Fiche récapitulative - Partie 2

Les fichiers

Python est un excellent langage de programmation pour effectuer des tâches sur son ordinateur. Pour ça, on a souvent besoin de savoir lire et écrire dans des fichiers.

On a vu ensemble le format `JSON` qui permet d'écrire et de lire des objets complexes comme les dictionnaires à l'intérieur d'un fichier.

Grâce au module `os`, on peut récupérer la liste des fichiers dans un dossier, et avec la fonction open lire et écrire dans ces fichiers. On peut ainsi aller récupérer toute sorte d'informations, faire des recherches avancées, créer des structures de centaines de dossiers et tout ça automatiquement et en un temps record.

Les dictionnaires

Dernier objet complexe de cette formation, les dictionnaires permettent de stocker des données selon un concept de clés / valeurs. Bien plus puissant que les listes, les dictionnaires sont partout, notamment avec l'avènement des api `REST`, très utilisées dans le web, et qui reposent sur cette façon de représenter les données (avec le format `JSON`).

Plus complets et aussi plus complexes à utiliser, il vous faudra un peu de temps avant de maîtriser totalement les dictionnaires. N'hésitez pas à refaire les exercices et à profiter de chaque occasion pour organiser vos données dans des dictionnaires pour vous familiariser avec le concept.

N'oubliez pas que les clés des dictionnaires sont uniques !

Et n'oubliez pas également l'utilité de la méthode `get` qui permet d'éviter les erreurs lorsqu'une clé n'est pas présente dans un dictionnaire.

La gestion des erreurs

Quand on crée des programmes, on passe la majeure partie de notre temps à gérer et éviter les erreurs potentielles qui peuvent se produire. Pour ça, on a

parfois besoin de tester certains bouts de code sans savoir au préalable si celui-ci va fonctionner.

On a ainsi vu deux façons de faire avec les méthodes `LBYL` (**L**ook **B**efore **Y**ou **L**eap) et `EAFP` (**E**asier to **A**sk for **F**orgiveness than **P**ermission).

Avec la méthode `EAFP`, on a découvert les blocs `try` / `except` qui nous permettent d'exécuter du code sans faire planter notre script et de gérer les erreurs potentielles.

N'oubliez pas de toujours préciser les erreurs que vous souhaitez récupérer dans votre bloc `try`.

Un bloc `try` qui cible toutes les erreurs peut masquer des problèmes potentiels et vous faire tourner en rond pendant un bon moment !

Les fonctions

Notion essentielle de la programmation, les fonctions permettent d'organiser votre code en différents blocs et d'éviter ainsi les répétitions.

D'apparence simple, il y a pas mal de petits détails auxquels il faut porter attention quand on crée et qu'on utilise des fonctions.

L'ordre des paramètres, les notions d'espaces global et local, l'utilisation des paramètres à l'intérieur de la fonction, les valeurs retournées, le passage par valeur ou par référence, autant de petits détails qui peuvent poser problème si on n'y fait pas attention.

Dans cette partie on a vu comment créer des fonctions, comment les appeler, comment modifier le code à l'intérieur d'une fonction grâce aux arguments que l'on envoie aux paramètres et comment récupérer les valeurs retournées par une fonction.

C'est vraiment un élément essentiel de la programmation alors n'hésitez pas à refaire des exercices et à essayer d'organiser vos scripts en différentes fonctions, cela vous permettra de vous habituer à passer des valeurs d'une fonction à une autre et à organiser votre code de façon plus logique.

Les modules et les packages

Avec les modules et les packages, on commence à voir au-delà de notre simple script. On passe notre temps avec Python à utiliser des modules qui existent déjà, mais il est bien entendu possible de créer nos propres modules.

Comme pour les fonctions, il s'agit là encore d'organiser notre code en différents fichiers cette fois-ci.

Et comme avec les fonctions, c'est avec la pratique que vous verrez de mieux en mieux comment scinder votre code en différents fichiers.

Pour le moment, retenez bien les concepts que l'on a vus et entraînez-vous à séparer votre code en différents fichiers pour pouvoir mieux maintenir et organiser vos programmes.

Documenter son code

Document son code c'est **LA** chose que l'on ne fait jamais quand on est débutant en programmation. Pourtant, c'est grâce à la documentation que l'on gagne énormément de temps.

Que ce soit pour soi-même, en nous permettant de revenir des mois plus tard dans un programme et de toujours comprendre ce que celui-ci fait.

Et pour les autres, en leur permettant de comprendre avec des mots ce que notre script fait avec du code.

Le logging

Le logging, qu'on appelle en français 'journalisation' (quel mot...) permet de rendre nos scripts un peu plus robustes et professionnels en affichant différents types de messages selon les situations.

Quand on commence à programmer, on abuse des `print` (et c'est une bonne chose !) pour afficher tout type d'informations qui nous permettent de mieux comprendre nos scripts.

Le logging nous permet d'aller encore plus loin et d'utiliser un système beaucoup plus avancé et pensé précisément pour n'afficher que ce dont on a besoin selon la situation.

Avec les différents niveaux de logging (`DEBUG`, `INFO`, `CRITICAL`...), on peut maîtriser le type d'informations que l'on souhaite afficher en fonction des circonstances.

En plus, il est possible d'écrire tous ces messages dans un fichier de log sur notre disque dur et ça c'est primordial pour la plupart des scripts qui fonctionnent sans notre supervision.

En cas de problème, il est ainsi beaucoup plus aisé de voir après coup ce qui a pu mal tourner dans un script pour prendre les actions nécessaires afin de réparer les problèmes.

Installer des packages avec PIP

Une des plus grandes forces de Python, c'est le nombre de packages disponibles qui nous permettent de ne pas ré-inventer la roue constamment. Avec plus de 272,000 packages disponibles sur PyPi.org, il existe une solution à presque tous nos problèmes.

Avec pip, on a vu comment nous pouvions installer ces packages pour les utiliser dans nos scripts.

C'est un utilitaire qui peut faire peur au début car il faut utiliser le terminal et des instructions en ligne de commande, mais une fois que vous connaissez les quelques commandes à utiliser, c'est un vrai jeu d'enfant.

Les environnements virtuels

L'utilisation de bibliothèques externes comporte son lot de problèmes et notamment les conflits qui peuvent arriver lorsqu'un de nos programmes doit utiliser une version différente d'une bibliothèque.

Pour gérer ces cas de figure, on a recours aux environnements virtuels qui nous permettent d'isoler différents environnements de Python avec chacun leurs librairies, ce qui permet par exemple d'avoir deux projets qui utilisent une version différente d'une même librairie.

C'est primordial quand on commence à travailler sur plusieurs projets et Python nous facilite grandement la tâche avec plusieurs modules qui permettent de créer en un rien de temps un environnement complet et isolé.

Dans cette formation on a vu le module `venv` et comment l'utiliser, là encore avec le terminal et des instructions en ligne de commande.

Cela peut être assez déroutant au début, surtout si vous utilisez Windows, alors n'hésitez pas à poser des questions sur le Discord si vous n'êtes pas sûr de comment réaliser toutes ces étapes ou de comment interpréter les possibles erreurs qui vous seront retournées.

La programmation orientée objet

La programmation orientée objet... tout un programme !

On pourrait faire une formation entière de 10h sur la programmation orientée objet.

C'est un paradigme complet qui peut prendre des années à être maîtrisé correctement.

Dans cette formation, on a vu les bases de la programmation orientée objet. Tous les concepts de base comme les classes, les instances, les attributs, les méthodes et comment les utiliser dans des cas concrets.

Je ne vais pas tout détailler de nouveau ici, ne vous inquiétez pas si certaines notions sont encore assez floues où si vous n'avez pas tout retenu.

Regardez de nouveau les parties de cette formation sur la POO, faites les projets, respirez, sortez dehors, et refaites tout ça encore et encore et vous finirez par y arriver.

Personnellement, la programmation orientée objet, ça m'a pris facilement 6 mois à comprendre à quoi ça pouvait servir et comment vraiment l'utiliser. Donc pas de panique si vous ne comprenez pas tout au bout de 2 jours de formation sur le sujet 🙂

Les bases de données

On avait vu précédemment dans la formation comment stocker des données notamment dans des fichiers `JSON`.

C'est pratique pour des petits programmes, mais quand on commence à vouloir aller plus loin et réellement organiser ses données, on est obligé de passer par une vraie base de donnée et un langage comme le `SQL`.

Dans cette partie, on a vu comment utiliser le module `sqlite3` avec Python pour créer, modifier, supprimer des tables et des entrées dans une base de donnée.

Ce n'est bien sûr qu'un aperçu rapide de ce qu'il est possible de faire, là encore on pourrait consacrer une formation entière de 10h sur le sujet.