Informatique 2 : Travaux Pratiques

Programmes avancés

Exercices avancés

Question 1: (🕓 *10 minutes*) Ecrire un programme qui récupère deux entiers **n** et **e** en arguments de la ligne de commande avec le module **argparse** et qui affiche le résultat de **n** à la puissance **e** (**n******e**). L'argument **e** est optionnel et vaut 1 par défaut.

>_ Exemple

python3 programme.py -e 3 -n 2 affiche **8** et *python3 programme.py -n 5* affiche **5**.

Question 2: (🕓 *15 minutes*) Ecrire un programme, à l'aide du module **argparse**, qui affiche la racine d'un nombre. Le programme aura deux paramètres :

- -n/–entier l'entier pour calculer la fonction racine (obligatoire).
- -o/–output si présent, enregistre le résultat dans un fichier *resultat.txt* (optionnel, faux par défaut).

Question 3: (🕓 *25 minutes*) Ecrire un programme qui prend des nombres passés en arguments de la ligne de commande et qui leur applique une fonction également passée comme argument de la ligne de commande (en dernier argument). Cette fonction peut être '+' pour la somme ou '*' pour le produit. Rediriger la sortie standard vers le fichier *output.txt*.

>_ Exemple

python3 programme.py 1 2 3 4 + affichera **10** et *python3 programme.py 1 2 3 4 **</* affichera **24**.

Question 4: (🕓 *20 minutes*) Reprendre l'exercice précédent, mais cette fois en utilisant le module **argparse**. Le programme aura plusieurs paramètres :

- -v/–verbose qui determine s'il faut afficher des messages d'erreur (optionnel, faux par défaut, voir **'store_true'**). Afficher sur la sortie erreur **'Mode verbose actif'** et **'n entiers lus'** avec n le nombre d'entiers passés en arguments.
- -n/–number qui contiendra la liste des nombres (obligatoire, doit contenir au moins un élément).
- -o/–operator qui détermine l'opération à utiliser (obligatoire). On utilisera l'option *choices* pour restreindre les valeurs possibles pour cet argument.

Question 5: (🕓 *10 minutes*) Ecrire un programme qui renomme les fichiers du répertoire courant, dont le nom commence par *test* et se termine par *.txt* en *1.txt, 2.txt, . . .* . La numérotation devra respecter l'ordre alphabétique.

>_ Exemple

Si le répertoire contient 4 fichiers : *programme.py, test_ab.txt, test_aa.txt* et *test_de.txt*, alors les fichiers devraient être renommés comme suit :

- test_ab.txt devient 2.txt
- test_aa.txt devient 1.txt
- test_de.txt devient 3.txt