UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Outils Info. - TD : Feuille 3

Exercice 1 – Lire une liste

Q1) Écrire un programme qui demande d'entrer une liste de mots (séparés par des espaces) et les affiche chacun sur une ligne, en écrivant le numéro de la ligne. Utilisez pour cela une boucle for, et les fonctions `len`, `range` et `split` (entre autre). Si l'utilisateur tape par exemple `Bonjour Toto`, le programme affichera 2 lignes : `1. Bonjour` et `2. Toto`.

Exercice 2 – Racines de nombres

Q2) Écrire un programme Python qui demande une liste de nombres réels, puis parcours et affiche ces nombres avec leurs racines cubiques (puissance $\frac{1}{3}$), chacun sur une ligne.

Q3) Comment faire pour que ce programme affiche « `Merci de tapper au moins deux nombres` » dans le cas où on ne lui passe qu'un (ou zéro) nombre ?

Exercice 3 – Patron accumulateur

Rappel : il est fréquent d'avoir à parcourir une liste pour calculer un résultat. On utilise la « technique de l'accumulateur » pour résoudre de nombreux problèmes de ce type. Le principe est le suivant :

- on initialise une variable (l'accumulateur), avant la boucle,
- pour chaque élément de la liste (chaque itération de la boucle), on calcule une nouvelle valeur de l'accumulateur en fonction de sa valeur précédente et de la valeur de l'élément courant,
- après la boucle, on utilise la valeur de l'accumulateur.

Q4) Écrire un programme qui crée une liste contenant des entiers (par exemple 3, 5, 7, 11, 13, 17) et qui calcule et affiche la somme de ces entiers (sans utiliser la fonction `sum`).

Q5) Écrire un programme qui crée une liste contenant des entiers (par exemple 3, 5, 7, 11, 13, 17) et qui calcule et affiche la moyenne de ces entiers.

Q6) Écrire un programme permettant de calculer et afficher $S_n = 1 + 2 + 3 + \ + n$, où n est un entier demandé à l'utilisateur (sans utiliser la fonction `sum`).

Q7) Écrire un programme permettant de calculer et afficher $n! = 1 \times 2 \times 3 \times ... \times n$, où n est un entier demandé à l'utilisateur.

Q8) Écrire un programme qui demande à l'utilisateur une liste de nombres (réels) et calcule et affiche la somme de ces nombres (sans utiliser la fonction `sum`).

Q9) Écrire un programme qui demande à l'utilisateur une liste de nombres. La liste retournée par `split` contient des `str` (*string*, chaînes de caractères). Faire que votre programme, crée une (autre) nouvelle liste et la remplisse progressivement avec les `float` (nombres à virgule flottante, nombres réels) correspondants. Le programme doit ensuite utiliser cette autre liste pour calculer et afficher la somme des nombres. Vous pourrez utiliser les fonctions `l.append(e)` et `sum(l)`.

Exercice 4 – Sommes Pondérées

Nous allons écrire des programmes qui calculent une moyenne pondérée, par exemple la moyenne d'une liste de notes avec des coefficients. Les programmes finaux auront tous les deux la même « interface

utilisateur », par exemple, si on lance un des programmes et tape `33.333 16 66.667 10`, cela affichera (environ) `la moyenne pondérée est 12.0`. On appellera ici « paramètres » les valeurs tapées.

NB:

- chaque note est associée à un poids,
- le programme n'est pas limité à 2 notes
- la somme des poids n'est pas nécessairement de 100, ($m_{pond} = \frac{\sum_{i=1}^{N} poids_i \times note_i}{\sum_{i=1}^{N} poids_i}$)

Q10) Écrire un programme `moypond0.py` qui parcourt les paramètres et affiche les valeurs et poids (sans calculer la moyenne). Dans l'exemple ci-dessus, il affiche `une note de 16 avec un coeff de 33.333` et `une note de 10 avec un coeff de 66.667`.

Q11) Écrire un programme `moypond1.py` qui manipule deux variables `somme_poids` et `somme_ponderee` pour calculer progressivement la somme des poids et la somme pondérée des valeurs, pour finalement afficher la moyenne pondérée.

Q12) Écrire un programme `moypond2.py` qui rempli progressivement deux listes `poids` et `poidsvaleurs` contenant les poids et les poids multiplié par la valeur (note). Le programme doit finalement utiliser ces deux listes pour afficher la moyenne pondérée. Vous pourrez utiliser les fonctions `l.append(e)` et `sum(l)`.

Exercice 5 – Utilisation de qtido

Rappel: la feuille de TD 2 liste les fonctions disponibles dans la bibliothèque qtido.

En exemple simple qui trace deux carrés (un rouge sur un bleu) peut s'écrire comme suit :

```
1
2  from qtido import *
3
4  f = creer(800, 600)
5
6  couleur(f, 0, 0, 1)
7  rectangle(f, 300, 200, 500, 400)
8
9  couleur(f, 1, 0, 0)
10 rectangle(f, 350, 250, 450, 350)
11
12 attendre_fermeture(f)
```

Q13) Écrire un programme qui utilise la bibliothèque qtido pour ouvrir une fenêtre graphique de 400 pixels de large et 400 de haut, la remplit en rouge vif, puis attend (pour terminer le programme) que l'utilisateur ferme la fenêtre.

Q14) Écrire un programme qui affiche un damier rouge et gris de 8x8 cases, chacune de 50x50 pixels. Aide – Essayer de décomposer le problème : comment tracer une case donnée ? comment tracer une des lignes du damier ? etc.

Q15) Écrire un programme qui affiche un damier dont les tailles sont demandées à l'utilisateur. Par exemple, il génère le même damier que précédemment si on lui tape `8 8 50 50`.

Q16) Écrire un programme qui affiche un damier dont les tailles et les couleurs sont demandées à l'utilisateur. Par exemple, il génère le même damier que précédemment si on lui tape `8 8 50 50 1 0 0 0.5 0.5 0.5`.

Q17) Ajouter un paramètre *optionnel* au programme précédent, qui est le nom du fichier image où sauver le damier. Par exemple, `8 8 50 50 1 0 0 0.5 0.5 0.5` affichera le damier, mais `8 8 50 50 1 0 0 0.5 0.5 0.5 damier.png` sauvegardera le damier sous forme d'une image `damier.png`.

Q18) Écrire un programme qui affiche une spirale