CMSC 201 Section 40 Spring 2020 Exam 1 Study guide

This study guide covers *the material you are responsible for on the first exam*. This guide is expressed in terms of skills you must be able to demonstrate on the exam.

The exam will consist of:

- Multiple choice and/or true/false questions
- Short answer questions
- Code analysis problems – you will be given short segments of Python code. For some segments, you will be asked to identify bugs that stop the code from doing what it is supposed to. For other code segments you will be asked to describe the output/result of the code
- Code writing problems – you will be given short pseudocode descriptions of solutions, and you will be expected to write valid Python code that implements that pseudocode.

*The order, length or depth of a section in this study guide is **no indication** of the relative importance of that topic or its likelihood to appear on the exam.* i.e. Don't ignore the short sections!

The contents of this document are not final until the exam review session of the course. I can change this between now and Monday, the 24th!

General

- *Define, compare and contrast* the three types of program control:
    - Sequential
    - Conditional
    - Iteration/Looping

Variables

- *Define* what a variable is and what it is used for in programming
- *Compare and contrast* instantiation and assignment
- *Identify* legal and illegal variable names in Python
- *Enumerate* the rules for variable naming in Python
    - NOTE: You ***are not*** responsible for memorizing any reserved words that have not yet been explained in class.
- *Enumerate* the following variable types and give examples of values
    - String
    - Int

    - Float
    - Boolean
- *Define* `int()` and `float()` and explain their uses
- *Define* `print()` and `input()` and explain their uses
- *Identify* the *output* of `input()`, i.e. what does it give you?
- *Define* literals, and provide an example literal for each of the four data types

Operators

- ***Resolve*** **an expression (including any of the operators discussed in class) following the proper order of operations.**
- *Enumerate* the arithmetic operators and explain what they do
- *Compare* and contrast integer division (//) and float division (/)
- *Solve* simple modulo expressions. They will be simple enough that you will not need a calculator
- *Enumerate* the order of operations of all operators covered in this guide
- *Explain assignment* and enumerate five assignment operators covered in class
- *Identify* expressions that are legal on the left and right sides of an '='. E.g. Is `x + y = 3` a legal python statement? Why or why not?
- *Explain comparison* and enumerate six comparison operators discussed in class.
- *Compare* and contrast `=` and `==`
- *Enumerate* the integer, float and string values that will evaluate to `True` and those that evaluate to `False`
- *Define* the three logical operators, **`and or`** and **`not`**
- *Complete* truth tables for the logical operators

Conditionals[1]

- *Define* the reserved words, `if`, `elif`, and `else`.
- *Resolve* if statements, including
    - `if` statements
    - `if/else` statements
    - `if/elif/else` statements
    - Nested `if (and/or elif/else)` statements
    - Any combination of the above
- *Explain* the difference between two consecutive `if` statements and an `if` statement followed by an `else` statement

[1] REMEMBER! The length of a study guide section is ***NOT*** indicative of how likely it is to be covered on an exam.

Algorithmic Thinking

- *Define* pseudocode
- *Describe* a problem solution using pseudocode
- *Implement* Python code for an algorithm described in pseudocode

Lists

- Define *lists* in Python and *explain* their usefulness
- *Initialize* empty lists and a list of starting values
- *Implement* code that:
    - Adds elements one at a time to an existing list
    - Removes elements one at a time from an existing list
    - Retrieves an element given that element's index
    - Update the value of an element given the element's index
    - Retrieves the length of a list
    - Determines whether a list has a particular item
    - Loops through a list using a loop and perform one or more operations on each element

For loops

- Define range() and explain its three parameters
    - Start value
    - End value
    - Hop size
- Explain what the defaults are for the start value and hop size when range is only given one or two parameters. Examples:
    - range(1, 3)
    - range(10)
- Define **`for`** loops and contrast them with while loops
- Define "for i" and "for each" loops and compare and contrast them
- Explain what lines of code a for loop does "for" you that a while loop does not

While Loops

- *Define* the reserved word, **`while`**
- *Define, compare and contrast* missed loops (that is, loops that are never executed) and infinite loops
- Given a loop, *describe* its behavior, and *count* how many times it will execute