CSCI 150
Foundations of Computer Science

Spring 2017
Lecture: MWF 1:10-2:00, MC Reynolds 110
Lab: Th 8:10-11:00, Snoddy Computer Lab
Website: `http://ozark.hendrix.edu/~yorgey/150/`
Course text: *Think Python* by Allen Downey
`http://ozark.hendrix.edu/~yorgey/150/ThinkPython-CSCI150-S17.pdf`

Instructor: Brent Yorgey, MC Reynolds 310
Office hours: any time my door is open, or make an appointment at
`http://byorgey.youcanbook.me`
Email: `yorgey@hendrix.edu`

Course Description

Introduction to computational problem-solving and computer programming. Topics include imperative programming constructs (variables, loops, conditionals, functions, recursion, file processing), basic object-oriented constructs (classes, objects), and some fundamental algorithms and data structures (dictionaries, arrays, linked lists, regular expressions). Students learn through studying the Python programming language.

Evaluation

Evaluation will be based on

- 25%: Weekly labs
- 35%: Three projects
- 15%: Quizzes, HW, and participation
- 5%: Midterm exam 1
- 10%: Midterm exam 2
- 10%: Midterm exam 3

Labs

Much of your experience with programming in this course will be through weekly labs, which will comprise 25% of your final grade. Lab attendance is required. Labs take place in the Snoddy Computer Lab, in the Bailey Library. Each lab

will be assigned on Thursday morning with time allotted to work through the materials, and will be due by the following **Tuesday at 10pm**.

Each student has four late days to spend throughout the semester as they wish. Simply inform me any time *prior* to the due date for an assignment that you wish to use a late day; you may then turn in the assignment up to 24 hours late with no penalty. Multiple late days may be used on the same assignment. There are no partial late days; turning in an assignment 2 hours late or 20 hours late will both use 1 late day.

On these labs, you may work with a partner on the lab assignments if you choose. Their name must be listed on any code you hand in as joint work. A partnership need only turn in a single copy of the assignment. If students working as partners wish to turn in a lab late, *both* students must use a late day.

Projects

You will have three projects in this course, one about every five weeks, for a total of 35% of your final grade (the first project is 5%, the second 10%, and the final project 20%). These projects will cover concepts we have discussed in class and in labs, and will be due approximately one week after they are assigned.

You must work individually on the first two projects. You may discuss concepts and ideas with your classmates, but the code you turn in must be your own. You will be graded not only on correctness, but also technique, documentation and evaluation of your solution. Further details on the grading standards and handin instructions for each project will be given when they are assigned.

Attendance and submission policies

Prompt lecture attendance is expected. Although I typically do not take formal attendance, unexcused absences may be reflected in your class participation grade. If you must be absent for some reason, please let me know in advance.

If you are absent from lecture (whether excused or unexcused) it is **your responsibility** to obtain notes from other student(s). Do not come to me and ask "what did I miss?".[1] On the other hand, if after obtaining notes you have specific questions or confusions regarding the topics covered, I would be happy to talk with you.

Disabilities

It is the policy of Hendrix College to accommodate students with disabilities, pursuant to federal and state law. Students should contact Julie Brown in

[1] I am likely to answer that you missed the part where that is your responsibility.

the Office of Academic Success (505.2954; `brownj@hendrix.edu`) to begin the accommodation process. Any student seeking accommodation in relation to a recognized disability should inform the instructor at the beginning of the course.

Academic Integrity

All Hendrix students must abide by the College's Academic Integrity Policy as well as the College's Computer Policy, both of which are outlined in the Student Handbook.

For specific ways the Academic Integrity policy applies in this course, please refer to the Computer Science Academic Integrity Policy.

The short version is that academic integrity violations such as copying code from another student or the Internet are **easy to detect**, will be **taken very seriously**, and carry a default recommended sanction of **failure in the course**.

If you have any questions about how the Academic Integrity policy applies in a particular situation, please contact me.

Course outline

A rough outline of the semester is as follows. This schedule may change. Please check the course website for corresponding readings from your textbook.

Week	Month	Topics
1	Jan	Introduction
2		Boolean logic, conditionals
3	Feb	Information encoding
4		EXAM 1, functions
5		While loops, strings
6		Lists
7	Mar	For loops
8		EXAM 2, recursion
9		File I/O, dictionaries
		Spring break
10		Hierarchies, debugging
11	Apr	Objects and classes
12		Objects and classes
13		Objects and classes, EXAM 3
14		Queues
15	May	Final projects

Learning objectives

By the end of the course you will be able to:

- Read, understand and execute a computer program written in Python.
- Read a set of requirements for a computer program in English, and write a short Python program (100 lines or less) that corresponds to them.
- Test a Python program and identify and fix programming errors.
- Identify some errors in a Python program without testing it.
- Without using a computer, write a very short Python code fragment (10 lines or less) that correctly implements a set of requirements.
- Understand and apply variables, loops, strings, lists, conditionals, and functions.
- Write programs to perform mathematical calculations.
- Understand the concept of a module.
- Write a Python program that is separated into at least two modules.
- Understand the concepts of class and object, and distinguish between them.
- Write a Python program including objects of at least one student-designed class.
- Write and understand appropriate comments in a Python program.
- Understand the concept of an algorithm and compare the efficiency of different algorithms for a simple task.