

## Recherche d'un élément dans un tableau

### Activité 1

Ecrire une fonction recherche qui prend en argument un tableau tab et un élément elem. La fonction doit renvoyer l'indice de elem s'il est présent dans le tableau et -1 sinon.

Exemples :

```
>>> recherche([1, 2, 3], 3)
2
>>> recherche([1, 2, 3], 4)
-1
```

```
[ ] : # compléter le bloc de code

def recherche(tab, elem):
    """
    Exemples et tests:
    >>> recherche([1, 2, 3], 3)
    2
    >>> recherche([1, 2, 3], 4)
    -1
    """
    ...
    # écrire quelques tests
...

```

Indice Voici une idée d'algorithme :

- Parcourir tout le tableau de taille n, de l'indice 0 à l'indice n-1.
- Pour le tour de boucle i, si la valeur courante du tableau est égale à elem, alors provoquer un renvoie *anticipé* qui arrête la fonction.
- Lorsque la boucle se termine, c'est que elem n'a pas été trouvé.

1

## Activité bonus 2

Compléter le code ci-dessous pour afficher le temps d'exécution de la fonction recherche\_2.

```
[ ] : # compléter le bloc de code
...

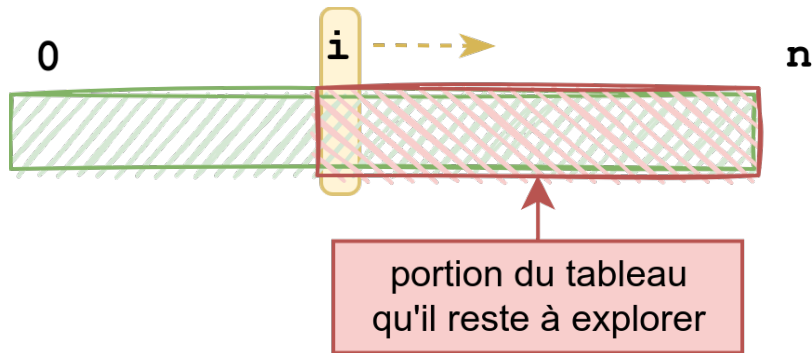
```

nombre de case	durée (s)
500 000	
1 000 000	
2 000 000	
4 000 000	
8 000 000	
16 000 000	
32 000 000	
64 000 000	

## Activité bonus 3

D'après toi, pourquoi les durées sont-elles si différentes ?

4



### Activité 2

Si ta fonction semble correcte, teste-la sur Moodle et **valide** l'activité.



[02. Recherche dans un tableau](#)

### Activité 3

D'après toi, qu'est ce qu'un fonction *efficace*? De quoi dépend l'efficacité de la fonction recherche?

### Activité 4

Compléter le code ci-dessous afin de déterminer le temps qu'il faut pour explorer un tableau de 500 000 cases.



```
[ ]: # compléter les '...' et ajouter des commentaires expliquant ce que fait la ligne d'instruction

from time import time
tab = [1] * ... # ...

t_0 = time()
recherche(tab, ...) # ...
t_1 = time()

print(t_1 - t_0) # ...
```

Compléter le tableau ci-dessous :

nombre de case	durée (s)
500 000	
1 000 000	
2 000 000	
4 000 000	
8 000 000	
16 000 000	
32 000 000	
64 000 000	

### Pour aller plus loin...

#### Activité bonus 1

Refaire l'activité 1, mais avec une boucle while.

```
[ ]: def recherche_2(tab, elem):
    """
    Renvoie l'indice de elem s'il est présent dans tab et -1 sinon
    Fonction codée avec une boucle `while`

    Exemples:
    >>> recherche_2([1, 2, 3], 3)
    2
    >>> recherche_2([1, 2, 3], 4)
    -1
    """
    ...

    # écrire quelques tests
    ...
```