CSC 148: Introduction to Computer Science

Week 12

Odds & ends, wrap-up and reminders

Exam review

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Wrap-up and reminders

You are no longer a novice programmer

- You have written significant Python code and covered a broad range of practical experience, including:
  - Working with code across multiple files and classes
  - Using code-writing tools like PyCharm and python_ta
  - Testing and profiling tools (pytest, hypothesis, etc.)
  - Useful Python libraries (e.g., pygame)

Debugging

- Main idea
  - Follow the logic of your code, step by step
  - Analyze if the behaviour of your code is as expected
- Important skill
  - Especially for complex code
  - Gets you out of tricky spots
  - Better alternative to using print statements
    - Logging in general does have benefits though …

Testing

- Important: test-as-you-go!
- Write meaningful tests
  - This is a very important skill!
  - Consider corner cases
  - Your code must gracefully handle all exceptional cases instead of crashing
  - Remember preconditions and representation invariants!
- Re-test your code on every revision
  - Whenever you make substantial changes, re-test that everything that used to work still does

Documentation

- We pester you to write good docstrings for a reason
- Communicate what your code does and how it is intended to be used
  - What is assumed/implied?
  - What is the expected result/outcome?
  - NOT how it is done internally!
  - Remember interface vs implementation!

Code Aesthetics is Communication

"Programs must be written for people to read, and only incidentally for machines to execute."

Harold Abelson

Documentation is Communication

"The most important single aspect of software development is to be clear about what you are trying to build."

Bjarne Stroustrup

Software Design is Communication

“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.”

Sir Charles Antony Richard Hoare

Exam Logistics

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Exam Date, Location

- When and where:
  - https://student.utm.utoronto.ca/examschedule/finalexams.php
- Check that you know the room and how to get to it, in advance!
- Make sure to go to the right room!
- Exam starts on the hour
  - Plan to be there 15-20 minutes ahead

Exam Coverage

- Review the handouts and labs from each week …
    - Memory model
    - Testing
    - Object-oriented design
    - ADTs, Linked lists
    - Recursion
    - Trees (of all kinds)
    - Complexity, etc.
- The preps, reading, and assignments are also useful.
- Practice your skills. Don’t just read!

Aim for Mastery

- "Mastery" means that you are the expert.
- Go beyond "re-reading and re-doing":
  - create multiple solutions
  - identify common mistakes or errors, and
  - explain ideas to others
  - make up new questions

Tips

- Do not panic! Take a deep breath, you've got this!
  - This is your chance to show us what you've learned
  - We WANT to give you the credit that you've earned
- Read carefully!
  - What is the question asking?
  - Don't confuse things
  - If you find something unclear, read it carefully again and answer the question to the best of your ability!
- Keep track of your time
  - Some questions take more time than others
  - Do not spend too much time on a question if you are stuck – might want to revisit it later