IC411 Python lab Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fall 2018 (AY19)

**Summary**: In this lab, you'll practice using Python for basic tasks. Those already familiar will find this very easy. Those new to Python will be introduced to coding tasks needed for the Scheduling lab.

**Learning Objectives**:

- Understand basic Python syntax and execution. Practice with lists, files, and objects.

**Estimated Completion Time**: 1.5-2.5 hours **Total Points**: 20 points

**Submission**: Submit mod.py and sub.py to the online submission site. No hardcopy required.

**Discussion**

Python is an interpreted language, which makes it easy to use cross-platform, like Java. Python's interpreter and runtime environment are built using C. Python commands can be run from the interpreter, or as .py files. The syntax is overall very compact and readable. Python uses *dynamic typing*, which means the programmer need not be concerned with explicit type declaration, but should be careful of unintended type inferences. Python also takes care of allocation and deallocation of memory for objects automatically. The Python interpreter will perform a basic syntax check before executing, but will not evaluate every line for errors, which means that errors in Python are often discovered at runtime, instead of through a compilation process.

There are two main branches of Python: 2.X and 3.X. Both are supported at present, but 3.X is the future of Python. There are occasional compatibility issues between the two, but none that will affect this lab.

Two common tasks we'll use for the scheduler lab (later) are file I/O and array (list) manipulation. These short exercises will help you practice those tasks.

**Tasks**:

Tutorial Review

Begin by exploring the official Python tutorial: <https://docs.python.org/3/tutorial/>

Focus on sections 1-4, 5.1 - 5.2, 6, and 7.

As an alternative, if you prefer, go through the "Python Basic Tutorial" here: <https://www.tutorialspoint.com/python/>

You can skip Tuples, Dictionary, Date & Time, and Exceptions.

**"Mod" (10)**

- Create a Python script called mod.py

- Have your script read in a text file called numbers.txt. Each line contains a number.

- Create three output files: mod0.txt, mod1.txt, and mod2.txt. Your script should write each number from numbers.txt to only one of these three output files, based on the number’s modulo 3 value. For example, the input number 300 from numbers.txt would get written out to the file mod0.txt. Numbers should be written out in the same order they appear in numbers.txt.

Points:

(4) Script executes without runtime errors

(4) Output data is all correct

(1) Script contains your name at the top.

(1) Appropriate coding style, and correct filenames used.

**"Sub" (10)**

- Create a Python script that reads in a series of strings and processes them. Your file should be called sub.py. It does the following:

- Read in a file called letters.txt. This file will contain a large number of rows, with one sequence of random letters on each line, terminated by a carriage-return.

- Your script should read through each sequence of characters to determine if that line contains the sub-sequence NAVY in it. Note that a sub-sequence is not the same as a substring. For example, the following character sequence contains the *substring* (and it’s a sub-sequence, too) NAVY:

HHFUIFHUIOHUIOFREW**NAVY**FUEQFHERUIWFHUWIFHWERLF

However, the following sequence also contains the *sub-sequenc*e NAVY:

QKOPSGLATJFBXEPFBDGPEF**N**DG**A**RWPAQQHBFH**V**ECMOCHX**Y**

The difference is that the letters of the *sub-sequence* need not be contiguous.

- If a particular line contains a NAVY *sub-sequence*, that line should be appended to your output file, navy.txt. But, before writing, your script should change all the N, A, V, and Y characters to lowercase (including additional N, A, V, and Y characters that did not participate in the subsequence match).

- After the script completes, navy.txt should contain, in order, all of the original lines of characters from letters.txt that contain a NAVY sub-sequence (with those letters lower-cased), but none that do not.

Points:

(4) Script executes without runtime errors

(4) Output data is all correct

(1) Script contains your name at the top.

(1) Appropriate coding style, and correct filenames used.

**Submission**

mod.py

sub.py

- Via the submission site (no hardcopies required)

- Please include your name in a comment at the top of each submitted file.

- Remember to cite (as code comments) any sources you used, including example code posted on the internet (add the URL).