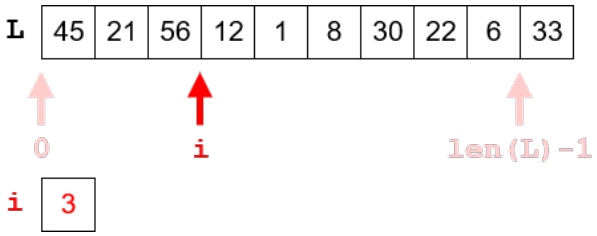
Parcours séquentiel d'un tableau

I. Les tableaux en algorithmique

I. 1. Structure d'un tableau

Un tableau est une structure de données représentant une séquence finie d'éléments auxquels on peut accéder par leur position (appelé aussi indice ou index) dans la séquence. On parle aussi de tableau indexé.



On ne s'intéresse qu'aux tableaux présentant des données du même type.

Dans le langage Python, les tableaux sont implémentés par des listes.

Exemple 1 (Quelques exemples de tableaux en Python)

```
liste_entiers = [1, 2, 3, 4, 5]
liste_reels = [0.0, 1.0, 2.0, 3.5]
lliste_caracteres = ["toi", "moi", "nous"]
```

I. 2. Parcours séquentiel d'un tableau

Déf. 1 Le parcours séquentiel d'un tableau consiste à faire évoluer un indice, et ainsi à déplacer un pointeur sur chaque case du tableau. La première valeur de l'indice est 0 (le début du tableau). La dernière valeur de l'indice est (longueur - 1).

Dans un pseudo-code, on écrit :

Algorithme : parcours(tableau)

```
/* algorithme qui renvoie le contenu d'un tableau                    */
Entrées : tableau : un tableau
début
    pour indice variant de 0 à longueur exclue faire
        ...
    fin pour
fin
```

Dans la langage Python, le parcours d'un tableau (d'une liste) s'implémente par :

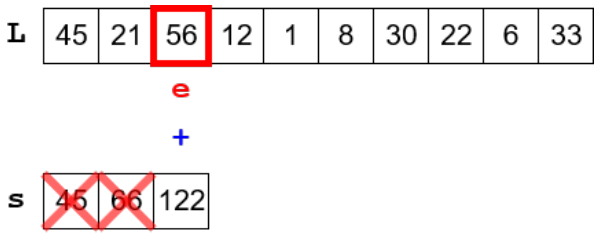
```
for i in range(0, len(liste), 1):
    ............
```

ou bien par :

```
for element in liste:
    ............
```

II. Moyenne des éléments d'un tableau

Pour calculer la moyenne des valeurs d'une liste, il faut calculer la somme de ces valeurs et diviser par le nombre de valeurs contenues dans la liste.
Pour calculer la somme on initialise une variable *somme* à zéro, et on itère sur l'ensemble des éléments de la liste. A chaque itération, on ajoute à *somme* la valeur de *element* .



1. Quelle est la moyenne du tableau proposé ci-dessus ?
2. Proposez un algorithme en pseudo-code permettant de calculer la moyenne des éléments d'un tableau.

Algorithme : moyenne(tableau)

```
/* Algorithme qui renvoie la moyenne des éléments d'un tableau                */
```
Entrées : *tableau* : tableau contenant des nombres
Sorties : *moyenne* : moyenne des éléments du tableau

```
début
    ..................;
    pour ..................................................... faire
        ..................
    fin pour
    ..........................................................;
    retourner moyenne
fin
```

3. Réalisez l'historique d'exécution pour les tableaux ci-dessous :

45	21

45	21	56

45	21	56	12

45	21	56	12	1

45	21	56	12	1	8

4. Pour ces différents historiques d'exécution, comptez le nombre d'instructions successives. Complétez alors le tableau suivant :

Longueur du tableau (n)				
Nombre d'instructions exécutées (C)				

5. Représentez le nombre d'instructions en fonction de la longueur du tableau. Quelle est la forme du graphique obtenu ?

   Donnez l'équation mathématique reliant C à n :

6. Combien d'instructions seraient exécutées pour un tableau contenant :
   (a) 100 éléments ?
   (b) 1 000 éléments ?
   (c) 1 000 000 éléments ?
7. On dit que le coût de cet algorithme est linéaire. Justifiez cette expression.

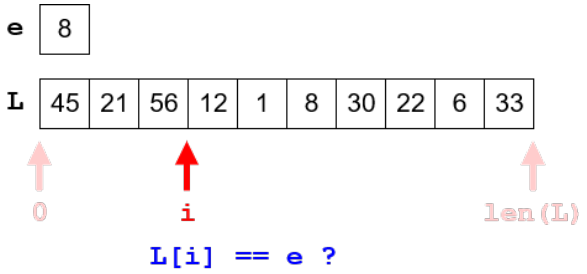
8. Proposez une implémentation en Python de cet algorithme.

```
def moyenne(liste):
    """fonction renvoyant la moyenne des éléments d'une liste"""
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    ..........................................................
    return moyenne
```

III. Recherche d'un élément dans un tableau

Il s'agit de déterminer la position (l'indice) d'un élément x prétendument présent dans un tableau.

Un "pointeur" i, initialisé à zéro, permet d'accéder aux éléments du tableau. À chaque itération, on compare l'élément d'indice i du tableau avec l'élément x recherché. S'ils sont égaux, on renvoie la valeur de l'indice et les itérations cessent. Dans le cas contraire, on incrémente i et on fait une nouvelle itération. Si l'élément x n'est pas dans le tableau, le pointeur va jusqu'au bout de la liste, et finit par prendre la valeur de la longueur de la liste. Dans ce cas, la valeur retournée est -1.



1. Dans l'exemple proposé ci-dessus, quelle sera la valeur renvoyée ?
2. Proposez un algorithme en pseudo-code (sur le même modèle que le précédent) permettant de rechercher l'élément x dans un tableau.
3. On souhaite réaliser les recherches ci-dessous :
   On recherche 56 dans | 45 | 21 | 56 |
   On recherche 12 dans | 45 | 21 | 56 | 12 |

On recherche 1 dans

45	21	56	12	1

On recherche 8 dans

45	21	56	12	1	8

En reproduisant l'exemple précédent (moyenne des éléments d'un tableau), indiquez si le coût de l'algorithme est linéaire ou non-linéaire.

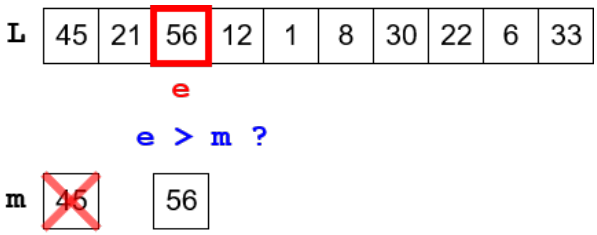
4. Proposez une implémentation en Python de l'algorithme.

```
def recherche(x, liste):
    """fonction renvoyant la position de x dans la liste s'il est présent
    et - 1 sinon."""
    ..........................................................
```

IV. Recherche du maximum dans un tableau

Il s'agit de déterminer l'élément de plus grande valeur présent dans un tableau.

Pour connaître la valeur maximum des valeurs contenues dans un tableau, il faut évaluer tour à tour tous les éléments du tableau. La valeur maximum est mémorisée dans une variable *maximum*, initialisée avec le premier élément du tableau. 'A chaque étape, on compare l'élément du tableau au contenu de la variable *maximum*. Si l'élément est supérieur à *maximum*, *maximum* prend alors la valeur de cet élément.



1. Dans l'exemple proposé ci-dessus, quelle sera la valeur renvoyée ?

2. On initialise la variable *maximum* avec la première valeur du tableau. Justifier cette valeur.

3. Proposez un algorithme en pseudo-code (sur le même modèle que le précédent) permettant de rechercher l'élément de plus grande valeur dans un tableau.

4. Con souhaite connaître le maximum de chacun des tableaux ci-dessous :

45	21	56

45	21	12	56

45	21	12	1	56

45	21	12	1	8	56

En reproduitant l'exemple précédent, indiquez si le coût de l'algorithme est linéaire ou non-linéaire.

5. Proposez une implémentation en Python de cet algorithme.

```
def maximum(liste):
    """fonction renvoyant le maximum de la liste."""
    ..........................................................
```

V. Exercices

Exercice 1.
Écrire une fonction *separer* permettant, à partir d'une liste de nombres, d'obtenir deux listes. La première comporte les nombres inférieurs ou égaux à un nombre donné, la seconde les nombres qui lui sont strictement supérieurs.
`separer([45, 21, 56 ,12, 1, 8, 30, 22, 6, 33], 30)` doit renvoyer :
[21, 12, 1, 8, 30, 22, 6], [45, 56, 33]

Exercice 2.
Écrire une fonction *plus_proche* permettant de rechercher la plus proche valeur d'un nombre dans une liste. `plus_proche([45, 21, 56 ,12, 1, 8, 30, 22, 6, 33], 20)` doit renvoyer 21

Exercice 3.
Écrire une fonction *compter_position* permettant de compter le nombre d'occurrences d'une lettre dans une chaîne de caractères et de donner leurs positions.
`compter_position("Numérique et Sciences Informatiques !", 'm')` doit renvoyer `2, [2, 27]`.

Exercice 4.
Écrire une fonction *compter_tout* permettant d'obtenir les nombres d'occurrences de toutes les lettres d'une chaîne de caractères, sous la forme d'un dictionnaire {lettre : nbre occurences}.
`compter_tout("Numérique et Sciences Informatiques !")` doit renvoyer : {'N' : 1, 'u' : 3, 'm' : 2, 'é' : 1, 'r' : 2, 'i' : 3, 'q' : 2, 'e' : 5, ' ' : 4, 't' : 2, 'S' : 1, 'c' : 2, 'n' : 2, 's' : 2, 'I' : 1, 'f' : 1, 'o' : 1, 'a' : 1, '!' : 1}