INST326 Python Style Guide

Style Matters

It might be tempting to think of programming as a purely technical exercise of assembling instructions in order to accomplish a particular task, but in fact writing code is as much a creative art as it is a skill. As such, it requires not only logic and precision, but also leaves room for a good amount of of creativity and style.

The leaders of the Python community have issued guidelines on various topics with the goal of promoting the health of the open-source community. These guidelines are known as Python Enhancement Proposals, or PEPs, and include a "Style Guide for Python Code," known as PEP8. Many of the rules below draw on PEP8, and certainly none of them contradict it. I would encourage you to familiarize yourselves with the entire document so you can write code more clearly, and keep your code in line with community best practices.

Course Style Guide

All Python programs are required to contain the following elements:

Header

A program header (contained within a block comment) which lists the student's name, directory ID, submission date, assignment name, and exercise number. See below for an example.

```
Name: I. M. Student
Directory ID: imstudent
Date: 2018-08-30
Assignment: Homework 1
```

Helpful Comments

Wherever you write a non-obvious algorithm or series of statements, include appropriate comments to explain the underlying reasoning. Meaningful variable and function names - any variables or functions that you create should be named according to purpose or use.

Functions and Classes

Unless otherwise instructed, please perform all meaningful computation inside functions or classes. Use of functions and classes facilitates testing and code reuse.

Global Variables

Global variables should only be used for constant values. Variables whose values will change over the course of the program should be defined within functions, methods, or classes and passed as arguments to other functions or methods as needed.

If Name Equals Main

Unless otherwise instructed, please put an

```
if __name__ == '__main__':
```

clause in your main scripts, and keep its contents to a minimum. This facilitates testing and code reuse. For more information, see: - https://docs.python.org/3/library/**main**.html and https://docs.python.org/3/tutorial/modules.html

Documentation Strings (docstrings)

docstrings are special strings that document scripts, classes, functions, and methods. Unless instructed otherwise, you are expected to include docstrings in every script, class, function, and method you write.

Readability

No individual line of code should exceed 80 characters. This includes comments and docstrings. If you are using Pycharm, you can configure it to show you where the 80-character point is. Go to Settings > Editor > Code Style. In the entry box labeled “Hard wrap at”, enter “80”. In the entry box labeled “Visual

guides", enter "80" again.) Use whitespace judiciously to separate and spread out segments of code so that programs are easy to follow.

Indentation

Use 4 spaces per indentation level (but any exception listed in PEP 8 is okay). Don't use tab characters. Most feature-rich text editors can be configured to insert four spaces when you press the tab key.

Imports

Use separate lines for imports from separate modules, for example:

```
import random
import sys
```

Don't import multiple modules on a single line:

```
import random, sys
```

Don't use wildcard imports:

```
from random import *
```

Naming Conventions

Variables, functions, module names, package names - use lower-case letters, plus underscores as needed to improve readability. It's okay to include numbers in names, ifn it makes sense to do so. Avoid using capital letters in these names.

```
def is_valid_phone_number(num):
```

Class Names - use CapWords (CamelCase). It's okay to use numbers in your class names, if it makes sense to do so. Avoid underscores in these names.

```
class PhoneNumber:
```

Constants - use capital letters, plus underscores as needed to improve readability. It's okay to use numbers in your constant names, if it makes sense to do so. Avoid using lower-case letters in these names.

```
UMD_COLORS = ['red', 'white', 'black', 'gold']
```

Note that many of these styling requirements are subjective and that there are legitimate circumstances under which some of these rules can/should be broken. When in doubt, add a comment. Style points will be deducted for deviations from these style requirements (up to 20% of the points for a given assignment).

Submitting Coursework

Homework assignments and take-home midterms in this course are to be submitted via upload to ELMS. Please follow the naming conventions below. Individual assignments may contain more specific instructions than the guidelines provided here. When more than one file is included in the submission, please combine the files into a single ZIP archive for submission.

Files that are corrupted, fail to open, or are empty will receive zero points. Exceptions will be handled on a case-by-case basis by the instructor.

File Naming Convention

Because we will generally want the ability to import Python scripts as modules, it's important to save these files with names that are valid module names. Please observe the following constraints when naming Python files:

Use only lower-case letters, numbers, and underscores only. Avoid hyphens, spaces, and other special characters at all costs.

- File names must begin with a letter (not a number or underscore).
- File names should be short but reasonably mnemonic.
- File names for python files should end in .py; data files and zip archives should be given the appropriate extension for the file format.
- For other kinds of files (essays, documentation, etc.), use mnemonic names.

Questions

If you have any questions about these requirements over the course of the semester, reach out to the instructor for further clarification.

Note: This document was based on an earlier version of the same, created by Aric Bills. v1.0, 2018-08-29