Recherche d'extremum

Résumé

Trouver la plus grande note d'une classe, la température la plus basse de la semaine ou le mot le plus long d'un texte : ces problèmes ont tous la même structure. On dispose d'une collection de valeurs et on cherche celle qui est la plus grande (ou la plus petite). C'est le problème de la **recherche d'extremum**. C'est aussi l'un des premiers algorithmes que l'on apprend à écrire, et il illustre une idée fondamentale : parcourir une liste en gardant en mémoire la meilleure valeur rencontrée.

Prérequis : Les listes, Les boucles.

① Mémo

Recherche du maximum

Principe Parcourir toute la liste en conservant le plus grand élément rencontré jusqu'ici.
Initialisation Toujours avec `L[0]`, jamais avec `0` (faux pour une liste de négatifs).
Complexité $O(n)$: un seul parcours, une comparaison par élément.

Recherche du minimum

La même idée, en remplaçant la comparaison > par <.
On peut aussi chercher l'**indice** de l'extremum en retenant `i_max` (ou `i_min`) au lieu de la valeur elle-même.

Pièges fréquents

- Initialiser avec `maxi = 0` au lieu de `maxi = L[0]` : échoue si la liste ne contient que des négatifs.
- Écrire `maxi = L[i]` au lieu de mettre à jour conditionnellement.
- Oublier que la fonction doit aussi fonctionner sur une liste à un seul élément.

Erreurs classiques

Code erroné	Code correct	Explication
`maxi = 0`	`maxi = L[0]`	Initialiser à 0 donne un résultat faux si la liste ne contient que des négatifs : le maximum serait 0 alors qu'il n'est pas dans la liste.
`if L[i] > maxi:` `return L[i]`	`if L[i] > maxi:` `maxi = L[i]`	Renvoyer dès la première valeur supérieure interrompt le parcours : on n'explore pas le reste de la liste.
Confondre indice et valeur : `maxi = i` au lieu de `maxi = L[i]`	`i_max = i` pour l'indice, `maxi = L[i]` pour la valeur	Il faut être clair sur ce qu'on retient : l'indice du maximum ou la valeur du maximum. Les deux algorithmes sont différents.
`for x in L[1:]` : sans initialiser à `L[0]`	`maxi = L[0]` puis `for x in L[1:]:`	Si l'on commence le parcours à l'indice 1, on doit initialiser avec `L[0]` ; sinon le premier élément n'est jamais comparé.

② Exemples

Programme 1 · Recherche du maximum

```python
def maximum(L):
    maxi = L[0]
    for x in L:
        if x > maxi:
            maxi = x
    return maxi

print(f"max = {maximum([3, -1, 7, 2])}")
# max = 7
```

Programme 2 · Maximum avec trace

```
def maximum_trace(L):
    maxi = L[0]
    print(f"init : maxi = {maxi}")
    for x in L[1:]:
        if x > maxi:
            print(f"  {x} > {maxi} -> maxi = {x}")
            maxi = x
        else:
            print(f"  {x} <= {maxi} -> pas de changement")
    return maxi

maximum_trace([12.5, 14.0, 11.8, 15.3, 16.1, 14.5])
```

Programme 3 · Indice du maximum

```
def indice_maximum(L):
    i_max = 0
    for i in range(1, len(L)):
        if L[i] > L[i_max]:
            i_max = i
    return i_max

jours = ["lun", "mar", "mer", "jeu", "ven", "sam", "dim"]
T = [12.5, 14.0, 11.8, 15.3, 13.7, 16.1, 14.5]
i = indice_maximum(T)
print(f"Jour le plus chaud : {jours[i]} ({T[i]} °C)")
```

③ Exercices

Exercice 1 *Minimum et indice*

1. Écrire une fonction `minimum(L)` qui renvoie le plus petit élément.
2. Écrire une fonction `indice_minimum(L)` qui renvoie l'indice du plus petit élément (le premier s'il y a plusieurs minimums).

Exercice 2 *Les deux extremums en un parcours* Écrire une fonction `extremums(L)` qui renvoie le tuple `(minimum, maximum)` en effectuant **un seul parcours** de la liste (une seule boucle `for`).

Exercice 3 *Amplitude thermique* On donne une liste de relevés de température sur une journée (un relevé par heure). Écrire une fonction `amplitude(T)` qui renvoie l'amplitude thermique (écart entre le maximum et le minimum) en un seul parcours.

Tester avec la liste suivante :

Programme 4 · Relevés de la journée

```
T = [14.2, 13.8, 13.1, 12.7, 12.5, 13.0, 14.8, 17.3, 19.1, 20.5, 21.2,
21.8,
  22.1, 21.6, 20.9, 19.5, 18.0, 16.8, 15.5, 14.9, 14.3, 14.0, 13.7,
13.4]
```

Exercice 4 *Les deux plus grands éléments* Écrire une fonction `deux_plus_grands(L)` qui renvoie les deux plus grands éléments d'une liste de taille $\geqslant 2$ en un seul parcours (sans trier la liste). Renvoyer un tuple `(max1, max2)` avec `max1 >= max2`.

Solutions des exercices

Corrigé de l'exercice 1.

```
def minimum(L):
    mini = L[0]
    for x in L:
        if x < mini:
            mini = x
    return mini

def indice_minimum(L):
    i_min = 0
    for i in range(1, len(L)):
        if L[i] < L[i_min]:
            i_min = i
    return i_min
```

Vérification : `minimum([-3, -7, -1, -5])` renvoie `-7`; `indice_minimum([3, -1, 7, 2])` renvoie `1`.

Corrigé de l'exercice 2.

```
def extremums(L):
    mini = L[0]
    maxi = L[0]
    for x in L:
        if x < mini:
            mini = x
        if x > maxi:
            maxi = x
    return mini, maxi
```

Vérification : `extremums([3, 7, 2, 9, 1])` renvoie `(1, 9)`.

Remarque : avec l'initialisation `mini = maxi = L[0]`, `elif` fonctionnerait aussi ; les deux `if` indépendants restent indispensables si l'on initialise autrement (par exemple `mini = float('inf')` et `maxi = float('-inf')`), et ne coûtent rien ici.

Corrigé de l'exercice 3.

```python
def amplitude(T):
    mini = T[0]
    maxi = T[0]
    for t in T:
        if t < mini:
            mini = t
        if t > maxi:
            maxi = t
    return maxi - mini
```

Résultat : `amplitude(T)` = 22.1 − 12.5 = 9.6 °C.

C'est une application directe de `extremums` suivie d'une soustraction.

Corrigé de l'exercice 4.

```python
def deux_plus_grands(L):
    if L[0] >= L[1]:
        max1, max2 = L[0], L[1]
    else:
        max1, max2 = L[1], L[0]
    for i in range(2, len(L)):
        if L[i] >= max1:
            max2 = max1
            max1 = L[i]
        elif L[i] > max2:
            max2 = L[i]
    return max1, max2
```

Principe : on maintient deux variables. Quand un élément dépasse `max1`, l'ancien `max1` descend dans `max2`. Quand il est entre les deux, il remplace seulement `max2`.

Vérification : `deux_plus_grands([3, 7, 2, 9, 1])` renvoie `(9, 7)`; `deux_plus_grands([5, 5])` renvoie `(5, 5)`.