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 and quizzes 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.

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:

- fundamental programming structures (selection, loops, functions)
- core data structures (variables, lists, strings, dictionaries)
- user and data interaction (user I/O, files, SQL)

- user and data interaction (user I/O, files, SQL)
- 3rd party tools including matplotlib and numpy

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 mastery model, where you will be rewarded based on the set of components in which you've demonstrated mastery.

You will be evaluated on the following sets of components:

- Foundational components:
  - Basic user I/O
  - Simple variables
  - For loops
  - While loops
  - Selection
- Core components:
  - Functions
  - Combining loops and selection
  - Nesting loops
  - Manipulating strings/lists
  - Looping over lists
- Advanced components:
  - Sets/Dictionaries
  - File input/output
  - Creating Databases
- Extension components
  - Complex data structures
  - 3rd party tools
  - SQL Querying

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

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

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 all foundational components
60%	Demonstration of all foundational + core components
70%	Demonstration of all foundational + core + advanced components
80%	Demonstration of all foundational + core + advanced + extension components
>80%	Students who demonstrate all components will be offered the opportunity to complete a project worth up to 20% of their final course grade

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.