UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Outils Info. TP2 : Listes, Boucles, Accumulateurs

Important : rendu sous forme d'archive

À la fin, vos réponses seront rendues dans une archive zip : cette archive contiendra un fichier `compte-rendu.txt` (avec votre **nom**, **groupe**, et les réponses aux questions précédées d'une apostrophe) et vos fichiers pythons.

Pour faire une **archive zip** de votre TP pour le rendu, exécutez, dans le terminal :

```
cd ~/outils-info/
zip -r tp2.zip tp2/
```

Important : mise en place

Comme pour tous les TPs :

- travaillez dans un dossier dédié au TP, lui même dans `~/outils-info`,
- créez un fichier `compte-rendu.txt` pour écrire la date, vos noms et les réponses aux questions précédées d'une apostrophe,
- créez un sous-dossier par exercice (sauf instructions contraires).

Pour démarrer, vous pouvez par exemple exécuter les commandes :

```
cd ~/outils-info/
mkdir tp2
cd tp2
emacs compte-rendu.txt &
```

Exercice 1 – Listes, boucles for, range

'Q1) Créez et ouvrez un fichier `fruits.py` pour écrire le programme des questions à venir, Quelle(s) commande(s) avez-vous utilisée(s) ?

Q2) Définissez une variable `liste_fruit`, initialisée avec une liste contenant trois chaînes de caractères qui sont des noms de fruits (par exemple banane, noix, myrtille).

Q3) Faites que votre programme affiche « `Il y a 3 fruits` » mais en prenant soin de faire que le « `3` » soit calculé comme étant la longueur de la liste.

'Q4) Testez votre programme. Quelle commande avez-vous lancé ?

IMPORTANT : il faut maintenant que vous lanciez votre programme à chaque question, pour voir s'il fait ce que vous voulez. Ceci est valide pour l'ensemble des questions de l'ensemble des TP (sauf cas vraiment particulier).

Q5) Faites qu'en plus votre programme affiche chaque élément de la liste (chacun sur une ligne) précédé de deux espaces et un tiret, par exemple la première ligne serait « `  - banane` ».

Q6) Créez et ouvrez un fichier `compte_dix.py` pour écrire le programme des questions à venir.

Q7) Écrivez un programme qui affiche, chacun sur une ligne les nombres de 1 à 10.

Q8) Faites que votre programme affiche en plus la racine carrée de chacun de ces nombres.

Exercice 2 – Interaction utilisateur

Q9) Créez et ouvrez un fichier `affiche_input.py` pour écrire le programme des questions à venir.

Q10) Écrire un programme qui demande à l'utilisateur de rentrer une chaîne de caractères, par exemple un nom de planète, et pour l'instant l'affiche tel quel.

Q11) Faites que votre programme affiche la chaîne entrée par l'utilisateur mais verticalement. Pour cela, il faut afficher les caractères un à un, chacun sur une ligne. Rappel : on peut parcourir et indexer une chaîne de caractères comme si c'était une liste de caractères.

Q12) Créez et ouvrez un fichier `compte_input.py` pour écrire le programme des questions à venir.

Q13) Écrivez un programme qui fait comme `compte_dix.py` mais s'attend à ce que le nombre soit tappé par l'utilisateur. Si on le lance et tappe `10` le programme fera exactement comme `python3 compte_dix.py`. Si on le lance et tappe `20` le programme ira jusqu'à afficher 20 et sa racine.

Nous allons utiliser la fonction `split` qui permet de prendre une chaine de caractères et de la découper en sous-partie, par exemple en utilisant l'espace comme séparateur. Si l'on écrit par exemple `li = str.split("bonjour toto !", " ")`, alors la valeur retournée (`li`) sera une liste contenant 3 éléments : les chaînes de caractères `bonjour`, `toto` et `!`. On peut écrire indifféremment `str.split("a b c", " ")` et `"a b c".split(" ")`. Cela fonctionne aussi bien avec n'importe quelles expression (variables, etc).

Q14) Créez et ouvrez un fichier `affiche_split.py` pour écrire le programme des questions à venir.

Q15) Créez une variable `entree` qui est une chaîne de caractère avec comme valeur `Bonjour Toto !!!`.

Q16) Utilisez la fonction `split` pour obtenir une liste (que l'on pourra appeler `parties`) en découpant la chaîne au niveau des espaces.

Q17) Faîtes que votre programme affiche « `Il y a 3 partie(s).` » où le « `3` » est calculé comme étant le nombre d'élément dans la liste `parties`.

Q18) Modifiez votre programme pour qu'il demande à l'utilisateur d'entrer la valeur de `entree` (au lieu d'utiliser la valeur ci-dessus) et affiche toujours le nombres de parties (qui dépend maintenant de ce que l'utilisateur aura tappé).

'Q19) Testez votre programme avec différentes valeurs. Quelles valeurs avez vous utilisé pour vérifier que le programme fonctionne ?

Q20) Faites que votre programme affiche aussi chaque partie sur sa propre ligne, suivant le format suivant : « `  - partie 0 : toto` » (on adopte la convention Python avec le premier indice à 0).

Exercice 3 – Technique de l'accumulateur

Dans cet exercice, chaque programme doit demander un entier « n » à l'utilisateur et afficher le résultat. Supposons que l'on veut par exemple calculer la somme $S_n = 1 + 2 + 3 + \cdots + n$ avec la technique de l'accumulateur. Nous allons créer une variable (l'accumulateur) que l'on mettra à jour progressivement grâce à chaque élément de la somme.

```
1 acc = 0
2 for element in range(1,n+1) :
3     acc = acc + element
4 print(acc)
```

Cette technique fait donc apparaître 3 parties :

- ligne 1 : initialisation de l'accumulateur avec la valeur « neutre pour la somme » c'est à dire 0,
- ligne 2 : parcours de la liste des entiers des 1 à n (liste retournée par `range(1,n+1)`,
- ligne 3 : mise à jour de l'accumulateur, à chaque fois,
- ligne 4 : exploitation de la valeur calculée (ici pour l'afficher).

Dans les questions ci-dessous, utilisez la technique de l'accumulateur après avoir demandé la valeur de n à l'utilisateur.

Q21) Écrivez un programme `fractions.py` permettant de calculer $Q_n = \frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1}$.

Q22) Écrivez un programme `factoriel.py` permettant de calculer $n! = 1 \times 2 \times 3 \times \cdots \times n$.

Q23) Écrivez un programme `plusmoins.py` permettant de calculer $A_n = 1 - 2 + 3 - \cdots \pm n$.

Q24) Écrivez un programme `moyenne.py` permettant de calculer $\mu_n = \frac{1+2+3+\cdots+n}{n}$.

Q25) Challenge : `rerere.py` doit calculer $R_n = 1 + \cfrac{n}{1+\cfrac{n-1}{1+\cfrac{\cdots}{1+\cfrac{2}{1+\cfrac{1}{1+0}}}}}$ (avec n divisions).

Exercice 4 – Moyenne des nombres

Q26) Écrivez un autre programme `moyenne.py` (ce coup ci dans le dossier `ex4`) qui attend un nombre quelconque de nombres (grâce à `split`). Ces nombres correspondent à une liste de notes (des nombres réels). Le programme doit calculer et afficher la moyenne de ces notes. Le programme doit par exemple afficher `La moyenne est 9.775` quand on le lance et entre `12.3 12 6.5 8.3`.

Q27) Écrire un programme `notespond.py` qui reçoit un nombre pair de nombres qui sont à chaque fois « un coefficient suivi d'une note ». Le programme doit afficher, pour chaque « coeff. et note », une phrase de la forme `une note de 16 avec un coeff de 33.333`.

Q28) Faites que le programme calcule et affiche la moyenne pondérée (somme des « notes multiplié par coeff. » divisé par la somme des « coeff. ».