SI411, Fall 2025 (AY26)

**Python Lab**

**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 soon.

**Learning Objectives**: Understand basic Python syntax and execution. Practice with lists, files, and objects.

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

**Lab Total:** 15 points (5 points for mod.py and 10 points for sub.py)

**Provided Files:** numbers.txt for Part 1, and letters.txt for Part 2.

**Part 0: Overview**

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.

Python 3.X is the current and future. Older versions of Python, numbered 2.X, are no longer supported and should not be used.

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.

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.

**Part 1: mod.py**

Your assignment is to create a Python script called mod.py that reads in a list of numbers and writes them to separate files depending on their value, modulo 3.

Specifically:

- 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.

- Submit mod.py to the submission site. Be sure to include your name in a comment at the top of the file.

Points:

(2) Script executes without runtime errors

(2) Output data is all correct

(1) Appropriate coding style including your name at the top, and correct filenames used.

**Part 2: sub.py**

Your assignment is to create a Python script called sub.py that reads in a series of strings and processes them to identify sub-sequences of letters. Specifically, it should do 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.

- Read through each sequence of characters to determine if that line contains the sub-sequence MIDS 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) MIDS:

HHFUIFHUIOHUIOFREW**MIDS**FUEQFHERUIWFHUWIFHWERLF

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

QKOPSGLATJFBXEPFBDGPEF**M**DG**I**RWPAQQHBFH**D**ECMOCHX**S**

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

- If a particular line contains a MIDS sub-sequence, that line should be appended to your output file, mids.txt. But before writing, your script should change all the M, I, D, and S characters to lowercase (including additional M, I, D, and S characters that did not participate in the sub-sequence match).

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

- TEST your code! Think about how to do this well. Historically, a non-trivial number of students have submitted a solution that they THINK is correct, but that does NOT actually work correctly on all lines.

- Submit sub.py to the submission site. Be sure to include your name in a comment at the top of the file. Also remember to cite (as code comments) any sources you used, including example code posted on the Internet (add the URL).

Points:

(4) Script executes without runtime errors

(4) Output data is all correct

(2) Appropriate coding style including your name at the top, and correct filenames used.