Listes

PCSI/PTSI

On utilisera pour ce TP l'éditeur en ligne : Future Coder

Notions travaillées :

— Listes, Objet `None`
— Compréhension, implémentation et production d'algorithmes

I Manipulation des listes

Exercice 1. *Création*

1. Affecter à la variable `l0` la liste `[1,3,6]`.
2. Essayer d'accéder à l'indice 3 de la liste `l0`.
3. Affecter à la variable `x` la dernière valeur de `l0`.
4. Affecter à la variable `l1` la liste contenant 144 entiers `12`.
5. Affecter à la variable `l2` la concatenation de `l0` et `l1`.
6. Affecter à la variable `l0` une liste vide.
7. Affecter à la variable `l1` une sous-liste de `l2` contenant les éléments d'indice `1,2,3` et `4`.

Exercice 2. *Modification*

1. Modifier l'indice `0` de `l1` pour être égal à 1.
2. Modifier l'indice `1` de `l1` pour être égal au quotient de la division. euclidienne de la valeur à l'indice `2` par la valeur à l'indice `1`.
3. Enlever le dernière élément de la liste `l1` et l'affecter à la variable `x`.
4. Modifier l'indice `2` de `l1` pour être égal à la longueur de la liste `l1`.
5. Ajouter le reste de la division euclidienne de 40 par `x` à la fin de la liste `l1`.

II Etude d'un algorithme

Algorithme `nadine`

entrées : l une liste d'entiers naturels

```
1.  n ← len(l)
2.  si n = 0 alors
3.      renvoyer None
4.  fin si
5.  m ← l[0]
6.  i ← 1
7.  tant que i < n faire
8.      si m < l[i] alors
9.          m ← l[i]
10.     fin si
11.     i ← i + 1
12. fin tant que
13. renvoyer m
```

On considère dans les exercices suivants l'algorithme écrit en pseudo-code ci dessus.

Exercice 3. *Simulation Papier*

On simule l'algorithme `nadine` sur l'entrée `[3,2,4,1,2,5,0,10,7,8]`.

1. Combien y a-t-il de tours de boucles ? (nombre de fois que l'on exécute le block de la boucle).
2. Quelles sont les valeurs de i et m avant chaque exécution de la ligne 7 ?
3. Donnez une expression de m en fonction de i et l, valide avant chaque exécution de la ligne 7.

Exercice 4. *Spécification de* `nadine`

Décrire la sortie de `nadine` sur une entrée quelconque l, liste d'entiers naturels.

Exercice 5. *Vérification*

Implémenter l'algorithme sur `Future Coder` et faire un appel sur l'entrée `[3,2,4,1,2,5,0,10,7,8]`. A l'aide du bouton `snoop` d'abord, puis du bouton `Python Tutor`, vérifiez les résultats de votre simulation.

III Algorithmique des listes

Exercice 6. *Minimum de liste*

Implémentez une fonction **`min`** qui prend en entrée une liste d'entier et renvoie le minimum de la liste.

Exercice 7. *Somme de liste*

Implémentez une fonction `somme` qui prend en entrée une liste d'entier et renvoie la somme des éléments de la liste.

Exercice 8. *Maximum, minimum, moyenne, variance*

Implémentez quatre fonctions `imax`, `imin`, `moyenne`, et `variance` qui prennent en argument une liste d'entiers et qui renvoient, respectivement, l'indice du maximum, l'indice du minimum, la moyenne et la variance des valeurs de la liste.

Exercice 9. *Recherche Linéaire*

Implémentez une fonction `recherche` qui prend en entrée une liste d'entiers l et un entier x et qui renvoie l'indice dans la liste de la première occurrence de x dans l. Si x n'apparaît pas dans l, la fonction devra renvoyer -1.

Exercice 10. *Somme de deux listes*

Implémentez une fonction `somme` qui prend en entrée deux listes de même longueur $\mathrm{L}_1 = [a_1, \dots, a_n]$ et $\mathrm{L}_2 = [b_1, \dots, b_n]$, qui renvoie la liste $\mathrm{L} = [a_1 + b_1, \dots, a_n + b_n]$ obtenue en sommant un à un les éléments de L_1 et L_2.

Exercice 11. *Liste des diviseurs*

Implémentez une fonction `diviseurs` qui prend en entrée un entier strictement positif `n` et qui renvoie une liste contenant les diviseurs de `n`.

IV Pour s'entraîner

Exercice 12. *Moyenne pondérée*

Implémentez en Python la fonction `moyenne_ponderee` qui prend en entrée une liste de nombres entiers positifs `L` et une liste de coefficient `C` de même longueur et qui renvoie la moyenne de la liste `L` pondérée par les coefficient de C.

Exercice 13. *Éléments impairs*

Implémentez une fonction `impairs` qui prend en entrée une liste d'entiers et qui renvoie une liste contenant les éléments impairs de la liste d'entrée.

Exercice 14. *Liste des multiples*

Implémentez une fonction `multiples` qui prend en entrée deux entiers naturels non nuls n et m et qui renvoie la liste des multiples de `m` plus petits que `n`.

Exercice 15. *Encadrement*

Implémentez en Python la fonction `encadrement` qui prend en entrée une liste de nombres entiers positifs `L` et un nombre entier `x` et qui renvoie le couple `(a,b)` tel que `a` est le plus grand élément de `L` plus petit ou égal à `x` (−1 s'il n'y en a pas) et `b` est le plus petit élément de `L` plus grand que `x` (−1 s'il n'y en a pas).

V Pour aller plus loin

Exercice 16. *Crible d'Eratostene*

Le crible d'Eratostene est un algorithme permettant d'obtenir la liste des nombres premiers plus petit qu'un entier fixé $n \geq 2$. Le crible fonctionne comme ceci :

1. noter tous les nombres entiers de 2 à n
2. prendre le premier nombre qui n'est pas éliminé.
   - il est premier, le noter dans la liste des nombres premiers.
   - éliminer ce nombre et tous ses multiples de la liste initiale.
3. Répéter `2.` tant que tous les nombres de la liste initiale ne sont pas barrés.

Implémentez une fonction `eratostene` qui prend en entrée un entier `n` supérieur à 2 et qui renvoie la liste des nombres premiers plus petits ou égaux à `n` en utilisant le crible d'Eratostene.

Exercice 17. *Tri de liste*

Décrire un algorithme en pseudo-code qui permet de renvoyer une copie triée par ordre croissant d'une liste d'entier.

Vérifiez votre algorithme en le simulant sur des exemples pertinents.

Implémentez l'algorithme en python et réalisez un jeu de test pertinent.

Exercice 18. *Calcul de la Médiane*

Implémentez une fonction `mediane` qui calcule la médiane d'une liste d'entiers.

Si la liste possède un nombre pair d'éléments, la médiane sera le milieu des deux éléments centraux de la liste. *e.g.* la médiane de `[4,3,8,7,5,7]` est le milieu de `5` et `7` soit `6`.