

Standard de Codage Python

CLS-ENV-NT-24-0397

Issue 1.0 – 20/06/2024

CHRONOLOGY ISSUES/HISTORIQUE DES VERSIONS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Issue/ Version | Date | Object/Objet | Written by/ Rédigé par | Checked by / Vérifié par | Approved by/ Approuvé par |
| 1.0 | 20/06/24 | Version initiale |  |  |  |
|  |  |  |  |  |  |

DISTRIBUTION/LISTE DE DIFFUSION

|  |  |  |
| --- | --- | --- |
| Company/Organisme | Means of distribution/ Format de diffusion | Names/Destinataires |
| CLS | Electronique | Equipe projet |
| CNES | Electronique | Equipe projet |

LIST OF CONTENTS/SOMMAIRE

[1 Introduction 4](#_Toc52547366)

[2 Exigences générales 4](#_Toc52547367)

[3 Présentation 5](#_Toc52547368)

[4 Règles de codage 6](#_Toc52547369)

[5 Bonnes pratiques 7](#_Toc52547370)

LIST OF TABLES AND FIGURES/LISTE DES TABLEAUX ET FIGURES

N/A

LIST OF ITEMS TO BE CONFIRMED OR DEFINED/LISTE DES AC ET AD

N/A

APPLICABLE DOCUMENTS/LISTE DES DOCUMENTS APPLICABLES

N/A

REFERENCE DOCUMENTS/LISTE DES DOCUMENTS DE REFERENCE

N/A

# Introduction

Ce document présente le standard applicable pour les projets ou composants développés en Python à CLS.

# Exigences générales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-GEN-0001/V | | None | Obligatoire | | |
| Exigence | Le projet/composant doit définir au plus tôt (conception générale) une politique unique de gestion des erreurs et des exceptions | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue de conception |
| Notes: Ne pas confondre gestion des erreurs (comportement prévus) et gestion des exceptions (comportement imprévus) | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-GEN-0002/V | | None | Recommandé | | |
| Exigence | Le langage Python permettant la programmation orientée objet, le projet/composant doit en respecter les principes fondateurs : encapsulation, héritage et polymorphisme | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue de conception |
| Notes: En particulier, la production et la consommation des variable est à vérifier. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-GEN-0003/V | | None | Recommandé | | |
| Exigence | Le projet/composant doit définir des interfaces claires entre les différents types d’objets : objets visuels, objets métier/applicatifs, système de persistance. | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue de conception |
| Notes: L’intérêt de séparer les objets en « couches applicatives » est de pouvoir tester plus facilement les différentes couches, mais aussi de pouvoir en remplacer une sans impacter le reste du code. | | | | | |

# Présentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-PRES-0010/V | | None | Obligatoire | | |
| Exigence | L’arborescence-type des projets Python doit respecter la structure suivante :  Dossier/  │  ├── Docs/  │ ├── hello.md  │ └── world.md  │  ├── Fonctions/  │ ├── fonction1.py  │ └── fonction2.py  │  ├── Classes  │ ├── sat1.py  │ ├── sat2.py  │ ├── sat3.py │  │ └── sat4.py  │  ├── Main1.py  ├── Main2.py  ├── Main3.py  │  └── README.md | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue de conception |
| Notes: Documentation de référence : <https://realpython.com/python-application-layouts/> | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-PRES-0011/V | | None | Obligatoire | | |
| Exigence | Le code Python doit respecter le formatage, l’indentation, et la convention de nommage du PEP 8 | | | | |
| Critère de vérification | Configuration des postes de travail | | | Jalon associé | Revue en continu |
| Notes: Documentation de référence : <https://www.python.org/dev/peps/pep-0008/> | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-PRES-0012/V | | None | Obligatoire | | |
| Exigence | Une ligne de code ne doit contenir qu’une seule instruction | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue en continu |
| Notes: Améliore la lisibilité du code | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-PRES-0013/V | | None | Recommandé | | |
| Exigence | Les noms d’éléments Python (modules, classes, fonctions, attributs, variables, …) doivent être explicites | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue en continu |
| Notes: | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-PRES-0014/V | | None | Recommandé | | |
| Exigence | Les entêtes de éléments Python doivent être les mêmes pour l’ensemble du projet/composant | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue en continu |
| Notes: Il n’existe pas de standard d’entête Python à CLS. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STD-PYTH-PRES-0015/V | | None | Obligatoire | | |
| Exigence | Le code doit être écrit en anglais et les commentaires en anglais | | | | |
| Critère de vérification | Revue par les pairs | | | Jalon associé | Revue en continu |

# Bonnes pratiques

Les bonnes pratiques de codage Python sont rassemblées dans le tableau ci-dessous :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thème | Bonne pratique | | Intérêt | |
| Tests unitaires | Ecrire ou modifier les tests unitaires avant le code | | Permet de produire un code plus sobre et plus efficace. | |
| Internationalisation | | Pour tout libellé à traduire, utiliser *gettext* | | Permet de centraliser les traductions |
| Complexité | Se limiter à 15 lignes de code par méthode. Au-delà, créer une sous-méthode. | | Augmente la lisibilité du code | |
| Règles de codage | Utiliser le plugin SonarLint (disponible pour Eclipse, IntelliJ, Visual Studio, VS Code) | | Permet de corriger les défauts dès la rédaction du code | |