**Coding Standards (Python)**  
**v1.0**  
**October 14, 2024**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Version Description

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 10/06/24 | RM and LA | Document Created |
| V1.0 | 10/14/24 | RM and LA | Python Documentation Included |
| v2.0 |  |  |  |
| v3.0 |  |  |  |

# Contents of this Document

* General
* Naming Conventions
* Commenting Code
* Parenthesis
* Constants
* Line Spacing
* Braces
* Declarations

# General

Our project’s functionality will include many services which will be needed for several Classes to ensure efficient implementation. Our scope will include the use of Python, an Object-Oriented Language. Coding standards will follow normal Python syntax and will make use of appropriate whitespace for enhanced readability.

Goals for code guidelines:  
1. All source code should be easily readable and understandable.

2. The resulting documentation should be easy to maintain.

3. Code changes should be relatively easy with no need for complete code refactoring.

# Naming Conventions

**Import Names:** Import names should be single lowercase words.  
  
**Class Names:** Concrete classes should use natural descriptive names, begin with a capital letter with capitalization of the first letter of every subsequent word. For example: UserAuthentication  
  
**Member Function Names:** Method ('member function') names should begin with a lowercase letter with each subsequent new word in uppercase, and subsequent letters in each word in lowercase. This is commonly known as CamelCase. All member functions should have “self” as the first parameter. If it is not a member function, then “self” will not be required.  
**Example:**  
class ExampleClass:  
 def doSomethingNeat(self, aValue):  
 def debugDumpToScreen(self):  
 def someClassMethod(self, aValue):

# Commenting Code

Classes: Class attributes and member methods should include comments specifying types of variables and the purpose of functions. Using docstrings (three double quotes), give a description of the function, list the parameter names along with their type and a brief description of its purpose or why it is being included. Brief descriptions cannot be longer than 2 sentences. Types should be lowercased. For example, int and str are correct, but not INT and STR.

Main: Same rules apply to main methods not in a class and variables outside of main.

NOTE: For methods that don’t return any value, don’t include a “Return” in the method comment. Single-line comments should use sharp (#), followed by a single space. Multi-line should use triple quotes (“““”””).

**Example:**

def \_\_init\_\_(self, name, age):

“““

Constructs all the necessary attributes for the animal object.

Parameters

name: str

The name of the animal.

age: str

The age of the animal.

”””

self.name = name

self.age = age

def myFunction(self) -> str:

“““

Gives a string representation of the Object.

Returns

str – A string including the current Object’s name.

return self.name

# Parenthesis

Parenthesis should always be used to make use of conditionals more obvious if necessary.

**Example:**  
if x + y:

NOTE: This example is really easy to interpret, so no parenthesis is needed.

if ((x\*\*2 \* (variable // value)) % (17 - variableTwo):

NOTE: This example is more complex, which is why we need the use of parenthesis to make the order of statements more obvious.

# Constants

Constants should be all uppercase with a comment #CONSTANT VALUE indicating the variable will be used as a constant.

**Example:**

#CONSTANT VALUE

DAYS = 7

# Line Spacing

Use blank lines to separate different logical blocks of code, such as function definitions, loops, conditional statements, and imports. For tabs, normal tabbing for loop, conditionals, functions and classes is required, i.e. use the default tabsize of 4. Blank lines should also be used between logical blocks of code. This will ensure good readability. Place a blank line before function definitions, a blank line before and after class definitions, and use blank lines to separate logical sections within a function, like between variable declarations and processing logic, or before returning a result.

**Example:**

import math

class Circle:

def \_\_init\_\_(self, radius):

self.radius = max(0, radius)

def area(self):

return math.pi \* self.radius \*\* 2

# Declarations

Per Python syntax, all variables must be initialized upon creation. Any indeterminate variable should receive a default value and a comment stating the reason for the value.

**Example:**

# float(‘inf’) is a default value to be used for the min variable since it is the largest value to start

minimum = float(‘inf”)