UNI CS 1510 (Fall 2023)
Introduction to Computing, Sections 01–06

Course Syllabus

Course Information

Course Name and Number

CS 1510 Introduction to Computing

Meeting Times

Lecture (Sections 01-03):	MWF	10:00am–10:50am	Wright 009
Lecture (Sections 04-06):	MWF	11:00am–11:50am	Wright 009
Lab (Sections 01, 04):	Th	8:00am–9:50am	Wright 112
Lab (Sections 02, 05):	Th	10:00am–11:50am	Wright 112
Lab (Sections 03, 06):	Th	3:30pm–5:20pm	Wright 112

Contact Information

(Lead Professor): Sarah Diesburg
Office: East Bartlett 39 (within Suite 19)
Email: cs1510-help@uni.edu
Office hours: Zoom only. Schedule here: https://tinyurl.com/yeysxdy3
Class website: Available through UNI eLearning: https://elearning.uni.edu
(Lab Instructor): Mark Jacobson – mark.jacobson@uni.edu

Credit Hours

4 Credit Hours (3 for lecture, 1 for lab)
This course meets the Course Credit Hour Expectation outlined in the Course Catalog. **Students should expect to work approximately 2 hours per week outside of class for every course credit hour.**

Course Materials (Required)

Lecture notes (posted on the class Web site)
Follow these directions to obtain access to your mandatory online textbook:

- Sign in or create an account at http://learn.zybooks.com
- Enter zyBook code: UNICS1510DiesburgFall2023
- Subscribe

Course Description

Introduction to software development through algorithmic problem solving and procedural abstraction. Programming in the small. Fundamental control structures, data modeling, and file processing. Significant emphasis on program design and style. (Fall and Spring)

Course Learning Outcomes

1. Students can analyze a programming problem and create a programming solution using the basics of programming:
    a. actions (input, processing, output)
    b. control structures (selection, repetition)

c. basic data structures (strings, arrays/lists, hash tables/dictionaries)
d. modularization (student-written procedures/functions)
e. data persistence (file input and output)

2. Students will increase their problem solving skills by creating programming solutions to solve real-world problems.
3. Students will increase their communication skills by writing readable programs suitable for collaboration with others.

Performance Evaluation

Grade Determination

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

Note: In-person programming exam may be merged with the concept exam if UNI classes are moved completely online.

Activity	Quantity	Points
Lab Work	12 @ 10 pts each	120
Individual Homework	10 @ 25 pts each	250
Concept Exams	125 and 150 points	275
Programming Exams	2 @ 125 pts each	250
zyBook Participation Activities	15 @ 7 pts each	105
Total		***1000***

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

Grading Scale

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

If you miss a class due to illness, quarantine, or any other reason, it is your responsibility to find out what was covered by watching the recording of the live Zoom lectures (link found on elearning). If you must miss an in-persona lab, you must let us know as soon as possible so that you do not fall too far behind.

ZyBook Participation and Challenge Activities

Your textbook is an interactive online textbook. You will be graded for completing participation activities before class on the day they are assigned. Again, participation activities are due *before class*. Challenge activities are not graded. They are good practice, and we will often go over them in class as sample problems.

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.

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, and will be accepted up If you must turn in an assignment late to due sickness or other approved reason, you should let us know as soon as possible so that you don't fall behind.

Description of Assignments

- Create a series of basic programs that (1) ask for input, (2) make calculation(s), and (3) display output.
- Create a series of programs that use selection statements to make choices based on input.
- Create a series of programs that use looping (repetition) to solve problems.
- Create program(s) that use nested looping (repetition) to solve problem(s).
- Create program(s) that demonstrate the use of advanced string functionality to solve problem(s).
- Create program(s) that illustrate the use of functions/procedures to solve problem(s).
- Create program(s) that read data from an external file, use data manipulation, and write output to a new file.
- Create program(s) that use lists/arrays as an internal data structure to solve problem(s).
- Create program(s) that use dictionaries/hash tables as an internal data structure to solve problem(s).
- Create a program design document for a program that involves many of the above components to solve a problem.
- Create programs(s) that use many or all of the above elements to solve problem(s).

Homeworks will by default be assigned after the lab period for which the new concept will be introduced. Please consult the main website for exact homework due dates and late policies.

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.

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.

Tentative Schedule (Subject to Change)

Date	Topics/Notes
8/21– 8/25	Computer Basics and Background
8/28 – 9/1	Intro to Basic Python
9/6– 9/11	Booleans, Conditionals, Selection Statements

9/13 – 9/25	Repetition
9/27 – 10/9	Basic Strings
10/11	Concepts Exam #1
10/12	Programming Exam #1 in WRT 112
10/13 – 10/16	File Basics
10/18 – 10/23	Functions
10/25 – 11/3	Basic Lists
11/6 –11/10	Dictionaries
11/13 – 12/1	Sets, Searching & Sorting Algorithms
12/7	Programming Exam #2 in WRT 112
Finals	10am class on Monday, December 11 from 10-11:50am 11am class on Tuesday, December 12 from 10-11:50am

Computing Environment

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

- **Wright 110 & 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).
- **EBAR Suite 19** – 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 4pm on weekdays (or when the last faculty member leaves).

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.

Scholastic Conduct

All three policies listed below come down to one thing: **If you can't explain code you turn in for homeworks or labs, it doesn't count.**

In addition, if I find that you have copied code from someone else, use unauthorized resources during exams, or otherwise attempt to subvert exam/homework/lab rules, you will be subject to violations of the UNI Ethics Policy, which include possible sanctions.

AI Code Generation Class Policy

Artificial Intelligence (AI) is are rapidly changing the face of software engineering. AI tools are rapidly developing that can generate full programs based on a prompt (e.g. ChatGPT and Bard) to being a "copilot" and suggesting lines of code as you are writing. These tools hold great promise to increase our productivity as programmers.

As software developers, we are tasked with the incredibly important job of making sure code we use or produce is correct, free from errors, and does not introduce security issues. To properly use these AI tools

in the future as professionals, we first need to understand how code works, inside and out. ***Only once we understand how to code can we read, test, and properly evaluate code generated by AI tools.*** Therefore, the purpose of this class is to teach you how to fully understand the foundations of coding by creating introductory-level programs to solve problems ***without the use of AI.***

The following policies apply to this class:

1. Do not use AI tools to generate programs for this class.
2. Using AI tools to create homework or lab solutions will put you at risk of a code review meeting with me and risk not understanding coding enough to pass the exams. Failing the exams will cause you to fail the class.
3. If I see advanced homework or lab solutions using coding that was not introduced in class, I will send you an email to meet with me within a week.
    a. If you have advanced programming knowledge and are able to explain the code you submitted, that's great! I'll learn a bit more about your previous experience and put you on a list not to bug again.
    b. If you cannot explain your code or do not meet with me within a week, the homework or lab assignment will turn into a grade of 0.

Seeking Help from other Internet Websites

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 usually better to try to figure things out on your own than to use something you don't understand.

Seeking Help from People

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 asking other people for help while 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).

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