UNI CS 1510 (Spring 2017)
Introduction to Computing, Sections 1–2

Course Syllabus (Version 1.1)

Lecture:		MWF 11:00am–11:50am	328 ITTC
Lab (Section 1):	Th	8:00am–9:50am	112 Wright
Lab (Section 2):	Th	10:00am–11:50am	112 Wright

Contact Information

Lead Instructor

Sarah Diesburg - diesburg@cs.uni.edu
Office: 311 ITTC Building
Office hours: M 1:00-3:00pm, WF 9:00-10:50am, and by appointment.
Class website: http://www.cs.uni.edu/~diesburg/courses/cs1510_sp17/index.htm and UNI eLearning

Teaching Assistants: Chanlika Parker, Duc Vu

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.
- 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.

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 (3nd edition)*, by William Punch and Richard Enbody

Tentative Schedule (Subject to Change)

Date	Readings and pre-class assignments	Session links and announcements
1/9		Intro - Course Introductions
1/11	Read sections 0.1-0.7	Session One - Understanding the history and basics of computers
1/12		Lab One - Getting Started in the lab
1/13	Read section 0.8	Session Two - Numerical representation
1/16	No class	
1/18	Read 1.1-1.8	Session Three - Problem Solving and Lab 2 Prep
1/19		Lab Two - Getting Started With Python
1/20		Session Four - Lab Debrief and Chapter 1 review
1/23	Read 1.9	Session Five - More with data and types
1/25	Read Section 1.11	Session Six - Basic Problem Solving in Python
1/26	Read section 2.1	Lab Three - Conditionals and if/else Selection Statements
1/27		Session Seven - Debrief Lab Three
1/30	Read section 2.2.1-2.2.9	Session Eight - Multi-way conditionals
2/1	Read section 2.2.10-2.2.15	Session Nine - Introducing Repetition
2/2		Lab Four - Exploring Repetition
2/3	Read section 2.5	Session Ten- Debrief and Practice
2/6		Session Eleven - More loop practice
2/8		Session Twelve - Nested looping
2/9		Lab Five- Loops
2/10	Read sections 3.1-3.5	Session Thirteen- Lab Debrief
2/13		Session Fourteen- Algorithms and Program Development
2/15	Read sections 4.1 and 4.2	Session Fifteen - Introduction to Strings
2/16	Read sections 4.4-4.6	Lab Six - String Lab
2/17	Read sections 4.7 and 4.8	Session Sixteen - Working with Strings
2/20	Read section 4.3	Session Seventeen - More with Strings - Methods
2/22	Read section 4.10	Session Eighteen - More with Strings - Penny Math
2/23		Lab Seven - Caesar Cypher
2/24		Session Nineteen - Lab Debrief
2/27		Session Twenty- Exam prep, various examples, and wrap up
3/1	**Note: Test in 102 Sabin Hall**	In-class Exam #1 (Chapter 1-4)
3/2	**Note: Test in 112 Wright Hall**	In-lab Exam #1 (Chapters 1-4)
3/3	Read sections 6.1-6.5	Session Twenty One - Files
3/6	Read section 6.10	Session Twenty Two - Files (pt 2)
3/8	Read sections 5.1-5.3	Session Twenty Three - Defining your own functions
3/9	Read sections 5.5	Lab Nine - Working with functions
3/10		Session Twenty Four - Lab 9 Debrief, more on functions
3/20		Session Twenty Five - Function wrapup, code comments
3/22	Read sections 7.1-7.5	Session Twenty Six - Introducing Lists
3/23	Read sections 7.6-7.9	Lab Ten - List Lab
3/24	Read section 7.11	Session Twenty Seven - More Lists
3/27	Read section 7.14	Session Twenty Eight - Lists - Indexing and Sorting
3/29	Read class handouts	Session Twenty Nine- Intro to Functional Decomposition
3/30		Lab Eleven - Design Lab (no link)
3/31		Session Thirty - More Design

4/3		Session Thirty One - Design Wrap-Up, Tuples
4/5	Read sections 9.1-9.3	Session Thirty Two - Introduction to Dictionaries
4/6	Read section 9.11	Lab Twelve - Exploring Dictionaries
4/7		Session Thirty Three - Debrief Lab Twelve
4/10		Finish Dictionaries
4/12	Read Chapter 8	Session Thirty Five - Sets
4/13		Lab Thirteen - Looking for actor relationships in IMDB
4/14		Review of Lab Thirteen
4/17		Session Thirty Six - Intro to Searching
4/19		Session Thirty Seven - Intro to Sorting
4/20		Lab Fourteen - Analyzing Customer Data
4/21		Debrief of Lab 14
4/24		Session Thirty Eight - More with Sorting
4/26		Session Thirty Nine - In-class review session
4/27		In-Lab Exam #2
4/28		Session Forty - Final Review
5/2	Final Exam, 10:00am-11:50am **in 102 Sabin Hall**	

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 Labs: The following labs have pre-configured software for this class:

- **Wright 112** – This is where you will meet for your lab 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. It generally closes at 5pm on weekdays.
- **Wright 339** – This lab is open the latest on weekdays (until 9:00pm or so).
- **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. It generally closes at 5pm on weekdays (or when the last faculty member leaves).

We **highly** suggest using your own desktop or laptop to complete your assignments so you aren't constrained by the open hours of the labs.

Working on your own laptop/computer: You are actually encouraged to work on your own laptop or computer. Having your own computer will greatly aid you in the computer science major, and the computer/laptop itself does not have to be very expensive. The class software is free and will work with Windows, OSX, and 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
Lab Work	13 @ 15 pts each	195
Individual Homework	10 @ 25 pts each 1 @ 30 pts each	280
Concept Exams	125 and 150 points	275
Programming Exams	2 @ 125 pts each	250
Total		***1000***

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

Class Attendance and Participation

Class attendance is required. If you miss a class, it is your responsibility to find out what was covered.

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 and/or effort.

Attendance to lab sessions is required - you will receive credit only for labs you attend. 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 an instructor, come to office hours, or consult a class TA.

All assignments are due at their assigned date and time. Assignments submitted one class period early (in class) may be eligible for a 10% bonus. In order to receive partial credit, always submit your best effort by the assignment due date. **Late work will not be accepted.**

Exams

There are a total of four exams this semester.

- Two will be concept exams offered during the lecture part of the course.
- Two will be programming exams offered during the lab portion of the course.

By default these exams are closed-book/closed-notes exams. 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.

Missing Labs or Exams

If you need to miss a lab or exam, it is YOUR RESPONSIBILITY to let me know BEFORE the time of the lab or exam. Example: Student A is sick and emails me the morning of the lab that he or she cannot make it. Student B just skips lab, and emails me a week later that he or she was sick and needs to make up the lab. Since student B did not follow the rule and show responsibility, student B cannot make up the lab.

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

Since cheating definitions and academic ethics policies are often written for other types of classes, you might tend to wonder how those translate to a computer science course. You may be surprised to hear there are many ways to write a program to solve a specific problem. This is very similar to how there are many different ways to write an essay addressing a particular topic. After a certain point in the course, I will be using plagiarism-detection software to detect similarities that are very unlikely to occur if students were working alone.

Additionally, you need to cite your source if you seek and use help found on the Internet (much like citing a source in an essay course). To do this, you need to put the URL and a brief description of the help you found in a comment directly above the affected block of code. I will show you how this is done further along in the class. However, if you do use code from the Internet, I reserve the right to ask you how it works line-by-line. If you cannot explain it to me, I will not give you credit for that part of the assignment. In other words, *if you use help or code found on the Internet, you must cite it **and** fully understand it.* It is always better to try to figure things out on your own than to use something you don't understand.

In this class, homework assignments must be done on your own as your own individual work. However, this does not mean that you cannot ask for help. Here are some general guidelines for keeping out of trouble.

If you are seeking help from a classmate:

- DO NOT ask to see their code or look at their code.
- DO explain your thought process and where you are stuck in words.
- DO draw diagrams on the board.

If you are helping another classmate:

- DO NOT show them your code.
- DO NOT directly modify their code.
- DO try to help them in words, similar examples from lectures and labs, and diagrams.

If I suspect a case of plagiarism or cheating, I will notify the student via email and allow the student to come in and explain what happened. If I determine that plagiarism or cheating has taken place, the following possible sanctions will occur (in accordance with UNI Academics Ethics Policies found at http://www.uni.edu/policies/301). The following list does not list all possible academic ethics violations, and it is your responsibility to be familiar with the full list (again, http://www.uni.edu/policies/301).

Policy Violation	Examples	Possible Sanctions
Level One	• Working with another student on a homework assignment or a lab when the instructor has not explicitly authorized collaborative work. • Failure to properly cite once in a paper, programming assignment, or project.	• Written warning in email. • Giving no credit for the assignment; course grade determined in the usual manner. • Reprimanding the student in writing in the form of a letter (permanent record, not visible to others).
Level Two	• Failure to properly cite more than once in a paper, programming assignment, or project. • Copying on an examination. • Giving unauthorized assistance to someone during an exam.	• No credit for the assignment; course grade determined in the usual manner. • No credit for the assignment; reduction in course grade. • Reprimanding the student in writing in the form of a letter (permanent record, is visible to others).
Level Three	• Using prohibited materials during an exam. • Acquiring or distributing exam questions from an unauthorized source. • Acquiring or distributing an exam answer key from an unauthorized source • Plagiarism on a large class project or assignment that affects a major or essential portion of work done to meet course requirements or else assisting others to do the same.	• Disciplinary failure for the course. (This will appear on the student's transcript.) • Reprimanding the student in writing in the form of a letter (permanent record, is visible to others).
Level Four	• Taking an exam for someone else or having someone else take an exam for you. • Repeated lower level violations such as fourth Level One, third Level Two or second Level Three violation.	• Permanent expulsion from the University and a notation of "academic disciplinary separation" on the student's transcript. • Reprimanding the student in writing in the form of a letter (permanent record, is visible to others).

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.

One Last Resource

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