TP3: listes

Le document de référence doit être apporté en TP. Les exercices 1 à 6, incontournables, doivent être parfaitement traités en séance de TP. Les exercices suivants peuvent être traités en fin de séance ou à la maison, pour s'entraîner. Tous ces exercices sont susceptibles d'être posés en DS.

Créer un dossier "TP3" dans votre répertoire personnel. Pour l'exercice 1, dans Pyzo, enregistrer un fichier nommé "exo1.py" dans le dossier "TP3". Pour l'exercice 2, dans Pyzo, enregistrer un fichier nommé "exo2.py" dans le dossier "TP3"..... Pour chaque exercice, le fichier python sera régulièrement sauvegardé et éxécuté intégralement avec la commande "Run file as script" du menu Run (Ctrl+Shift+E). Les instructions d'affichage seront saisies sur le fichier python. Au fur et à mesure de l'avancée de l'exercice, les instructions devenant inutiles et gênantes pourront être désactivées en utilisant le caractère #.

Exercice 1. *Recopier et faire éxécuter la suite d'instructions suivante et examiner les affichages afin de s'assurer d'avoir compris le fonctionnement.*

$l = [9, 2, 5, 7, 12, 16, 7, 8]$
$print(l)$
$n = len(l)$
$print(n)$
$print(l[0])$
$print(l[1])$
$print(l[2])$
$print(l[n-1])$
$l[2] = -6$
$print(l)$

Exercice 2.

1. *Affecter à la variable l la liste $[-7, 2, 5, -8, 2, 9]$. Faire afficher l.*
2. *Affecter à la variable n la longueur de la liste.*
3. *A l'aide d'une boucle, faire afficher les couples $(i, (l[i])^2)$ avec i indice de la liste.*
4. *Ecrire une fonction "somme" ayant pour argument l (liste non vide de nombres) et retournant la somme des termes de l. Faire afficher $somme(l)$.*
5. *Ecrire une fonction "moy" ayant pour argument l (liste non vide de nombres) et retournant la moyenne des termes de l. Cette fonction devra appeler la fonction somme. Faire afficher $moy(l)$.*
6. *Ecrire une fonction "nbpos" ayant pour argument l (liste de nombres) et retournant le nombre de termes positifs de cette liste. Faire afficher $nbpos(l)$.*

Exercice 3. *Recopier et faire éxécuter la suite d'instructions suivante et examiner les affichages afin de s'assurer d'avoir compris le fonctionnement.*

$l = []$
$print(l)$
$l.append(8)$
$print(l)$
$l.append(7)$
$print(l)$
$l.append(4)$
$print(l)$

Méthode pour construire une liste par récurrence :
On affecte à une variable l la liste vide puis on ajoute à la liste l des termes en fin de liste, les uns après les autres, en utilisant notamment une boucle.

Exercice 4.

1. *Ecrire une fonction nommée "carres" ayant pour argument n (entier naturel) et retournant la liste* $[0^2, 1^2, ..., (n-1)^2]$. *Faire afficher* $carres(10)$.
2. *Ecrire une fonction nommée "licon" ayant pour argument n (entier naturel) et c (objet) et retournant la liste* $[c, c, ..., c]$ *de longueur n. Faire afficher* $licon(10, 0)$.

Exercice 5 (Recherche séquentielle)**.** *Dans cet exercice, le terme tableau désigne une liste.*

1. *Ecrire une fonction nommée "rechsequ" ayant pour arguments t (tableau) et x (objet) et retournant un indice k tel que* $t[k] = x$ *si il existe un tel indice, et retournant None sinon. Faire afficher* $rechseq([9, 3, 5, 4, 8, 2, 7], 8)$ *et* $rechseq([9, 3, 5, 4, 8, 2, 7], 1)$.
2. *Déterminer, en fonction de n, le nombre de comparaisons == effectuées pour éxécuter* $rechseq(t, x)$ *avec t tableau de longueur n, dans le pire des cas (ie au maximum) ? Quel est ce pire des cas ?*

Remarque.

Le nombre de comparaisons == effectuées pour éxécuter $rechseq(t, x)$ avec t tableau de longueur n est, dans le pire des cas, de l'ordre de grandeur de n. On dit que la complexité dans le pire des cas est linéaire en ordre de grandeur.

Exercice 6.

1. *Ecrire une fonction "ispos" ayant pour argument l (liste de nombres) et retournant True si tous les termes de la liste sont positifs et False sinon.*
2. *Tester la fonction ispos avec des listes de longueur 2. On choisira des listes représentatives des différents cas de figures. Tester aussi la fonction avec une liste particulière.*

Exercice 7.

1. *Ecrire une fonction nommée "ppi" ayant pour argument l (liste de nombres non nulle) et retournant le plus petit indice k tel que* $l[k] \neq 0$. *On utilisera une boucle "Tant que". Faire afficher* $ppi([0, 0, 1, 2, 0, 3])$.
2. *Ecrire une fonction nommée "ppibis" ayant pour argument l (liste de nombres) et retournant le plus petit indice k tel que* $l[k] \neq 0$ *si l est non nulle, ou retournant la longueur de l si l est nulle. On complètera :*

```
def ppibis(l) :
    n=len(l)
    i=...
    while ... and ... :
        ...
    return(i)
```

Faire afficher $ppibis([0, 0, 1, 2, 0, 3])$ *et* $ppibis([0, 0, 0, 0, 0, 0])$

Exercice 8. *Ecrire une fonction "sommepos" ayant pour argument l une liste non vide d'entiers et retournant la somme des termes positifs de la liste. Faire afficher sommepos(*$[2, -6, 2, 0, -5, 4]$*).*

Exercice 9. *Ecrire une fonction "termespairs" ayant pour argument une liste l d'entiers et retournant la liste composée des termes pairs de l. Faire afficher termespairs(*$[2, 2, 1, 3, 2, 1, 4]$*).*

Exercice 10. *Ecrire une fonction nommée invlis ayant pour argument une liste l et retournant la liste obtenue en inversant l'ordre des termes de l. Faire afficher* $invlis([-3, 4, 2, 1, 5])$.

Exercice 11. *Pour une liste* $l = [a, b, c, d, ...]$ *d'entiers, on appelle liste des sommes cumulées de l la liste* $[a, a+b, a+b+c, a+b+c+d, ...]$. *Ecrire une fonction "sommecumu" ayant pour argument une liste l d'entiers et retournant la liste des sommes cumulées de l. Faire afficher sommecumu(*$[1, 3, 4, 9]$*).*

Exercice 12. *Les indices de* $l = [2, 3, 1, 3, 3, 6, 7]$ *donnant un terme égal à 3 sont 1,3,4 et forment donc la liste* $[1, 3, 4]$. *Ecrire une fonction "indlis" ayant pour argument une liste l et un objet x et retournant la liste (éventuellement vide) formée des indices de l donnant un terme égal à x. Faire afficher indlis(*$[2, 3, 1, 3, 3, 6, 7]$*,3).*