Séance 6b: Exercices sur les listes d'entiers

Université Paris-Diderot

Objectifs:

- Manipuler les listes d'entiers
- Concevoir et programmer des algorithmes sur les listes

Dans cette séance, vous résoudrez des exercices et des problèmes sur des listes d'entiers. Vous écrirez des boucles et définirez des fonctions intermédiaires pour rendre votre code plus lisible et concis.

Exercice 1 (Maximum, ⋆)
Écrire une fonction `maximum` *qui prend en argument une liste d'entiers* `lis` *et qui renvoie l'indice* `i` *du maximum de cette liste.*

Contrat:
`lis=[1000,1,1,1]`
`print(maximum(lis))` *affiche* `0`.

□

Exercice 2 (Signes, ⋆)
Écrire une fonction `samesign` *qui prend en argument deux listes d'entiers* `l1` *et* `l2` *et qui renvoie la liste* `lis` *dont la valeur en la coordonnée* `i` *vaut* `1` *si les deux listes* `l1` *et* `l2` *ont des valeurs de même signe en la coordonnée* `i`, `-1` *sinon. Si les tailles des listes sont différentes la fonction renvoie la liste* `[0]`.

Contrat:
`l1=[1000,1,-1,1], l2=[-1,1,-1,-1000]`
`print(samesign(l1,l2))` *affiche* `[-1,1,1,-1]`.

□

Exercice 3 (Variance, ⋆⋆)
Écrire une fonction `variance` *qui prend en argument une liste d'entiers* `lis` *et qui renvoie la variance de cette liste, c'est-à-dire la moyenne des carrés des écarts à la moyenne.* $var(x_1, x2, \dots x_k) = \frac{1}{k}\sum_{i=1}^{k}(x_i - m)^2$, *où* $m = \frac{1}{k}\sum_{i=1}^{k} x_i$ *est la moyenne*

Contrat:
`lis=[100,1,1,1]`
`print(variance(lis))` *affiche* `1837.6875`.

□

Exercice 4 (Décalage, ⋆⋆)
Écrire une fonction `shift` *qui prend en argument une liste d'entiers* `lis` *et qui renvoie une copie de cette liste où toutes les valeurs sont décalées d'une case vers la droite (et la dernière valeur se retrouve en position 0).*

Contrat:
`lis=[1000,1,2,3]`
`print(shift(lis))` *affiche* `[3,1000,1,2]`.

Même chose mais avec un décalage de `n` *cases (*`n` *est un paramètre). Que faire quand* `n` *est négatif ?* □

Exercice 5 (Pairs et Impairs, ⋆⋆ – ⋆⋆⋆)
Écrire une fonction `paritysort` *qui prend en argument une liste d'entiers* `lis` *et qui renvoie une liste contenant les valeurs de* `lis` *et tel que tous les nombres pairs se trouvent à gauche des nombres impairs.*

Contrat:
`lis=[1,2,3,4,5,6,8,10]`
`print(paritysort(lis))` *affiche (par exemple)* `[2,4,6,8,10,1,3,5]`.

(bonus) Modifier la fonction pour que les valeurs ayant la même parité soient triées par ordre croissant.

Contrat:
`lis=[5,8,4,2,3,1,10,9,7]`
`print(paritysort(lis))` *affiche* `[2,4,8,10,1,3,5,7,9]`.

□

Exercice 6 (Fibonacci (bis), ⋆⋆)
La suite de Fibonacci $(F_n)_{n\geq 1}$ *est définie par* $F_0 = 0$, $F_1 = 1$ *et* $F_{n+2} = F_{n+1} + F_n$. *Écrire une fonction* `fibonacci` *qui renvoie une liste contenant les* n *premiers termes de la suite.*

Contrat:
`print(fibonacci(5))` *affiche* `[0,1,1,2,3]`.

□

Exercice 7 (Crible d'Ératosthène, ⋆⋆⋆)
Le crible d'Eratosthène est un moyen de trouver tous les nombres premiers plus petits qu'un entier donné `n`. *Pour cela on construit la liste des nombres de 2 à n, puis, en partant de 2, on supprime tous les multiples (propres) des éléments de la liste. Ainsi, on commence par retirer de la liste tous les nombres multiples de 2 strictement plus grands que 2. Le nombre suivant dans la liste est alors 3, on supprime donc tous les multiples de 3 strictement plus grands que 3, et ainsi de suite avec le prochain élément de la liste, 5. À la fin, il ne reste dans la liste que les nombres premiers plus petits que* n.

Écrire une fonction `crible` *qui prend un paramètre un entier* `n` *et qui renvoie la liste* `[2,3,..., n]` *à laquelle on a appliqué le crible d'Ératosthène.*

Contrat:
`print(crible(10))` *affiche* `[2,3,0,5,0,7,0,0,0]`.

□