Coding Standards and Guidelines

Created by Renwei Hu, last modified on 10 Sep, 2023

Importance

> Code is read much more often than it is written. The guidelines are intended to improve the **readability** of code and make it **consistent** across the whole project.

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves the readability and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect errors easily.
- It promotes sound programming practices and increases the efficiency of the programmers.

Languages Used

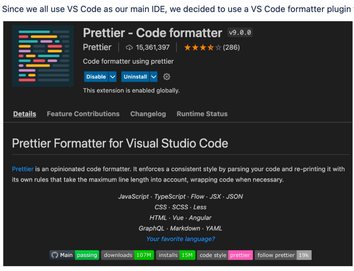
Python

- ROS 2
- YOLO
- API Scripts

Formatter Tools

1. Prettier (VS Code Plugin)

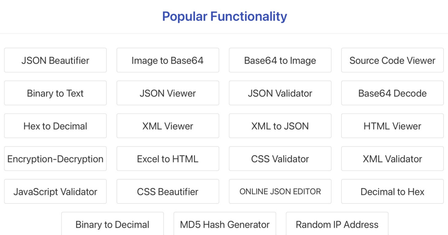
Since we all use VS Code as our main IDE, we decided to use a VS Code formatter plugin to minimise the work for a consistent coding style across the project.



2. Code Beautify (Online Tool)

Free Online Tools For Developers - codebeautify.org

Code Beautify is an online code formatting tool which supports various programming languages.



Python Coding Standards

> Unless otherwise specified, follow PEP8

Naming Conventions

Component	Convention	Example
Variable	Use lowercase with words separated by underscores	`my_variable`
Function	Use lowercase with words separated by underscores	`my_function`
Class	CapWords convention	`MyClass`
Method	Same as function naming rules	`class_method`
Constant	All capital letters separated by underscores	`MY_CONSTANT`
Module	Short, all-lowercase names	`my_module.py`
Package	Same as module naming rules, but better discard underscores	`mypackage`

Code Layout

Component	Convention
Indentation	Use 4 spaces per indentation level
Tabs / Spaces	Prefer spaces, don't mix tabs with spaces
Maximum line length	Limit all lines to a maximum of **79** characters
Line break	Prefer to break before binary operators
Blank lines	Surround top-level function and class definitions with two blank lines
Imports	Imports should usually be on separate lines

String Quotes

- Choose either single or double quotes and ensure consistency.
- When a string contains single or double quote characters, however, use the other one to avoid backslashes in the string.

```
# Escape the quotes in a string
s_sq = "ab\"c"
print(s_sq)
# ab"c

# Just use single quotes (preferred)
# No need to insert escape characters
s_sq = 'ab"c'
print(s_sq)
# ab"c
```

White Spaces in Expressions

- Avoid extraneous whitespace when:
  - In parentheses, brackets or braces.
  - Between a trailing comma and a following close parenthesis
  - Immediately before a comma, semicolon, or colon

```
# Correct:
spam(ham[1], {eggs: 2})
# Wrong:
spam( ham[ 1 ], { eggs: 2 } )

# Correct:
foo = (0,)
# Wrong:
bar = (0, )

# Correct:
if x == 4: print(x, y); x, y = y, x
# Wrong:
if x == 4 : print(x , y) ; x , y = y , x
```

- Always surround these binary operators with a single space on either side:
  - Assignment, Augmented assignment
  - Comparison
  - Boolean

```
# Correct:
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

- Don't use spaces around the = sign when used to indicate a keyword argument

```
# Correct:
def complex(real, imag=0.0):
    return magic(r=real, i=imag)

# Wrong:
def complex(real, imag = 0.0):
    return magic(r = real, i = imag)
```

Comments

Component	Convention
Line comments	Use complete sentences, starting with a capital letter
In-line comments	On the same line as the statement they refer to, separated by 2+ spaces from the statement
Block comments	Indented to the same level as that code. Each line of a block comment starts with a **#** and a single space.
Documentation strings	Write docstrings for all public modules, functions, classes, and methods. Start and end multiple lines of comments with triple #

```
# Normal line comments

x = x + 1                 # In-line comments

# This is block comment
# Made using # character
# Used multiple times
print ("Block Comment")

"""Return a foobang
Optional plotz says to frobnicate the bizbaz first.
This is documentation strings
"""
```