Coding style

This document defines how your code should be styled for assignments submitted for this subject. By following these conventions, you will produce code which is more readable and easier for you or others to follow. These guidelines apply to all programming languages but the examples shown are in Python . Being able to write clean legible code will also improve your employability in the software industry.

It also makes sense to use an IDE or text editor with plugins (I suggest VS CODE) when developing your code is it will help you to manage the indentation and commenting in a consistent manner.

Identifiers

Variables/attributes

identifies for all attributes (variables) should be meaningful and if appropriate contain its own unit of measure. The unit of measure helps in understanding how to use it without reading the comments.

Variables/attributes should always start with a lower case letter.

Examples

weightInKilos,   heightInMetres,   delayInMilliseconds;

Camel case

Always use camel case to help aid reading of identifies, so an upper-case letter is introduced for every new word. See previous examples.
The PEP style guidelines say that the style with lowercase and underscores should be used, unless the prevailing style is different: https://www.python.org/dev/peps/pep-0008/#id45

Given that you will be writing in many languages during your time at the department, I suggest that you use the prevailing style of these languages (and of the department), which is camel case. Most importantly though is just to stay consistent.

Class identifiers

Always start with an uppercase letter and should always be a noun. So Encryption is not a good name but Encryptor or EncryptionHelper is fine, again camel case is used.

Examples

Person, Doctor, Appointment, DatabaseConnector

Constants
Python does not have constants (variables that once defined cannot change). You can distinguish a "would-be constant" by using all CAPS with underscores to separate words, and simply remember that you must not change a variable that is all caps.
Examples
PI=3.149265445
MAX_ITERATIONS =4

Function names

Start with a lower case letter then follow camel case. All function names need to be verbs.

Comments
Comments must always precede the part of the code that they refer to.

The code should ideally be what is termed self-documenting, that is it is simple enough to understand without constantly referring to the comments. If the code is hard to follow then it may need to be simplified and in some cases broken up. Remember – comments start with a #

Pydoc
When used, all documentation comments should follow the pydoc format (ie: with triple quotes), this allows code documentation to be generated automatically.

Class comments
All classes you start off with a pre-amble describing the purpose of the class, what data it stores and what services it provides. So a user of the class can quickly determine how to use the said class.

Function comments
At a minimum each function in a module should be commented. This is vital since functions form the interface to the code. Each function should have comments to both the input variables and the returned values (if there are any).

Example

```
# factorial(input) – calculates the factorial of input
# Parameters: input – int
# Returns: float – the calculated factorial
def factorial(input):
	if (input<0):
		return -1.0 #invalid negative input

	ans=1
	while (input>1):
		ans=ans*input;
		input--;

	return ans
```

Indentation

All code should be indented in a consistent way in order to correctly convey the program structure, and to create a functionally correct python program.

Following these guidelines will help you to keep the code easier to read and will help you when you are debugging and maintaining you work.