

CSC110 Lecture 32:

Wrapping Up and Looking Ahead

David Liu and Tom Fairgrieve, Department of Computer Science

Navigation tip for web slides: press ? to see keyboard navigation controls.

Announcements and Today's Plan

Announcements

- Please complete the [PythonTA Survey 2](#)
 - Due December 8
- Final exam info has been [posted](#)
 - Regular office hours end after today
 - Additional office hours will begin on Friday
- Please complete our [Course Evaluation](#) to help make CSC110 better next year!



JOIN US FOR

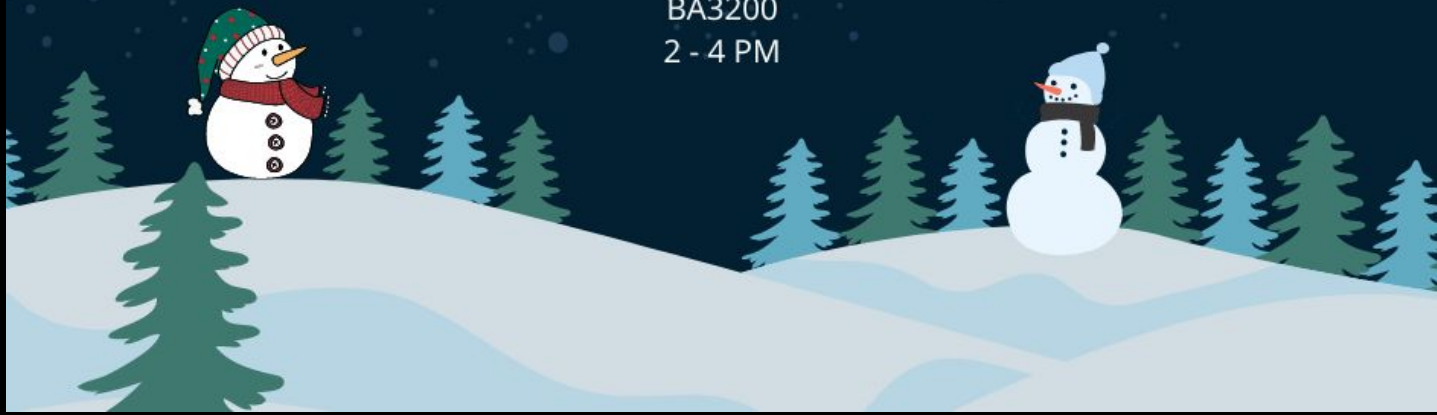
DEC EXAM DE- STRESS EVENT

COME FOR SOME HOT COCOA, GINGERBREAD
DECORATING, AND GAMES!

WEDNESDAY, DECEMBER 7, 2022

BA3200

2 - 4 PM



Today's objectives

1. Review information about the final exam
2. Share some [study strategies](#) and [test-taking strategies](#)
3. Review some big takeaways from CSC110
4. Look ahead to CSC111

Final Exam Info

Final Exam Info Page ([Quercus link](#))

Location and time: see [Faculty of Arts & Science webpage](#)

- find the building on a map [before](#) the day of the exam

Cover page and reference sheets have been posted!

Additional Office Hour Schedule has been posted!

On the day of the exam ...

- bring pens/pencils and eraser (with some backup)
 - no pencil cases on desk
- bring your T-card, but ...
 - can use Passport, Driver's Licence, Canadian Health Card
 - the worst that can happen is that they ask you to fill out form
- plan to arrive 15 minutes early
 - but if you are late, speak to the exam presiding officer. There may be options.
- Tom's hypothesis:
 - the larger the water bottle, the earlier the bathroom request

Test content

- Covers Chapters 1 to 11—the entire course!
- Lecture 31 (yesterday) is not covered
- Questions will target all parts of the course
- Slight focus on Chapters 9-11; Weeks 8-11

Test format

Very similar to term tests:

- **eight** questions total
- mixture of short answer, programming, proof, and running-time analysis questions
- vary in difficulty, but overall a bit more challenging than the term test questions—[assessing a higher level of mastery of course material](#)

Advice: Studying for the final exam

1. When **reviewing material**, be active!

- Take notes, summarize, write down questions, make connections to other concepts

2. When **doing practice questions**:

- If you get stuck, write down the progress you've made and what you're stuck at, and ask for help
- If you finish a question, practice checking your work
- Try to identify **patterns** and **structure** in questions—and then create your own

3. Practice completing questions in a test-like environment

- Practice writing code on paper
- Use a watch and keep track of time

Advice: Taking the final exam

1. Get a good night's sleep, and eat well before the exam.
2. Treat the test paper as a **priority queue**, not a queue.
 - Skim the entire test and identify the questions you feel most confident in, and do those first. If you had worried about a particular question being on the exam, the mystery will be over.
3. Bring a non-smart watch and keep track of time during the test.
 - Be careful not to spend too much time on any one question!
 - Don't count on there being a clock that is visible in the exam room.
4. If you get stuck, don't be afraid to move onto the next question.
 - You can still get partial credit for an incomplete solution.
 - You can come back and add more to your solution later.
 - Your brain will keep working on the problem while also solving the next one.

Big Takeaways from CSC110

The four “layers” of a computer program

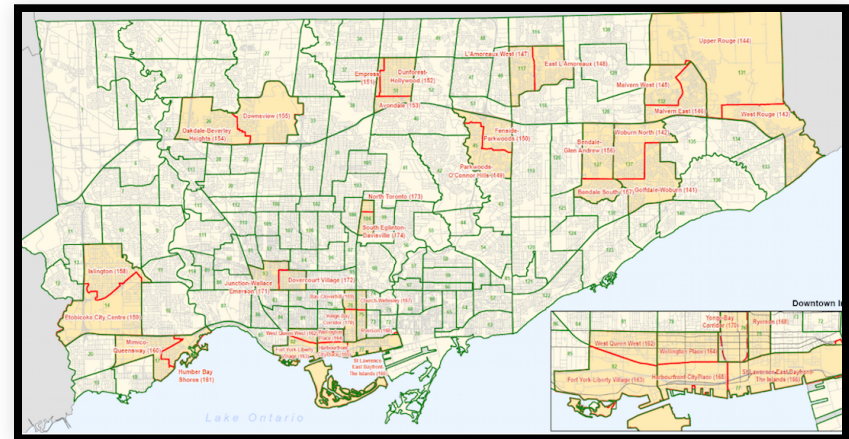
Data

Algorithms

User interface

Social impact

Computing with data (what we've done)



Computing with data (what's to come)

Data is everywhere!

At U of T:

- STA130 and the [Data Science Specialist program](#)
- Look for computational positions at research labs across U of T

Online:

- Look for [“X” datasets](#) or [“X” API \(Application Programming Interface\)](#)
- Shout out: [Data is Plural](#) weekly newsletter

Developing technical skills:

- Data science Python libraries: numpy, pandas, SciPy, plotly

Processing Data: Algorithms (what we've done)

Iteration	x	y
0	124124124	110
1	110	14
2	14	12
3	12	2
4	2	0

1. g, p

2. $a; A = g^a \% p$

3. $b; B = g^b \% p$

4. $k_A = B^a \% p; k_B = A^b \% p$

1. $(n, e), (p, q, d)$

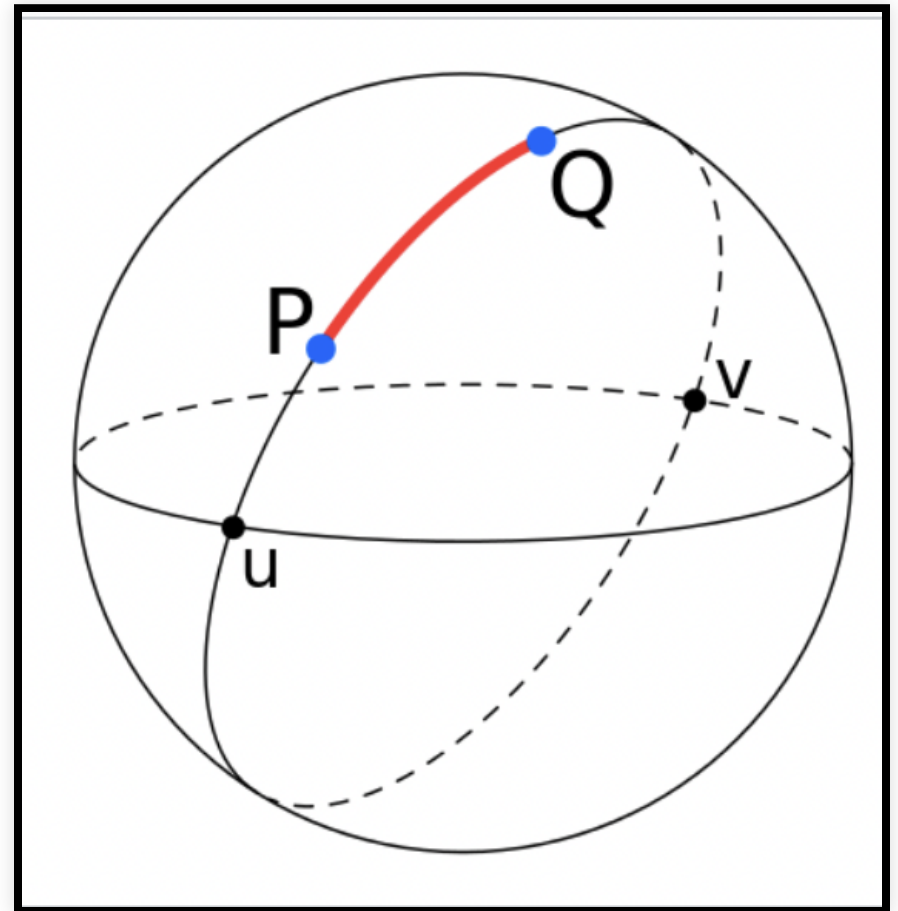
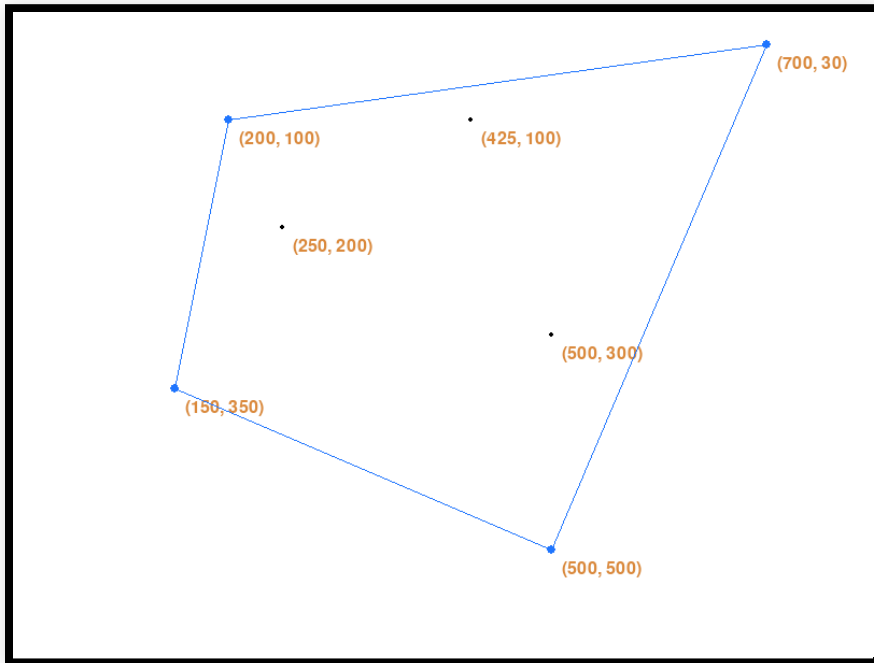
2. $c = m^e \% n$

3. $m' = c^d \% pq$

And the ever-present:

- `for item in <collection>:`
- `for i in range(0, len(<collection>):`

Processing Data: Algorithms (what we've done)



Processing Data: Algorithms (what's to come)

Every area of computer science requires complex algorithms:

- Artificial intelligence and machine learning
- Computer graphics
- Scientific computing and numerical modeling
- Any time you want to store and analyze data efficiently

Many future courses will explore these domains!

Programming in Python

Data: data types, literals, operators, comprehensions, data classes; object mutation

Functions: built-in functions and methods; defining functions

Control flow: if statements, for loops, while loops; raising exceptions

Modules: import statements; `math`, `datetime`, `random`; `doctest`, `pytest`, `hypothesis`, `python_ta`

Class design: defining an initializer and other methods; inheritance

But writing professional software isn't just about knowing how to write code...

From programming to software engineering

“Software engineering” encompasses not just the act of writing code, but all of the tools and processes an organization uses to build and maintain that code over time. What practices can a software organization introduce that will best keep its code valuable over the long term? How can engineers make a codebase more sustainable and the software engineering discipline itself more rigorous?

–Software Engineering at Google: Lessons Learned from Programming Over Time (Titus Winters, Tom Manshreck, Hyrum Wright)

Software Engineering (what we've done)

You've gotten practice with many stages of the software development process:

1. **Designing** data types and functions for a program.
 - Creating **logical specifications** for data types and functions (preconditions, representation invariants)
 - Breaking down a problem domain into various classes and functions (food delivery)
2. **Implementing** these data types and functions in Python.
 - Using programming building blocks
 - Learning some libraries

Software Engineering (what we've done)

3. Debugging and testing your code.

- Using the PyCharm debugger
- Using `doctest`, `pytest`, and `hypothesis`

4. Analysing your code...

- for efficiency (running-time analysis)
- for good code design and style (`python_ta`)

Software Engineering (what's to come)

Upper-year CS courses will go deeper into:

- software design and specification (e.g., CSC207, CSC301, CSC410)
- programming techniques, patterns, and programming language design (e.g., CSC324, CSC367)

Many of these courses are **project-based**, so you'll get experience with general software engineering skills:

- **communication!**
- breaking down a project into tasks and milestones
- using version control to track changes and collaborate with teammates on shared code

Mathematical logic and proof

Mathematics gave us tools for **precisely representing data and reasoning about code**:

- Preconditions and representation invariants
- Loop invariants and assert statements
- Big-O/Omega/Theta notation, running-time analysis

Mathematical theory and proofs formed the basis of several algorithms:

- Formulas like Euclidean distance
- Prime number checking
- The Euclidean Algorithm
- Cryptosystems

Looking Ahead to CSC111

CSC111 at a glance

CSC111 is a direct continuation of CSC110.

- A regular half-course: 3 lecture hours and 2 tutorial hours per week
- Less work, but topics are more advanced
- Same overall structure and themes as CSC110
- Builds on material from CSC110

Topics include:

- Designing and implementing new data structures (linked lists, trees, graphs)
- New algorithms and algorithm analysis (including sorting!)
- Induction (the proof technique) and recursion (the programming technique)
- Digging deeper into the Python programming language

If you are taking MAT157 and MAT240 ...

- Consider taking [CSC240](#) next term with Prof. Faith Ellen
- Enriched Introduction to the Theory of Computation

Questions? Comments?

CSC110 is coming to an end ...
... but you'll see each other again in CSC111!
And ... introducing ...



Mario Badr will join with David Liu to teach CSC111 !!!

The End 🥲

- Good luck with the final exam!
- Please stay in touch:
 - Drop by BA 4230 to tell me how you are doing!