Coding Standards

*Published by Calum Oke*

Contents

[1. Declaration 3](#_Toc493075264)

[1.1 Variables 3](#_Toc493075265)

[1.2 Constants 3](#_Toc493075266)

[1.3 Functions 3](#_Toc493075267)

[1.4 Classes 3](#_Toc493075268)

[2. Spacing 4](#_Toc493075269)

[2.1 Indentation 4](#_Toc493075270)

[2.2 Whitespace Usage 4](#_Toc493075271)

[2.3 Line Length 4](#_Toc493075272)

[2.4 Line Separation 5](#_Toc493075273)

[3. Commenting 6](#_Toc493075274)

[3.1 Inline 6](#_Toc493075275)

[3.2 Block 6](#_Toc493075276)

[4. Code Layout 7](#_Toc493075277)

[References 8](#_Toc493075278)

# Declaration

Since all programming will be done in python, following the **PEP 8 -- Style Guide for Python Code** is a recommended style guide, which this project will similarly be following.

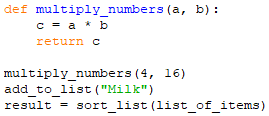
## Variables

Variables should be declared using underscore case with all characters being in lower case and have self-explanatory names. An underscore character should be placed in between the end of a word/name and beginning of a new one within the variable. Some examples are shown below:  

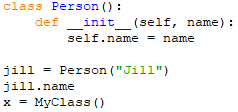

## Constants

Constant variables use the same principle of underscore case, however all characters in the variable should be upper case. The purpose of this variable type is to remove “magic numbers”, which are numbers appearing to be inputted without context. Constants should also be labelled near the top of the .py file. Examples shown below:  


## Functions

Function names will use the same character styling as variables, as a function will be identified from the parenthesis. Names should also indicate the purpose of the function without excessive characters. Examples shown below:  


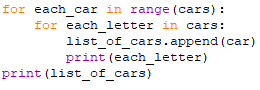
## Classes

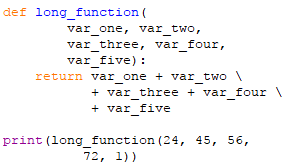
Class names should be labelled using the capital word convention, with a capital character to indicate a new word/name in the class. Examples shown below:  


# Spacing

Appropriate spacing important to ensure easy readability and reduce excessive whitespace usage.

## Indentation

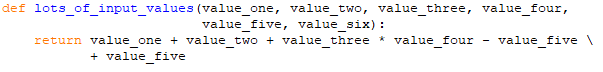
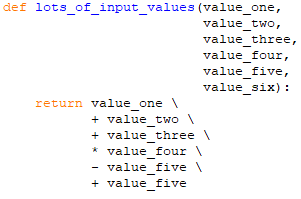
A classic rule of indentation in python is **4 space characters per indentation** **level**. An example is show below (*Note: indentation is how Python distinguishes each step)*:  
  
When using long functions with multiple arguments, they can be laid out in multiple lines, as long as the indentation is 4 spaces after the first function name character. An example shown below:



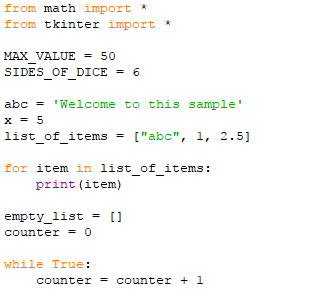
## Whitespace Usage

A single whitespace character should be used after every operation, comma or word/variable for easier readability. However, they should not be used after a left bracket/parenthesis and before the right one, as it adds excessive white-spacing. Examples are shown below:  
  
**Not:**However, exceptions of not using whitespace characters around the ‘=’ sign are applied when assigning values to parameters or in-built functions that can have inputted values. Examples below:  


## Line Length

The maximum line length should not go over 80 columns wide. Long lines of code should be broken over multiple lines by using the ‘\’ character (except in definitions) to indicate code continues over a new line. Examples below:  
  
To improve readability, it is preferred that related code is layered out by one value per line. Example below:  
  
Note that the arithmetic operator is in front of every value on the new line. This is how the related code should be implemented as well. *Example of return values may be a bit excessive in new lines, however it is to show how they should be laid out on each line*.

## Line Separation

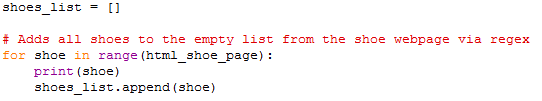
Lines of white space should be implemented between each unrelated block of code or blocks that have different functionality. Examples below:  
  
Imports, constants, variables etc. are separated by a newline to indicate new functionality

# Commenting

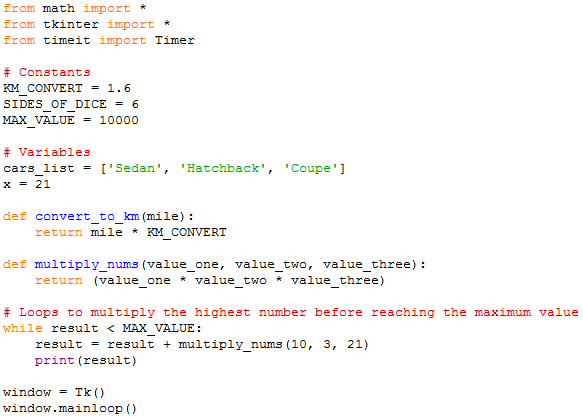
## Inline

Inline commenting should be try to be avoided, unless code is not interoperable by a person or requires explanation of purpose. For example:  
  
Usually using self-explanatory variable names or block comments eliminates the usage of inline comments

## Block

Block commenting should always be used when complex blocks of code occur that need functionality explaining. These comments should always be placed above each block of code and under each line separation. Examples shown below:  


# Code Layout

A general Python document should be laid out in the same concept as below:  
  
This example displays briefly how the code should be laid out. Imports should always be placed at the very top of a program. Constants and global variables should be separated and placed near the top. Function definitions should be placed under variable declarations and above main program functionality. Appropriate commenting structure is also shown above.

# References

van Rossum, G., Warsaw, B., & Coghlan, N. (2001, July 5). *PEP 8 -- Style Guide for Python Code*. Retrieved from python.org: https://www.python.org/dev/peps/pep-0008/#programming-recommendations