Python, programmation objet

-Réference: **SII-299** -Durée: **5 Jours (35 Heures)**

Les objectifs de la formation

- Maîtriser la syntaxe du langage Python
- Acquérir les notions essentielles de la programmation objet
- Connaître et mettre en oeuvre les différents modules Python
- Mettre en oeuvre les outils de test et d'évaluation de la qualité d'un programme Python
- Concevoir des interfaces graphiques

A qui s'adesse cette formation ?

POUR QUI :

- Développeurs, ingénieurs, chefs de projets proches du développement.

PRÉREQUIS :

- Disposer de connaissances de base en programmation (idéalement en langage objet).

Programme

- **Syntaxe du langage Python**
  - Les identifiants et les références.
  - Les conventions de codage et les règles de nommage.
  - Les blocs, les commentaires.
  - Les types de données disponibles.
  - Les variables, l'affichage formaté, la portée locale et globale.
  - La manipulation des types numériques, la manipulation de chaînes de caractères.
  - La manipulation des tableaux dynamiques (liste), des tableaux statiques (tuple) et des dictionnaires.
  - L'utilisation des fichiers.
  - La structure conditionnelle if/elif/else.
  - Les opérateurs logiques et les opérateurs de comparaison.
  - Les boucles d'itérations while et for.
  - Interruption d'itérations break/continue.

Programme

- La fonction range.
- L'écriture et la documentation de fonctions.
- Les Lambda expression.
- Les générateurs.
- La structuration du code en modules.
- Travaux pratiques Installation et prise en main de l'interpréteur Python.

- **Approche Orientée Objet**
  - Les principes du paradigme Objet.
  - La définition d'un objet (état, comportement, identité).
  - La notion de classe, d'attributs et de méthodes.
  - L'encapsulation des données.
  - La communication entre les objets.
  - L'héritage, transmission des caractéristiques d'une classe.
  - La notion de polymorphisme.
  - Association entre classes.
  - Les interfaces.
  - Présentation d'UML.
  - Les diagrammes de classes, de séquences, d'activités.
  - Notion de modèle de conception (Design Pattern).
  - Travaux pratiques Modélisation en UML d'un cas d'étude simple.

- **Programmation Objet en Python**
  - Les particularités du modèle objet de Python.
  - L'écriture de classes et leur instanciation.
  - Les constructeurs et les destructeurs.
  - La protection d'accès des attributs et des méthodes.
  - La nécessité du paramètre Self.
  - L'héritage simple, l'héritage multiple, le polymorphisme.
  - Les notions de visibilités.
  - Les méthodes spéciales.
  - L'introspection.
  - L'implémentation des interfaces.

Programme

- Les bonnes pratiques et les modèles de conception courants.
- L'utilisation du mécanisme d'exception pour la gestion des erreurs.
- Travaux pratiques Pratique des différents concepts objet au travers de l'implantation de l'étude de cas.

- **Utilisation StdLib**
  - Les arguments passés sur la ligne de commande.
  - L'utilisation du moteur d'expressions régulières Python avec le module "re", les caractères spéciaux, les cardinalités.
  - La manipulation du système de fichiers.
  - Présentation de quelques modules importants de la bibliothèque standard : module "sys", "os", "os.
  - path".
  - Empaquetage et installation d'une bibliothèque Python.
  - Les accès aux bases de données relationnelles, le fonctionnement de la DB API.
  - Travaux pratiques Mise en oeuvre de modules Python : expressions régulières, accès à une base de données,

- **Outils QA**
  - Les outils d'analyse statique de code (pylint, pychecker).
  - L'analyse des comptes rendus d'analyse (types de messages, avertissements, erreurs).
  - Extraction automatique de documentation.
  - Le débogueur de Python (exécution pas à pas et analyse post-mortem).
  - Le développement piloté par les tests.
  - Les modules de tests unitaires Python (Unittest.
  -).
  - L'automatisation des tests, l'agrégation de tests.
  - Les tests de couverture de code, profiling.
  - Travaux pratiques Utilisation des outils pylint et pychecker pour la vérification d'un code Python.
  - Mise en oeuvre de tests unitaires.

- **Création IHM TkInter**
  - Les principes de programmation des interfaces graphiques.
  - Présentation de la bibliothèque TkInter.
  - Les principaux conteneurs.
  - Présentation des widgets disponibles (Button, Radiobutton, Entry, Label, Listbox, Canvas, Menu,

Programme

Scrollbar, Text.

-).
- Le gestionnaire de fenêtres.
- Le placement des composants, les différents layouts.
- La gestion des événements, l'objet event.
- Les applications multifenêtres.
- Travaux pratiques Conception d'une interface graphique avec la bibliothèque Tkinter.

- **Interfaçage Python/C**
  - Présentation du module Ctypes.
  - Le chargement d'une librairie C.
  - Appel d'une fonction.
  - La réécriture d'une fonction Python en C avec l'API Python/C.
  - La création de modules C pour Python avec Pyrex.
  - L'interpréteur Python dans C.
  - L'utilisation du profileur de code.
  - Travaux pratiques Appel de fonctions écrites en C depuis Python.
  - Création de modules C pour Python avec Pyrex.

- **Conclusion**
  - Analyse critique de Python.
  - L'évolution du langage.
  - Eléments de Webographie et de bibliographie.

(+212) 5 22 27 99 01

(+212) 6 60 10 42 56

Contact@skills-group.com

Nous sommes à votre disposition :
De Lun - Ven 09h00-18h00 et Sam 09H00 – 13H00

Angle bd Abdelmoumen et rue Soumaya, Résidence Shehrazade 3, 7éme étage N° 30
Casablanca 20340, Maroc