it be class prep, in-lab, or a programming assignment) should be your own **individual, original** work unless otherwise specified. To do otherwise is to cheat yourself out of understanding, as well as to be intolerably dishonorable.
- That does not mean, however, that you can't talk about assignments with your classmates or on the class discussion board. 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. **Don't ask to see their answer/code**, but ask them to explain how to approach a problem.

In addition to the activities we can all agree are cheating (plagiarism, bringing unauthorized notes to a closed book exam, etc), assisting or collaborating on cheating is cheating (providing answers to exam questions, discussing test questions with a student who hasn't taken the exam yet). Cheating can result in failing the course and/or more severe disciplinary actions.

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

Accessibility

Please address any special needs or special accommodations with me at the beginning of the semester or as soon as you become aware of your needs. Those seeking accommodations based on disabilities should obtain a Student Academic Accommodation Request (SAAR) form from Student Disability Services (SDS) (phone 319-273-2677, for deaf or hard of hearing, use Relay 711). SDS is located on the top floor of the Student Health Center, Room 103.

Guidelines for Success in this Course

Once you have allocated the necessary time for this course, the following suggestions, compiled from student experiences, should help you plan your time use and prepare for the labs and exams:

- Prepare for lecture! Read any assigned readings *before* the start of lecture.
- Be on time. Class sessions will start promptly.
- Write code on your own! Think of simple problems on your and solve them. If you wonder "what if," TRY IT!
- Start assignments early so you have time to ask questions.
    - Face to face questions are better than email questions.
    - If you email me the night before something is due you shouldn’t get mad if I don’t respond.
    - If you spend more than 15 minutes staring at the computer stuck on something, ask for help!
    - Make use of the office hours early! Don't wait until late in the term to seek help.
- Make sure that your code runs in the lab. This is especially important for those who complete portions of their work at home. Verify that code that runs on another machine actually runs in the lab so there are no surprises at grading time.
- Finally, remember, programming takes practice. You may not get it the first time, but keep trying, asking for help, and caring. Eventually, you might find out that you are pretty good at this whole process!

Finally, I encourage you to utilize the Academic Learning Center's free assistance with writing, math, 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.