CSCI111 Final Exam Prep

Topics

Everything through the second exam. Cumulative. A little more focus on topics since the second exam, but everything is fair game.

Object-oriented programming

- Benefits, use
- Developing classes – what is our process?
    - instance variables
    - Representing new data types
    - Special methods
        - `__init__`
        - `__str__`
        - `__lt__`
        - `__eq__`
    - other methods, helper methods
- Terminology (not already mentioned above)
    - Instance of (as opposed to instance variables)
    - Overriding
- Testing defined classes
- Using others' defined classes

Search techniques

- Linear search
- Binary search

Exception handling

Lists

- 2D lists – accessing, processing

Recursion

Programming language characteristics

What is Computer Science? What are fields in CS?

What I expect from you on exam:

- To know the Python/programming terminology
    - E.g., names for types of statements
- To know the appropriate Linux commands and how to use them, given a typical situation from lab
- To be able to read a program and describe what the program is doing at a high level in plain English (comments), trace through the program's execution given input (control flow), and say what the program outputs
- To be able to write a program (given an algorithm or creating your own algorithm, given a problem)
- Syntax must be very close to correct (correct keywords, indentation, special characters, variable naming, operations)
- Greater emphasis on ability to read and write code

Suggestions on how to prepare:

- Practice programming on paper and verify program in Python. (Use problems from class, labs, or textbook.)
    - What types of problems should you focus on?
- Practice reading through programs, tracing through them, and saying what the output should be
    - The interactive book is helpful for showing you what happens when you run a program. Try to determine what happens first, before looking at what actually happens.
- Read through slides for vocabulary, review questions, and non-problem-solving exercises
- Do the practice/interactive exercises in the textbook. They are helpful!
- Create your own classes
    - Example classes: `Deck`, `Student`, `Course`, `BankAccount`
    - What are other good things to make into classes?
- Using classes you've defined
    - Create a Card game (war is an example)
- Review Linux commands

Exam will be on Canvas

- Be extra careful about writing code—paying attention to indentation, capitalization, etc. Beware of autocorrection to something you didn't mean to type.