Fall 2018

Midterm 1 Study Guide

Some tasks just can't be performed by computing machines, no matter how good the programmers or how powerful the hardware.

Algorithms

An algorithm is, in the simplest terms, a set of instructions for performing a task.

- In the briefest terms, what is an algorithm?
- In general terms, how does Hilbert's Decision Problem (Entscheidungsproblem) relate to limitations of computers?
- What are some factors that go into determining whether an algorithm is "good" or "bad"?

How long it takes and how much space

Variables and types

- What is the first thing that needs to happen to any variable in a Python script? **give a value.**
- What is the assignment operator?
- Python is dynamically typed. What does this mean in terms of variables and their values?
- What are the syntactic rules for naming variables? (i.e., what kinds of names will the Python interpreter allow?)
- What are the numerical types available in Python that we've used so far? How do they differ?
- What are the sequential data types we've looked at so far? What are some similarities between them?
- What is casting? How is it done in Python? Why/when is it necessary?
- What is mutability? What does it mean for a type to be immutable? Which types have we seen in Python are immutable? Which types are mutable? Does mutability affect assignment of values to variables? If so, how?
- What are some commonly used string methods?

Arithmetic and math

- What do the following operators do: `+`, `-`, `+=`, `/`, `//`, `%`, `*`, `**`?
- What does the increment operator look like, and how does it behave?
- What are two ways of finding a square root of a number?
- What needs to be done in order to use functions from the `math` library module?
- "Overloading" operators means having operators carry out different functionality in different contexts. The `+` and `*` operators are both overloaded in that they have different behaviors depending on the types of their operands. What are some examples of this?

Booleans and conditionals

- What are the possible values of a Boolean type?
- What operator should be used to assess whether an arithmetic equation is true?
- What is the difference between `==` and `is`?
- What do `<=`, `<`, `>=`, and `>` do?
- How are the `not`, `and`, and `or` operators related to Boolean values? What are these operators used for?
- What is the structur and meaning of an `if/elif/else` statement? Which parts of that structure are required, which are optional, and what rules govern their ordering? (For example, can the structure begin with `elif`? Can an `elif` block precede an `else` block?

Loops

- In what respect are `while` loops and `if` statements similar?
- What is necessary in order to use a `for` loop?
- What is a simple way to implement a `for` loop that repeats some action a fixed number of times?

- If a `for` loop begins `for n in “hello”:` how many times will the code in the following block execute? What will the value be of `n` each time the block executes?
- If a `for` loop begins `for n in zip(range(3, “hello”):` how many times will the code in the following block execute? What will the value be of `n` each time the block executes?
- Can a `for` loop contain another `for` loop? Are there any limits on what sorts of loops and conditionals can be nested in each other?

Lists, tuples, and generators

- What types of objects can be elements of lists? What restrictions are there, if any, on what types of values can populate the same list together?
- How can a specific element of a list be accessed?
- Lists and tuples are both ordered sequential data types. What is the most important difference between lists and tuples?
- What style of loop is a list comprehension closely related to?
- What does the `zip()` function take as arguments? What does it return? (Careful, it doesn’t return a list!)
- What do lists have in common with generators such as `range()` and `zip()`?
- What construction this an example of? `[ord(c) for c in "Python"]`
- What type of object does the code in the previous bullet point generate?
- Given the following 2d list (list of lists)
  ```
  my_2d_list = [[‘a’,’b’,’c’], [‘d’, ‘e’, ‘f’], [‘g’, ‘h’, ‘i’]]
  ```
  how specifically would you access the character ‘e’ in this data structure?
- How might you split a string representing a sentence into a list of strings representing words? (A naive approach is fine here).

Libraries and modules

- What functionality from the `sys` and `math` libraries have we seen so far?
- What is the syntax for bringing the functionality of a library into your script?

General Python knowledge

- Python is whitespace sensitive. What does this mean, and what is an example of this?
- What is a “block” of code in Python. What character indicates the start of a new block, and how is the code of a block identified as such?
- What is the programming “paradigm” used by Python? Is there just one?
- What style of capitalization is typically used in Python? What is it called?
- What is a REPL?

Programming practice

- Be able to read and write short programs that may use `input()`, `for` loops, `while` loops, `if/elif/else` conditionals, type casting functions, strings, lists, string & list indices, common string & list functions (like `len()`), list comprehensions, arithmetic operators, comparison operators, increment/decrement operators, `sys.argv` (including correctly importing the necessary library module), `main`, `split()`, `print()`, `range()`, and `zip()`.
- Remember, you’ll need to be able to use these constructions *from memory.* In this sense, you can think of an exam as comparable to a whiteboard programming interview, in that you’ll be expected to be able to do basic programming tasks without referring to external resources. One way to practice this is to try to recreate some of your homework and lab exercises from memory.