# Langage Python

*Référentiel décembre 2023*

Editions ENI

## Référentiel de compétences

| **Description du métier, de l'activité ou de la situation professionnelle à partir duquel le dispositif visant la certification est initié :** |
| --- |
| La certification IT **Langage Python** s'adresse aux développeurs et autres professionnels de l'informatique (administrateurs système et réseau, ingénieurs DevOps, administrateurs cloud) qui souhaitent ajouter le langage Python à leurs compétences, ainsi qu'aux professionnels du traitement de la donnée qui ont besoin de compétences en langage Python pour étendre leurs capacités professionnelles (analystes de données, statisticiens, data scientistes, scientifiques...)<br><br>Les candidats valident la certification s'ils obtiennent un score minimal de 700 points sur 1000. |

# Langage Python

*Référentiel décembre 2023*

<table>
<tr><th colspan="4">Référentiels</th></tr>
<tr><th rowspan="2">Référentiel de compétences</th><th colspan="3">Référentiel de certification</th></tr>
<tr><th colspan="2">Modalité(s)</th><th>Critères</th></tr>
<tr><td rowspan="3">Ecrire un script à l'aide des fonctionnalités essentielles du langage Python (variables, fonctions…) pour créer une application qui s'exécute en mode ligne de commande.<br><br><u>Applications métiers :</u><br><br>- Administrateur système : Ecrire des routines permettant de manipuler des données issues d'un système de surveillance du réseau.<br><br>- Statisticien : Manipuler des données structurées afin de générer des graphiques en utilisant une librairie telle que Matplotlib pour générer des courbes représentant l'évolution de la population.</td><td>C1 - Manipuler correctement les variables en respectant les types de données et en réalisant des conversions si nécessaire pour identifier les données d'un programme.</td><td rowspan="3"><u>Evaluation théorique</u> :<br><br>Le candidat doit répondre à un questionnaire de type QCM/QCU à visée professionnelle.<br><br><u>Evaluation pratique</u> :<br><br>Le candidat doit écrire une partie de code Python en intégralité selon un problème donné, ou bien doit compléter une structure de code existante en ajoutant les instructions manquantes.</td><td>Cr 1 Les types de données sont respéctés<br>Cr 2 Des conversions sont faites si nécessaire en utilisant les fonctions str(), int() et float().</td></tr>
<tr><td>C2 - Manipuler des structures de données en utilisant les types composés (listes, tuples, dictionnaires) et leurs méthodes appropriées pour traiter des lots de données.</td><td>Cr 1 Une sous-liste est extraite d'une liste existante selon des critères de sélection à l'aide de la boucle for, de la structure if, de la fonction append() et de l'opérateur in.<br>Cr 2 Des données sont extraites d'une structure de données hiérarchique à l'aide de dictionnaires, des fonctions items(), keys() et values() et de la structure for.</td></tr>
<tr><td>C3 - Mettre en œuvre les structures de contrôle conditionnelles et itératives en écrivant des algorithmes complexes pour traiter des ensembles de données.</td><td>Cr 1 Les instructions (if, elif, else) et opérateurs (and, or, not) conditionnels sont correctement utilisés et combinés pour tester diverses conditions.<br>Cr 2 Les instructions for et while sont correctement utilisées pour répéter des actions sur des ensembles de données, en maîtrisant l'utilisation des contrôles de boucles comme break et continue pour modifier le flux d'exécution au sein des boucles.<br>Cr 3 Des algorithmes efficaces et performants sont élaborés et implémentés pour traiter un ensemble de données.<br>Cr 4 Les erreurs courantes liées aux ensembles de données (valeurs manquantes, types inattendus…) sont gérées de manière appropriée.</td></tr>
</table>

| | | | |
|---|---|---|---|
| | **C4** - Utiliser les entrées et sorties standards en exploitant les fonctions et concepts associés du langage (f-string, input(), format()) pour lire et écrire des données utilisateurs. | | **Cr 1** La fonction print() est utilisée pour écrire des données correctement formatées à l'écran.<br>**Cr 2** La fonction format() des chaînes de caractères est utilisée.<br>**Cr 3** Les f-string sont utilisés.<br>**Cr 4** La fonction input est utilisée pour lire des saisies clavier.<br>**Cr 5** La structure while est utilisée. |
| | **C5** - Utiliser les fonctions de base du langage en réalisant des opérations élémentaires pour dénombrer des éléments, générer des nombres… | | **Cr 1** La fonction len() est correctement utilisée pour dénombrer les éléments d'une liste ou d'une chaîne.<br>**Cr 2** Les listes, les tuples et les dictionnaires sont correctement utilisés pour stocker et dénombrer des éléments.<br>**Cr 3** La fonction range() est utilisée pour générer des séquences de nombres.<br>**Cr 4** Les fonctions de la bibliothèque random sont utilisées pour générer des nombres aléatoires. |
| Répartir le code des fonctionnalités d'une application écrite en langage Python avec des fonctions organisées en modules et packages pour permettre leur réutilisation future dans d'autres applications.<br><br><u>Applications métiers</u> :<br><br>- Data Scientist : Utiliser des fonctions de traitement de données avec la librairie Scikit-Learn dans des applications utilisant l'apprentissage automatique.<br><br>- Développeur Web : Créer une librairie de consommation d'une API pour appeler des services distants permettant d'obtenir un comparatif de prix sur un site marchant. | **C6** - Invoquer une fonction après l'avoir déclarée en spécifiant correctement les paramètres pour factoriser des instructions de code. | | **Cr 1** Le mot clé def est utilisé.<br>**Cr 2** Les paramètres sont correctement déclarés et passés au moment de l'invocation de la fonction.<br>**Cr 3** L'indentation du code est respectée dans le corps de la fonction.<br>**Cr 4** La valeur de retour est correctement retournée et exploitée. |
| | **C7** - Créer un module dans une arborescence de package en utilisant plusieurs fonctions utilitaires pour organiser les fonctionnalités de code. | | **Cr 1** Le module est stocké dans une arborescence de package.<br>**Cr 2** Le fichier du module est correctement nommé.<br>**Cr 3** Les fonctions sont correctement déclarées dans le module. |
| | **C8** - Invoquer directement les fonctionnalités d'un module, telles que des fonctions et des classes, depuis un autre module en utilisant les mots clés adéquats pour utiliser des structures de code avancées. | | **Cr 1** Les mots clés import et from sont utilisés.<br>**Cr 2** Le mot clé as est utilisé pour lever un conflit sur des noms similaires de fonctions. |
| | | | |

# Langage Python

*Référentiel décembre 2023*

| | | |
|---|---|---|
| Ecrire des fonctionnalités métiers d'une application développée en langage Python en utilisant les concepts de programmation orientée objet (classes, attributs, objets, méthodes...) pour permettre une meilleure modularité de l'application.<br><br>Applications métiers :<br><br>- Développeur : Concevoir et implémenter un progiciel de gestion RH ou une application de gestion de la prise de commandes sur la base d'une analyse fonctionnelle métier.<br><br>- Scientifique : Représenter un ensemble de molécules chimiques et leurs relations. | **C9** - Déclarer une classe avec des attributs en prévoyant la structure de code pour les initialiser afin de pouvoir créer des objets. | **Cr 1** Le mot clé class est utilisé.<br>**Cr 2** L'indentation du code est respectée.<br>**Cr 3** Le constructeur est correctement déclaré dans la méthode __init__().<br>**Cr 4** Les attributs sont initialisés dans le constructeur. |
| | **C10** - Déclarer des méthodes dans une classe à l'aide des mots clés appropriés et des paramètres nécessaires pour doter les objets de comportements spécifiques. | **Cr 1** Le mot clé def est utilisé.<br>**Cr 2** Le premier paramètre déclaré est nommé self.<br>**Cr 3** Les attributs sont accédés au travers de l'objet courant self. |
| | **C11** - Instancier une classe en initialisant ses attributs pour créer des objets. | **Cr 1** Le nom de la classe est utilisé comme fonction de création d'objet.<br>**Cr 2** La variable référençant l'objet est utilisée pour accéder aux membres. |
| | **C12** - Encapsuler les attributs d'une classe en les rendant accessibles avec des propriétés pour garantir l'intégrité des objets. | **Cr 1** Le nom des attributs est préfixé d'un double underscore (__).<br>**Cr 2** Des propriétés sont déclarées pour accéder aux attributs.<br>**Cr 3** Les décorateurs de déclaration des propriétés sont utilisés. |
| | **C13** - Déclarer des membres de classes en utilisant le décorateur @classmethod pour implémenter des fonctionnalités transverses. | **Cr 1** Le décorateur @classmethod est utilisé sur les méthodes de classes.<br>**Cr 2** Le premier paramètre des méthodes de classe est nommé cls.<br>**Cr 3** Les attributs de classe sont déclarés explicitement.<br>**Cr 4** L'accès aux membres de classes est réalisé au travers du nom de la classe. |
| | **C14** - Mettre en œuvre l'héritage en déclarant une ou plusieurs sous-classes pour créer une hiérarchie de classes et permettre la réutilisation d'objets dans une application. | **Cr 1** La déclaration d'une sous-classe est faite correctement en indiquant la super-classe entre parenthèses derrière le nom de la sous-classe.<br>**Cr 2** Le constructeur de la sous-classe invoque celui de la super-classe.<br>**Cr 3** Le mot clé super est utilisé.<br>**Cr 4** L'objet créé à partir de la sous-classe invoque correctement une méthode héritée.<br>**Cr 5** L'objet créé à partir de la sous-classe invoque correctement une méthode déclarée dans la sous-classe. |

| | | | |
|---|---|---|---|
| | **C15** - Redéfinir des méthodes dans des sous-classes en spécialisant leurs comportements pour les adapter aux besoins des sous-classes. | | **Cr 1** La signature de la méthode redéfinie est respectée.<br>**Cr 2** La méthode de la super-classe est invoquée dans la méthode redéfinie. |
| | **C16** - Mettre en œuvre le polymorphisme en invoquant une même méthode sur des objets distincts pour exploiter les comportements spécifiques de ces objets. | | **Cr 1** La signature de la méthode héritée est respectée.<br>**Cr 2** Le type d'un objet est testé avec la fonction isinstance(). |
| Implémenter la gestion des exceptions en utilisant les mécanismes appropriés du langage Python pour améliorer la robustesse du programme.<br><br><u>Applications métiers :</u><br><br>- Développeur Web : Gérer les erreurs sur un site marchand afin de rediriger l'utilisateur vers des pages d'erreur en cas de problème.<br><br>- Administrateur système : Créer un système d'alerte en cas de défaillance d'un ou plusieurs services serveur dans une infrastructure informatique. | **C17** - Déclencher une exception en identifiant des préconditions non respectées à l'exécution d'une méthode ou d'une fonction pour notifier d'une erreur applicative. | | **Cr 1** Des tests sur les paramètres sont effectués.<br>**Cr 2** Un objet d'exception est créé.<br>**Cr 3** Le mot clé raise est utilisé. |
| | **C18** - Mettre en œuvre un gestionnaire d'exception à l'aide des mots-clés try et except afin d'effectuer le traitement d'une erreur applicative. | | **Cr 1** Les blocs try et except sont utilisés.<br>**Cr 2** Le type de l'exception est correctement identifié dans le bloc except. |
| | **C19** - Déterminer l'ordre de mise en œuvre de plusieurs gestionnaires d'exceptions en utilisant des blocs except multiples pour permettre de gérer plusieurs exceptions dans l'ordre approprié au sein d'une structure de code. | | **Cr 1** Le bloc try est utilisé.<br>**Cr 2** Plusieurs blocs except sont utilisés.<br>**Cr 3** Les types des exceptions sont correctement identifiés dans les blocs except.<br>**Cr 4** La hiérarchie des exceptions est respectée dans l'ordonnancement des blocs except. |
| | **C20** - Gérer les erreurs d'une application en créant des classes d'exceptions personnalisées pour améliorer la compréhension des dysfonctionnements d'une application. | | **Cr 1** Une classe héritant de Exception est déclarée.<br>**Cr 2** Un objet de cette classe est créé et correctement géré.<br>**Cr 3** Les blocs try et except sont utilisés sur ce type d'exception. |

| | | | |
|---|---|---|---|
| Exploiter les techniques de gestion de fichiers et d'interaction avec le système d'exploitation en utilisant la bibliothèque standard de Python pour écrire des programmes et des scripts utilitaires multiplateformes.<br><br>Applications métiers :<br><br>- Data Analyst : Extraire des données statistiques d'un fichier texte et les écrire dans une base de données relationnelle après traitement de ces données.<br><br>- Administrateur système : Lire des données de configuration de machines depuis un fichier CSV et appliquer ces modifications sur les machines en exécutant les processus nécessaires. | **C21** - Mettre en œuvre les modules sys et os de la bibliothèque standard en utilisant leurs fonctions basiques pour obtenir des informations sur le système d'exploitation. | | **Cr 1** Les fonctions basiques du module os telles que getcwd(), getlogin() sont utilisées.<br>**Cr 2** La variable environ est manipulée pour obtenir les variables d'environnement du système. |
| | **C22** - Manipuler le système de fichiers en utilisant le module pathlib de la bibliothèque standard pour lire et écrire des données dans des fichiers. | | **Cr 1** La classe Path est instanciée pour créer des objets représentant des chemins sur un système de fichiers.<br>**Cr 2** Les méthodes de la classe Path telles que absolute(), exists(), is_dir(), is_file() sont utilisées pour naviguer dans une arborescence.<br>**Cr 3** Les méthodes de la classe Path telles que open(), readlines(), writelines() sont utilisées pour créer, lire, écrire et supprimer des fichiers au format texte. |
| | **C23** - Exécuter des commandes sur le système en mettant en œuvre les fonctions du module subprocess de la bibliothèque standard pour faciliter l'intégration de Python avec le système d'exploitation. | | **Cr 1** La fonction run() est utilisée avec ses différents arguments pour lancer une commande système et collecter sa sortie standard dans un fichier. |