CSCI111 2nd Exam Prep

General Topics

Everything up through the first exam (necessarily cumulative)

Strings
- representation (ASCII)
- common, useful methods, operations

Lists
- creating, accessing, processing
- common, useful methods
- similarities, differences to strings

Files
- creating file objects
- reading and writing files
- handling numbers
- common methods

Functions
- use, benefits
- defining your own
- formal, actual parameters (input to function)
- returning output from function
- using functions you've defined
- variable lifetime/scope
- use of None
- default values for parameters
- testing functions

Documentation
- documentation strings, appropriate comments for functions

Development Approaches
- Bottom-up design
- Refactoring
- Top-down design

Creating modules

Indefinite loops
- syntax
- similarities, differences to for loops

Dictionaries
- creating, accessing, processing
- common, useful methods
- similarities, differences to lists

What I expect from you on exam:

- To know the Python/programming terminology
- 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, 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) or class
  - Syntax must be very close to correct (correct keywords, indentation, special characters, variable naming, operations)
  - Since it's on paper, there is some leniency—you may mark up your exam somehow if, for example, something should be indented
  - No need for constants or comments on an exam *unless specifically requested*

Suggestions on how to prepare:

- Practice programming on paper and verify program in Python.
  - Use problems from class, labs, or textbook.
- 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. Make sure you try 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!
- Use techniques on other problems. For example,
  - Refactoring code to use functions (lots of problems where you could "functionalize it")
  - Writing test functions (practices functions and lists!)
  - …