Algo 3 - Parcours séquentiel d'un tableau

I – Importance de l'algorithmique

ALGORITHME : un **algorithme** est un ensemble **fini** (au sens "pas infini") d'instructions précises permettant de résoudre une classe de problèmes. L'algorithme doit être écrit de façon suffisamment clair pour lever toute ambiguïté, et être transposable facilement dans un langage de programmation.

ENTREE(S) → Algorithme → SORTIE

On implémente donc les algorithmes sous forme de fonctions.

L'**ALGORITHMIQUE** consiste à étudier des problèmes en vue de les résoudre à l'aide d'un algorithme.
On cherche donc :

- à vérifier qu'il s'arrête bien sur n'importe quelle entrée correcte fournie (avec une **preuve de terminaison**)
- à vérifier que la réponse éventuelle soit juste (avec une **preuve de correction**)
- à estimer la rapidité d'exécution de l'algorithme (son **coût** / sa **complexité**)

On notera qu'un algorithme performant tournant sur une machine ancienne est souvent préférable à un algorithme naïf tournant sur une machine récente.

II – Algorithme de recherche linéaire (linear search)

But : trouver la position de la **première occurrence** de l'élément x recherché dans un tableau t.

Principe :

1. On commence par la case d'indice 0
2. TANT QUE la case i existe ET ne contient pas le contenu cherché : passe à la case suivante.
3. SI la valeur finale de l'indice est valide : la réponse est bien l'indice. SINON, la réponse est -1/VIDE.
4. On RENVOIE la réponse

ENTREES : **t** : un tableau **x** : l'élément recherché
SORTIE : un entier correspondant à l'indice ou -1.

ALGORITHME (en 4 temps)

```
i ← 0

TANT QUE i < longueur ET que t[i] est différent de x
    i ← i + 1
Fin TANT QUE

SI i < longueur du tableau
    reponse ← i
SINON
    reponse ← -1
Fin Si
Renvoyer reponse
```

Ex 01 Utiliser les instructions suivantes pour revoir comment fonctionne un tableau :

```
>>> t = [5, 10, 7]
>>> len(t)          >>> t[0]          >>> t[1]          >>> t[2]          >>> t[3]
```

Ex 02 Quel est le seul cas où a ET b répond VRAI ?

Ex 03 Trouver à quelle phrase correspond le mieux la condition **i < longueur du tableau** puis **t[i] est différent de x.**

Ex 04 : Ecrire le déroulé de l'algorithme s'il travaille sur le tableau ci-dessous en recherchant 1024. Voir le début de la rédaction sur le site. Quelle est la condition qui provoque l'arrêt du TANT QUE ?

Index	**0**	1	**2**	3	**4**	5	**6**
Elément	**13**	18	**89**	1024	**45**	-12	**18**

Ex 05 : Appliquer l'algorithme à la main avec une recherche sur 1025. Quelle est la condition qui provoque l'arrêt ?

Ex 06 : 18 apparaît deux fois dans le tableau. La valeur renvoyée est : A : **0** B : **1** C : **6** D : **(1,6)**

Ex 07-11 : Implémenter cet algorithme en Python.

Ex 12 : Réaliser l'implémentation avec **for+return**.

Ex 13 : Quelle est la meilleure version d'un point de vue algorithmique ? Programmation ?

```
L01 def rechercher(t:list, x:'Element') -> int:
L02     """Renvoie l'indice de la première occurrence de x dans t, -1 sinon"""
L03
L04     # version while
L05     i = 0
L06     while i < len(t) and t[i] != x:
L07         i = i + 1
L08
L09     if i < len(t):
L09         reponse = i
L10     else:
L11         reponse = -1
L13     return reponse
```

```
L04 # version for+return
L05 for i in range( len(t) ):
L06     if t[i] == x:
L07         return i
L08
L09 return -1
```