Python コーディング規約

目次

１．概要

本書は、ミールシールドで製作する Python プログラムのコーディングに対して規則を定める。この規則にそってプログラムをコーディングすることにより保守性の向上や統一した品質を保持することを目的とする。

２．適用

ミールシールドで制作する Python プログラムに適用する。

３．コーディング規約の設定基準

(1) Python で生じやすいコーディングエラーを減らすために利用可能なコーディングスタイルを限定する。

(2) レビュー及び保守を行う上で、次の基準を満たすこと

- 十分に読みやすいこと
- 技巧に走った難解なプログラミングでないこと
- 必要なコメント情報が記述されていること

(3)レビュー時に、チェックできる記述であること

４．Python コーディング規約

本章では、Python のプログラムを作成する上で守るべき規約について説明する。規約については下線を引いている。

４．１　全般

(1)　　関連のある関数やクラスを 1 ファイルにまとめる。

(2)　　複数のモジュールに共通の項目（定数や設定）は、専用の設定ファイルまたはモジュールにまとめて記述する。

(3)　　原則として、一行一文とする。

(4)　　一行は原則 79 文字とし、100 文字以内とする。

(5)　　共用体（C 言語のユニオンに相当するもの）の使用は避ける。

４．２　Python のシンタックス

４．２．１　演算子

(1) 多様な演算子と優先順位に気を付けること。

(2) 優先順位が分かりにくい場合は括弧でくくること。

(3) 式が 2 行以上にまたがる場合、優先順位のもっとも低い二項演算子の直前で改行すること。

(4) 関数呼び出しが 2 行以上にまたがる場合は、カンマの直後で改行すること。

４．２．２　関数定義

(1) 戻り値を持たない関数は None を返すこと。

(2) 関数定義の行は、一行目から書き始めること。

(3) 関数名の後に括弧()を続け、その中に仮引数を記述すること。仮引数ごとに改行を入れ、仮引数の後ろにコメントを記述すること。

(4) 関数に対応する:の後には何も書かないこと。

(5) 関数の中の処理は簡潔にすること。100 ステップ以内で収まることが望ましい。

(6) 各処理で共通に使われるデータは引数渡しにより受け渡しすること。

(7) 関数名や変数名など識別子は英字で始まる英数字列にする。

４．２．３　クラス定義

クラス定義は、一般的なクラス定義の形式に従う。

４．３　コードレイアウト

ソースファイルは通常以下の定義部を含む。

- インポート文
- クラス定義
- 関数定義

これらはソースファイル上では次のような構成にすること。

〈例〉

```
# インポート文
import os
import sys

# クラス定義
class Example:
    pass

# 関数定義
def main() -> None:
    pass
```

４．４　データの定義

(1) 使用するデータ型に気を付けること。

(2) 特定のハードウェアに依存しないようにプログラミングに気を付けること。

(3) 変数の初期化は必ず行うこと。

４．５　命名

(1) グローバルな関数名、ファイル名、変数名の先頭には必ず機能 ID をつけること。

(2) 関数名、ファイル名、変数名の単語と単語の区切りにはアンダースコアを使用すること。

(3) 関数名、ファイル名、変数名は、全て小文字で形成すること。

４．６　実行文の書き方

(1) 一行に二つ以上の文を書かないこと。

(2) 条件文や繰り返し文が二行以上にまたがる場合は、単文でも:の後にインデントを入れること。

(3) 余分な else は書かないこと。

(4) switch 文（Python では match 文）の default 部は最後に記述すること。

(5) return 文は原則として各関数に 1 つとすること。

〈例〉

```
if condition:
    # statement
    pass
else:
    # statement
    pass

match value:
    case 1:
        # statement
    case 2:
        # statement
    case _:
        # default statement
```

４．７　コメントの書き方

(1) コメントはまとまった処理の主要箇所に挿入すること。

(2) 二行以上のコメントの場合、途中の行の先頭に#を書くこと。

(3) 変数名や型などのソースコードに強く依存した情報はコメントに含めないこと。

(4) 関数定義の直前に関数の表題コメントを入れる。

(5) 理解を助けるため、適宜日本語を使用する。

〈例〉

```
def example_function(param1: int, param2: str) -> None:
    """
    Function Name: example_function
    Designer: 芝浦 太郎
    Date: 1996.7.12
    Function: シミュレーションデータの計算を行う。また、異常データの検出も行う。
    Return: None
    """
    # main process
    pass
```

４．８　移植性の高いプログラムの書き方

(1) 特定の環境に依存しないコードを書くこと。

(2) 標準ライブラリを優先して使用すること。

４．９　ソースコードの修正

(1) バージョン管理システム Git を使用すること。

(2) バージョン管理システム GitHub にて PR で承認を得ること。

４．１０　ファイルの構成

(1) 関係が強い変数や関数は、一つのファイルにまとめること。

(2) Python ファイルの構成

- 処理コードファイル（.py ファイル）
- インクルードファイル（設定や定数を定義するファイル）
- データファイル（プログラム実行中に読み込むパラメータなど）
- 説明用ファイル（プログラムの各機能／構成を説明するテキストファイル）