|  |
| --- |
| University of Sunderland |
| Good Code Style |
| A basic set of guidelines on how to keep your code the same throughout projects, using coding styles. |

|  |
| --- |
| Heslop, Alan John  11-1-2020 |

### Indentation

Python requires an indentation for classes, loops, conditions, lists, and functions. Indentation can be either 4 spaces, or the tab key – but not a combination of both. The difference from Python 2 to 3, is that you can use an indentation with both tabs and spaces, but Python 3 will not correctly run and result in an error, there must be consistency within the indentation.

### Maximum Line-Length

An uncommented line should not exceed 79 characters; however, a comment can be no longer than 72 characters. Any line of code which may exceed the value can be wrapped using a backslash (\) which creates a line-break

### Naming Conventions

It is important to use the same style code throughout the whole application;

* Functions
  + When using a function use a lowercase word or words. To increase readability, you may use snake\_case.
* Variable
  + Variables should begin with a lowercase single letter. To increase readability, you may use snake\_case
* Classes
  + Start each class with a capital letter, use camelCase.
* Methods
  + Use a lowercase word to separate. To increase readability, you may use snake\_case
* Modules
  + Use a short, lowercase word. Separate the words with an underscore to increase readability.
* Packages
  + A package style is a full word, do not separate with an underscore.

### Comments

There are 3 types of comments available to use to document certain parts of your code for better readability and understanding.

* Block comments
  + A block comment is where the comment starts with a # and spans over different levels, all indented at the same level.
* Inline comments
  + An inline comment is where the comment is placed 2 spaces after the item, function, class on the same line as the code.
* Docstrings
  + A docstring is enclosed within double or single quotation marks, these are presented on the first line of functions, classes, methods or modules and are used for explaining what the item may represent.

### Line Spacing

Also known as blank lines, these lines can help visually split up specific items for overall readability.

* Classes
  + Leave 1 blank line between the methods in a class – this helps with readability.
* Definitions (Module and Class)
  + Leave 2 blank lines between class definitions and module definitions

### SOLID Principle

The solid principles are a standard that developers use for developing software to prevent future mistakes.

* Single Responsibility Principle
  + Using this principle gives the user the ability to create a class that should have one job, if there’s more than one item in the class then it should be split up into a different class. This helps the overall application because any future changes may affect that class which has multiple “jobs”.

### DRY (Don’t Repeat Yourself)

Code shouldn’t be repeated, if repeated this can lead to future problems with maintaining the application. Correct use of DRY code can give the application more flexibility, it becomes easier to understand, simply to test, easier to debug, and its easier to change when required.

### Extract Variables

* + Variables have the ability store a value in memory, and variables can be reused throughout the program. It is a benefit to the developer to use variables opposed to directly printing out a result.