1 General Notes

- The midterm will take the entirety of the class period on the Thursday of week 4 (09/24).
- You can bring a page of notes (double sided is ok) to use on the test
- Everything in this document will be tested on the midterm, and nothing that is **not** on this document will be tested.

Example questions will be posted early in week 4. They will be a mix of code writing, true/false, code reading/prediction, and short answer formats.

To prepare effectively for the midterm, we recommend looking at the topics below, and trying to formulate *your own* example question that would test your understanding of the topic. This is the method we will use to prepare the test. If you are unsure whether your question is a good one for a given topic, ask us; if you are unsure how to pose a question in an area, it means you don't understand it well and will probably struggle with it on the test. Use that as a signal to ask for help. It can be especially effective to do this exercise with a study buddy.

2 Study Guide

In each of these topic areas, you are well-prepared for the test if you can answer these questions with an affirmative or otherwise exhibit your understanding.

2.1 Variables, Types, and Values

- Do I understand how variable assignment works?
  - Could I trace a sequence of assignment operations and predict what values each variable will have in the end?
- Do I know when to use each of Python's basic types (int, float, str, bool), and what sorts of values they can contain?
- Given one or two variables or values, can I figure out which operators (like `+`, `or`, or `*`) and functions (like `str` or `math.sqrt`) are valid to use with them?
  - Can I manipulate and simplify boolean expressions (e.g., that `(a and b) and b` is the same as `a and b`)?

- Can I predict the value and type that will result from a Python expression?

2.2 Functions

- Do I know how to define and call functions with zero, one, or many parameters?
- Do I understand Python's variable scoping rules, so that even if two variables have the same name I can distinguish them?
  - Given a series of assignments and function calls, can I predict what values variables will hold?
- Can I trace the execution of a program through a series of function definitions and calls, and predict the types and values of the outputs?
  - Do I know the difference between print and return, and how return influences control flow?

2.3 Modules

- Can I write programs that use built-in Python modules like random or turtle?
  - Do I know what the "dotted name" syntax (e.g., `math.sqrt`) does, and how it looks different depending on how the module is imported?
- Do I remember—and know how to use—important functions from the modules turtle, random, and math, as we have done in homework assignments?

2.4 Loops

- Can I write Python loops that iterate over a particular range of numbers?
- Can I write Python loops that iterate until a condition is met?
- Do I know what the break statement means and its effect on loop behavior?

- Can I trace the execution of a Python code snippet through loops, and potentially nested loops?
  - Can I predict how many times a loop will execute and what the contents of variables will be at each iteration?
- Can I write functions that use loops to accumulate an integer, boolean, float, or string value in one or more variables?
- Do I know when to use a for loop versus a while loop?

2.5 Conditionals

- Can I write Python code that does different things depending on a boolean condition?
- Do I understand how Python decides what to execute in a conditional with many `elif` branches?
- Can I predict which inputs would lead a Python program to follow one code path or another through conditionals, even nested conditionals?
- Do I understand how loops, return, and other Python constructs interact with conditionals?
- Given a conditional statement, can I recognize whether it's actually unnecessary and the same behavior could be achieved through other means like boolean operations?