# Scripting – B2

# Python

# Exercices

Créer un fichier .py exécutable pour chacune des étapes.

## 1 – Manipulation de bases

a. Demander deux nombres à l'utilisateur et afficher le résultat de l'addition des deux nombres. Contrôler que l'utilisateur a bien saisi deux nombres et pas autre chose

b. Demander une saisie utilisateur de plusieurs prénoms. L'utilisateur peut choisir d'arrêter la saisie en appuyant sur la touche 'q'. Les stocker dans un dictionnaire. Afficher le résultat en ordonnant les prénoms par ordre alphabétique.

c. Demander une saisie utilisateur de plusieurs notes. Afficher la moyenne de notes saisies, lorsque l'utilisateur décide d'arrêter la saisie.

Dans un deuxième ajoutez la gestion des noms : l'utilisateur saisit un prénom et une note à pour chaque entrée que vous stockerez (un dictionnaire Python serait l'idéal pour stocker ces données). Affichez ensuite la moyenne et un top 5 des meilleures notes.

Coder un programme qui contient les trois exercices précédents. Chacun de ces programmes doit être dans une fonction. Un menu se présente à l'utilisateur pour choisir quel fonctionnalité il veut utiliser (addition, tri alphabétique, moyenne).

Habituez-vous à utiliser des fonctions :)

d. Jeu du plus ou moins. Générer un nombre aléatoire (entre 0 et 100 par exemple). Demander une saisie utilisateur tant qu'il ne trouve pas le nombre que vous avez généré. Afficher à chaque nombre saisi s'il est plus grand ou pus petit que le nombre que vous avez généré.

## 2 – Next step

e. Jeu du plus ou moins qui lit dans un fichier. Développer un jeu du plus ou moins qui tourne en démon. Pour ce faire, il “suffit” de créer une boucle infinie qui lit dans un fichier et qui attends un certain temps. A la place de saisir un nombre en exécutant le fichier, il suffira d’écrire un nombre à l’intérieur d’un fichier spécifique, qui sera lu par le programme démon. Le programme démon pourra alors écrire “plus grand” ou “plus petit” à l’intérieur du fichier.

f. Ecrire un programme qui résout le programme précédent.

## 3 – Sauvegarde (module shutil, gzip, os, sys)

g. Ecrire un script qui permet d’effectuer une sauvegarde incrémentale.

Une sauvegarde incrémentale permet de ne sauvegarder uniquement les éléments nouveaux par rapport à des sauvegardes précédentes.

Les sauvegardes doivent être stockées au format tar.gz

Le contenu des archives compressées (tar.gz) ne doit être mis à jour que s’il y a eu une différence avec la sauvegarde précédente (plusieurs méthodes sont possibles, essayez d’en trouver une judicieuse !).

Utiliser le module sys pour afficher des informations dans la sortie standard plutôt qu’avec print, et pour retourner un code d’erreur spécifique en cas d’échec du programme.

Utiliser le module signal pour intercepter le signl SIGINT et effectuer une action personnalisée.

Gérer les exceptions et affichez dans la sortie d’erreur un texte personnalisé en cas d’échec d’une étape importante (pas les droits sur le fichier, ou sur la destination, ou encore, en cas de remplissage du disque par exemple).

h. Adaptez le script de sauvegarde précédent pour être lancé avec des options (module argparse). Si l’utilisateur le souhaite, il peut choisir de copier l’archive résultante sur un serveur distant SSH. Les options permettront de préciser un utilisateur et un mot de passe

pour la connexion SSH.
Pour ce faire utiliser le module paramiko et le module scp suivant : https://github.com/jbardin/scp.py.

i. Faire en sorte que le script de sauvegarde tourne en démon et surveille la taille d'un répertoire sur le disque, ou plusieurs. La liste des répertoires à surveiller se trouve dans un fichier spécifique (fichier de configuration).

Inclure à ce fichier de configuration la configuration SSH dans le cas d'une sauvegarde distante.

Faire en sorte que ce démon se lance avec systemd.

j. Développer un HIDS simple. C'est un détecteur d'intrusion. Son but est de détecter les changements dans des répertoires à surveiller, et d'en faire des rapports. Utilisez plusieurs moyens différents pour déterminer si un fichier a changé ou nom (plusieurs fonctions de hachage par exemple). Vous aurez besoin de conserver une trace de chacune de vos analyses pour comparer avec chaque novuelle analyse. Vous utiliserez, pour ce faire, une base de données (de type MySQL par exemple), il est parfaitement possible d'utiliser le module SQLAlchemy pour ce faire.