Pittsburgh MTA Train System

Coding Standards

Procured by Asher Goodwin, Ralph Gonsalves, Aidan Gresko, Julia Koma, Beryl Sin, Reed Yulis

Version 1.0

September 27, 2023

Table of Contents

1. Introduction

Python is a widely used language in a variety of applications; therefore, conventions pertaining to this language change throughout years of development. For the specific use of our application, we are using the widely applied and recommended PEP 8 programming style.

This coding standards document will cover styling associated with PEP 8.

2. Naming Conventions

To maintain the readability of names in code, names will never be a single character. Numbers will also be excluded from all types of names used in code to avoid the confusion of the English alphabet and numbers that may look similar in some fonts; such as, the letter 'O' and the number '0'. All names should be self-explanatory and clear to other collaborators and viewers of the code.

2.1 Class

Class names will follow the CapWords convention (i.e., ExampleClass). Class names can at most be 2 conjoined words.

2.2 Properties or Attributes

Both properties and attributes will be in lowercase. Note, there will be no prefixes or underscores used before properties and attributes. Therefore, to avoid conflict choose a different name. Properties and attributes will be in mixed case, just like variables, if more than one word is needed.

2.3 Methods

Methods will use lowercase words separated by underscores (i.e., example_variable). This is to improve readability.

2.4 Variables

Variables will follow mixed case naming conventions (i.e., exampleVariable). In the case of temporary loop variables, the use of single letter variables is permitted (i.e., 'i', 'j', 'k'). This is the only exception to using single characters for naming.

2.5 Constants

Constants will be identifiable in all capital letters (i.e., EXAMPLE_CONSTANT). This is for easy differentiation between constant variables and other variables. If constants are more than one word, underscores will be used in between each word.

2.6 Comments and Comment Style

Comments will be used to provide further insight, for other collaborators, into the code that follows. They should not be utilized to state obvious information, as this is covered by naming conventions. Keeping comments up to date with the code is important.

Comments are to be written in complete sentences using proper English with the first letter in the sentence being capitalized and the last character being a period. They will always be written in python syntax using a #.

For classes and methods, docstring comments will be used. For classes, the docstring comment will provide details about the class, as well as properties or attributes of the class and their variable types. For methods, one-line docstrings can be used for obvious cases with what the method returns. Otherwise, the docstring comment for methods will span multiple lines and detail the method, and provide descriptions of the parameters that are used, the type of variable they are, as well as their default values. See below for examples:

- **Class Docstring:**

```
class ExampleClass:
  “””
  This is an example of a class docstring
  Attributes:
    example (str): description of example
    exampleTwo (int): description of exampleTwo
    exampleThree (bool): description of exampleThree
   “””
```

- **Method One-line Docstring:**

```
def example_one_line_method
    “””
    return something from this method
    “””
```

- **Method Multi-line Docstring:**

```
def example_one_line_method
    “””
    This is an example of a method docstring
    Attributes:
        example (double): description of example (default 0.0)
        exampleVariable (str): description of exampleVariable (default
    “””
```

3. Specific Requirements

3.1 Indentation and Brackets

Indentation will be equal to that of four spaces. Spaces are preferred over tabs, but tab indentation can be set to four spaces. Spaces and tab indentation are to not be mixed in code. When in the same scope level (i.e., in a method), indentation will be equal to indicate that the code is grouped together. Brackets are not utilized in python. This is important to note as indentation will determine the scope of code.

3.2 Spacing

Avoid extraneous whitespace in code. Single spaces should be used after commas. Infix operators must have single spaces surrounding them. Lastly, keywords should also have a space directly after them (i.e., if, for, while, etc.).

- **Infix Operators:**

```
#Do                                  #Don't

exampleStr = 'hello' + 'world'       exampleStr = 'hello'+'world'

exampleInt = 1 + 2 + 3               exampleInt= 1+2+3
```

3.3 Exception Handling

When handling exceptions, the default ‘try’ and ‘except’ method can be used. This is the most common and basic way for handling errors in python; thus, we will implement exception handling this way. Multiple exceptions can be handled using branching if needed. Raising user-defined exceptions using ‘raise’ is also prohibited. Lastly, clean-up actions can be defined through ‘else’ and ‘finally’ branching (See below).

```
try:
    raise Error
except FirstTypeOfError:
    print("First type of error!")
except SecondTypeOfError:
    print("Second type of error!")
else:
    print("result")
finally:
    print("in finally")
```

4. References

As noted in the naming convention, most of the styling followed comes from the PEP 8 styling guide (https://peps.python.org/pep-0008/). Changes are implemented as seen fit for each naming convention or style.