Fiche 3.0 : scripts et modules

BUT Informatique
IUT de Vélizy

1 Notion de script

Jusqu'à présent, vous avez écrit du python dans l'environnement interactif *jupyter*. Mais en général les programmes python sont indépendants de ce genre d'environnement et peuvent être exécutés "directement".
Un *script* python est simplement un fichier texte, dont l'extension est en général `.py` et qui contient du code. Un programme python est en général un ensemble de scripts.
En python, un script s'appelle un *module*, il y a quelques nuances mais ça ira pour l'instant. Plus de détails dans `https://docs.python.org/fr/3/tutorial/modules.html`
Pour écrire un tel script, on peut utiliser un simple *éditeur de texte* (gedit, vim, emacs, geany, sublimetext,...) ou un *environnement de développement intégré* (IDE) (eclipse, pycharm, jetbrains,vscode,...), qui est un éditeur de texte avec de nombreuses fonctions supplémentaires pour écrire du code.
Dans cette séquence, afin de ne pas tout mélanger, nous utiliserons l'éditeur **geany** qui est très simple. Par la suite, vous ferez comme vous voudrez.

2 Prise en main

☐ 0. Ouvrez l'éditeur de texte *geany*. Tapez quelques lignes de code dans l'éditeur, par exemple effectuez une boucle pour écrire 10 fois "bonjour".

☐ 1. Enregistrez dans votre dossier de la sequence3 ce programme sous le nom `first_script.py`

☐ 2. Maintenant, ouvrez un terminal et naviguez en ligne de commande jusqu'à vous trouver dans le répertoire où se trouve le script.

☐ 3. Tapez dans la ligne de commande

```
python3 first_script.py
```

N'oubliez pas que la touche TAB permet de compléter automatiquement. En principe, vous devriez voir dans le terminal la sortie de votre programme. Sinon, c'est peut-être que vous avez un message d'erreur.

☐ 4. Retenez la procédure ci-dessus, il faudra toujours la faire pour exécuter les programmes python.

3 Ecriture d'une librairie pour gérer les trucs à faire

Quand vous exécutez le script, python va tout lire et exécuter, dans l'ordre du script, de haut en bas. Quand python tombe sur la définition d'une fonction,(`def ...` il n'exécute pas le code de la fonction, il prend juste note de son existence. Pour vraiment exécuter la fonction, il faut l'appeler !

Vous allez maintenant réaliser un petit module avec quelques fonctions ayant pour but de gérer une liste de tâches à réaliser ou `todolist`. Plus précisément, on veut avoir en mémoire un ensemble de tâches à réaliser comme :

```
Tâches priorité haute :
- réviser cours dev
- appeler Mamie
Tâches priorité moyenne :
- vaisselle
Tâches priorité basse :
- finir de regarder F&F 42
- réviser encore cours dev
```

Vous pouvez imaginer autre chose et faire des variations plus avancées sur ce principe (possibilité de prévoir des tâches à des dates fixées, etc.)

☐ 5. Créez deux scripts, `todolist.py` et `test_todo.py`. Le premier contiendra l'ensemble des fonctions pour manipuler la todolist mais pas de code qui s'exécute directement, et le second les tests. Vous n'exécutez jamais le premier directement, vous exécutez le module de test qui lui se servira des fonctions écrites dans l'autre.

☐ 6. A vous de décider comment vous allez gérer la sauvegarde des tâches et leur priorité. Créez une fonction qui affiche l'ensemble des tâches comme ci-dessus. Bien entendu, vous écrivez la spécification de la fonction.

☐ 7. Dans votre fichier de tests, qui doit se trouver dans le même répertoire, écrivez `import todolist` avant votre code. Ecrivez maintenant une fonction de test de votre affichage et appelez-la.

☐ 8. Maintenant, à chaque fois vous devez écrire la fonction, sa spécification, et la fonction de test. De préférence, d'abord la fonction de test. Créez une fonction pour ajouter une tâche. Attention, je n'ai pas dit saisir une tâche au clavier.

☐ 9. Ecrivez aussi des fonctions pour supprimer une tâche, ou pour archiver une tâche déjà effectuée, pour changer la priorité d'une tâche, et toute autre fonction que vous pourrez imaginer qui pourrait servir dans votre librairie.

☐ 10. Ecrivez un test complet de votre librairie : cela crée une todolist, ajoute quelques tâches, les affiche, supprime et ajoute d'autres tâches, affiche, etc.