57 rue Labat de Savignac - 31500 Toulouse
info@api-society.com - www.api-society.com - SIREN : 833 816 994

Intitulé de la formation préparant à la certification :

« Développer en langage Python orienté objet » préparant à la certification API Society

Durée : 4 jours ou plus - en présentiel ou distanciel - formateur certifié mention Expert ou Instructeur

Pré-requis : Connaître les bases de l'algorithmie et un langage de programmation

Programme

Le formateur a la liberté pédagogique d'aborder les points du programme dans l'ordre qui lui paraît le plus pertinent, dès lors qu'il traite l'intégralité du programme.

Introduction

Historique (auteur, date de la première version)
Versions de Python (branches 2 et 3)
Caractéristiques du langage (multi-paradigme, typage dynamique fort, syntaxe claire)
Panorama de la bibliothèque standard
Modules d'extension et commande pip
Principe de fonctionnement de l'interpréteur (bytecode PYC)
Interpréteur officiel CPython et autres interpréteurs (micropython, brython, pypy, numba)
Ressources (site Internet python.org, accès aux documentations)
Fonction help() et chaînes documentaires
Principe de l'indentation pour délimiter les blocs d'instruction
Commentaire
Mots-clés réservés
Conventions de nommage
Interpréteur interactif
Programme autonome
Fonctions intégrées élémentaires : print(), type(), input(), len()

Types de données non-modifiables

Utilité des types non-modifiables (optimisation mémoire), fonctions id() et hash(), opérateur is
Principe des séquences ordonnées (str, tuple et list) et collections (dict, set)
Booléen (bool), objets True et False
Nombre (int, float, complex), constructeurs, opérateurs >>, <<, |, &, ^, //, % et **
Notations exponentielle, binaire, octale et hexadécimale, fonctions hex(), oct(), bin(), chr(), ord()
Chaîne de caractères unicode (str), définition avec simple et double guillemets, chaines multilignes avec triple simple ou double guillements, mode raw, constructeur
Indiçage positif et négatif, tranche de valeurs (slice), opérateurs +, * et in, itération

Méthodes incontournables de str : split(), replace(), lower(), upper(), strip(), join()
Chaîne de caractères formatée (%s, %d, %f), méthode format() et f-string
Principe du depacking de variables
Tableau d'octets (bytes), constructeur
Tuple (tuple), constructeur, indiçage, itération, opérateurs +, * et in, méthodes count() et index()
Objet None et fonction repr()

Types de données modifiables

Listes (list), constructeur, indiçage, intération, opérateurs +, * et in, méthodes append(), insert(), fonction del(), méthodes sort(), reverse(), remove(), extend(), pop(), clear()
Listes : manipulation de références, copie superficielle via la méthode copy() ou l'indiçage [:], copie en profondeur avec la fonction deepcopy() du module copy
Fonction sorted()
Principe de fonctionnement des objets itérables
Fonctions reversed() et range()
Dictionnaires (dict), constructeur, indiçage, opérateur in, fonction del(), méthodes keys(), values(), items(), update(), get()
Dictionnaires : manipulation de références, copie superficielle via la méthode copy(), copie en profondeur avec la fonction deepcopy() du module copy
Set (set), constructeur, opérateurs - | & et ^

Structures conditionnelles et répétitives

Structure conditionnelle if ... elif ... else
Opérateur ternaire
Structure répétitive while
Structure répétitive for
Instructions break et continue
Fonction enumerate()
Bloc else sur structure répétitive
Liste en intension (comprehension list), dictionnaire en intension (comprehension dict)

Fonctions, modules et paquets

Définition et appel d'une fonction
Espace de noms local, global, pré-défini (__builtins__) et fonction dir()
Retourner des valeurs, instruction return
Fonctions génériques (duck typing)
Valeurs par défaut
Passage par étiquette
Nombre d'arguments arbitraire (*args, **kwargs)
Fonctions anonymes (lambda)
Fonctions eval(), exec(), map() et filter()
Importation de modules
Création d'un module
Bloc if __name__ == "__main__"
Importation de paquet
Création d'un paquet (__init__.py)
Instruction yield

Manipulation des fichiers

Fonction open() et méthode close()
Méthodes readline() et readlines()
Objet itérable
Instruction with avec les fichiers
Méthodes read() et write()
Méthodes tell() et seek()
Méthode writelines()
Modules complémentaires : struct, csv, json, xml
Sérialisation avec le module pickle
Sérialisation avec le module shelve

Programmation Orientée Objet

Concepts fondamentaux de la POO (séparation du code, encapsulation, héritage)
Notions de classe d'objet, d'objet (instance), d'attribut et de méthode
Définition d'une classe d'objet
Instanciation d'objets, fonction isinstance()
Constructeur (__init__)
Attributs et méthodes
Le paramètre self
Surcharge d'affichage (__str__)
Surcharge d'opérateurs (__eq__, __add__)
Propriété (fonction spéciale property), accesseur et mutateur
Espaces de noms global, de l'objet, de la classe
Variable de classe
Constructeur à nombre d'arguments arbitraire (*args, **kwargs)
Agrégation / Composition
Héritage de classe (généralisation), fonctions issubclass(), super() et méthode mro()

Exceptions

Principe de fonctionnement
Exceptions pré-définies et arbre d'héritage
Instructions try ... except ... else ... finally
Propagation des exceptions
Déclenchement d'exceptions
Définition d'une exception

Modules de la bibliothèque standard

Interaction avec l'interpréteur : module sys
Interaction avec le système d'exploitation : modules os et pathlib
Interaction avec le système de fichiers : module os.path
Expressions rationnelles : module re, fonctions search(), match(), split() et sub()
Tests unitaires : instruction assert, module unittest
Tour d'horizon d'autres modules intéressants de la bibliothèque standard : datetime, math, timeit, urllib, collections, csv, json, sqlite3