NOM : ..... Prénom : .....

**CPGE PSI* 2026/2027**
**Lycée La Fayette**

**Informatique**
**Nathalie Planche**

# Devoir surveillé n°1 - Test de rentrée - Jeudi 3 septembre 2026

## Version n°1

**Exercice 1 :** Ecrire une fonction `maximum(L)` qui renvoie **un couple** constitué de la valeur de l'élément maximum de la liste `L`, et de la position de ce maximum dans la liste (si ce maximum est atteint plusieurs fois, l'indice renvoyé sera le plus petit).

```
def maximum(L):
    maxi = L[0]          # initialisation du maximum
    indice = 0           # initialisation de l'indice où il est
    for i in range(len(L)):
        if L[i] > maxi:  # cas où il faut changer
            maxi = L[i]
            indice = i
    return maxi, indice  # on retourne un tuple
```

**Exercice 2 :** Ecrire une fonction `cherche(texte,mot)` qui à partir d'une chaine de caractère `texte` et d'une chaine de caractère `mot` , détermine si le mot fait partie du texte. Cette fonction devra renvoyer :

- l'indice de la première occurrence du mot dans le texte s'il en fait partie
- `False` sinon

*Exemple :* `cherche('bonjour les PSI','jour')` doit renvoyer 3.

```
def cherche(texte,mot):
    for k in range(len(texte)-len(mot)+1):
        if mot==texte[k:k+len(mot)]:
            return k
    return False
```

ou

```
def cherche(texte,mot):
    for k in range(len(texte)-len(mot)+1):
        cpt=0
        for i in range(len(mot)):
            if mot[i]==texte[k+i]:
                cpt+=1
        if cpt==len(mot):
            return k
    return False
```

**Exercice 3 :** 1. Compléter le script suivant pour que la fonction trie la liste `L` selon les valeurs croissantes, en suivant le principe du **tri à bulles**.

On rappelle que le principe de ce tri consiste, par parcours successifs, à faire remonter les plus grands éléments de la liste à la fin de cette liste. (au premier parcours, par échange successifs, le plus grand élément se retrouvera à la fin de la liste)

*Remarque : cette fonction ne retourne rien, quand on l'exécute elle modifie la liste passée en paramètre.*

```
1  def tri_bulles (L):
2      n=len(L)
3      for j in range (n-1)
4          for i in range (n-j-1):
5              if L[i]>L[i+1]:
6                  L[i],L[i+1] = échange les éléments L[i+1],L[i]
7      return L
```

2. Donner, en justifiant, la complexité en temps dans le pire des cas, en fonction de la taille $n$ de la liste `L`.

• la boucle externe s'exécute $(n-1)$ fois
• pour chaque itération $j$, la boucle interne s'exécute $(n-j-1)$ fois
Le reste est en temps constant, donc l'ordre de grandeur de la complexité est

$$\sum_{j=0}^{n-1}(n-j-1)=\sum_{k=0}^{n-1}k=\frac{(n-1)n}{2}=O(n^2)$$

complexité quadratique

**Exercice 4 : A faire uniquement si les autres exercices sont finis**

Programmer l'algotihme de recherche dichotomique dans un tableau trié. La fonction devra renvoyer la position de l'élément s'il est présent, et `False` sinon.

Savez-vous justifier la complexité de cet algorithme ?

```
def dicho(L,elt):
    d,f=0,len(L)-1
    while d<=f:
        milieu=(d+f)//2
        if L[milieu]==elt:
            return milieu
        if L[milieu]<elt:
            d=milieu+1
        if L[milieu]>elt:
            f=milieu-1
    return False
```

Complexité :

Si la liste est de longueur $n=2^p$, alors la boucle s'exécute p fois dans le pire des cas (car la longueur (f-d) est divisée par 2 à chaque tour !)

Donc complexité en $O(p)$

Or $n=2^p \Leftrightarrow p=\log_2(n)$

Donc

$$C(n)=O(\log_2(n))$$

complexité logarithmique