CSCI111 Exam 1 Study Guide

Topics

Introduction to Computer Science

- algorithms, programs
- programming languages – characteristics, motivation for

Programming Basics

- process of writing and executing Python programs
  - interactive and batch mode, interpreter
- Python keywords
- data types
- variables, identifiers, constants
- numbers and arithmetic operations
- float vs integer division
- operator precedence
- calling functions
- importing modules and using functions from modules
- data type constructors (converting between data types)
- testing, debugging
- Style: good variable naming, readability

Object-oriented programming

- Constructors – creating objects
- Calling methods on objects
- Using APIs

Control Structures

- `for` loops (how does `range` work?)
- accumulator design pattern

Functions

- use, benefits
- defining your own
- formal, actual parameters (input to function)
- returning output from function
- using functions you've defined
- variable lifetime/scope
- testing functions
- putting your own functions in modules

Documentation

- documentation strings, appropriate comments for functions

Development Approaches

- Bottom-up design
- Refactoring

Linux

- terminology
- basic commands
- file structure

What I expect from you on exam:

- To know computer science, Python, and programming terminology
    - E.g., names for types of statements
- To know 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 know how to read/understand/use the graphics API
- 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)
    - Since the exam is on paper, there is some leniency—you may mark it up somehow if, for example, something should be indented
    - No need for constants or comments on a timed exam (unless explicitly requested)
- To be able to explain concepts clearly/concisely as if you were interviewing for a job, and you needed to make it clear that you understood the concept to be hired.

What I do *not* expect from you:

- to memorize the Graphics API
- to write comments—unless they help you or unless otherwise specified

Suggestions on how to prepare:

- Review the many in-class exercises, handouts, and review questions
- Practice reading through programs, tracing through them as the computer would, and saying what the output should be
- Practice programming on paper and verify program in Python.
  - Use problems from class, labs, and textbook. There are problems that we did not complete in the slides.
- Read through slides for vocabulary and non-problem-solving exercises
- Use alternative development approaches to solve a problem
  - For example, refactoring code to use functions (lots of problems where you could "function-ize it"). Then, as appropriate, create a test function with good test cases using `test.testEqual`.
- Review Linux commands and common scenarios

Some Example Problems to Solve:

- Draw 4 horizontal green lines equal distance apart in a 400x400 window
  - Modify your code to allow different numbers of lines (perhaps getting as input from a user)
- Allow the user to move the circle multiple times (with or without animation)
- Allow the user to pick the points to draw a line
- Use the Pythagorean theorem to calculate the length of the long side of a right triangle ($a^2 + b^2 = c^2 \rightarrow$ c=$\sqrt{a^2 + b^2}$)
  - Make a function that computes it. Make a main function that gets user input and displays the result.