Course Syllabus

Welcome to CSCA20: Introduction to Programming

How will this course operate?

We want to offer you lots of opportunities to learn in whatever ways work best for you. With this in mind, we have split the course into a small number of mandatory components, and many optional components.

Mandatory

Lectures: We will be presenting material and building code together during lecture. You can watch the web-option after the fact, but you will get a better experience by participating live. In any case, you are responsible for any material presented during lecture.

Tutorials: This is where you will work with TAs to demonstrate your knowledge. Labs will be completed in tutorial section, so attendance is mandatory. If you can't make your assigned tutorial section on a given week, you will need to get in touch with a member of the teaching team to arrange an alternative opportunity for you to demonstrate your lab.

Optional

Readings/Videos/Links: We don't have an official course textbook. This does not mean that the links we post to readings won't be helpful, but rather that you can decide which works best for you. Most students will need at least some external guidance, but videos will work better for some, interactive problems for others, formal readings for still other students, try them all and see which one(s) suit your learning style.

Practice Problems: We won't be marking these formally, but they are good practice and good opportunity for you to evaluate how much you're getting from the lectures/readings/videos. The TAs will be more than happy to work through them with you if you need help.

What will we be covering?

In this course, we will cover fundamental programming concepts in Python, including:

- loops
- selection
- file input/output
- strings, lists, sets, and dictionaries
- 3rd party tools including matplotlib and numpy
- SQL and database interfaces

How will you be evaluated:

This course doesn't use a traditional "bucket of points" model where each piece of work has some weight, and your final grade is how many points you've accumulated. Instead, we will be using a competency/mastery model, where you will be rewarded based on the set of components in which you've shown competency and/or mastery.

You will be evaluated on the following sets of components:

- Core components:
  - for loops
  - while loops
  - selection
  - strings/lists
  - functions
  - sets/dictionaries
- Advanced components:
  - File input/output
  - CSV files
  - 3rd party tools
  - SQL Databases

You will have multiple opportunities to demonstrate competency or mastery of each component, including:

- Weekly labs (6 total)
- Term tests (2 total)
- Project
- Final Exam

The grading scale for each component is as follows:

- N = Not Yet (more work is required to demonstrate competency in this component)

- N = Not Yet (more work is required to demonstrate competency in this component)
- C = Competency (student has demonstrated competency in this component)
- M = Mastery (student has demonstrated mastery in this component)
- E = Extension (student has gone beyond basic requirements of the component showing initiative and passion in extending their work)

So how does this translate to grades?

We want you to focus on learning instead of grades, so we've set benchmarks for completion rather than direct grades. However, we also get that students want to know how their grades will be computed, so here we go:

Base Grade:	Minimum Requirement to guarantee base grade
50%	Demonstration of C on all core components
60%	Demonstration of C on all core and advanced components
70%	Demonstration of M on all core components, C on all advanced components
80%	Demonstration of M on all core and advanced components
>80%	Grades above 80 will be based on Demonstration of extension which will be primarily available from the project, term tests and exam

Your base grade represents the minimum grade you will receive for completing the given requirements. Your grade can increase from the base grade by adding additional evaluations.