

TP 02

Recherche séquentielle

Savoirs et compétences :

- Th. 1 : Recherche séquentielle dans un tableau unidimensionnel. Dictionnaire.

Proposition de corrigé

Activité 1 – Autour des listes

Question 1 Écrire une fonction de signature `generer_liste_entiers_01(n: int) -> list` renvoyant la liste des entiers compris entre 0 (inclus) et n (exclu). On utilisera une boucle for.

```
def cor_generer_liste_entiers_01(n):  
    L=[]  
    for i in range(n):  
        L.append(i)  
    return(L)
```

Question 2 Écrire une fonction de signature `generer_liste_entiers_02(n: int) -> list` renvoyant la liste des entiers compris entre 0 (inclus) et n (exclu). On utilisera une boucle while.

```
def cor_generer_liste_entiers_02(n):  
    L=[]  
    i=0  
    while i < n:  
        L.append(i)  
        i+=1  
    return(L)
```

Question 3 Écrire une fonction de signature `generer_liste_entiers_03(deb: int, fin:int) -> list` renvoyant la liste des entiers compris entre deb (inclus) et fin (inclus). On utilisera une boucle for.

```
def cor_generer_liste_entiers_03(deb,fin):  
    L=[]  
    for i in range(deb,fin+1):  
        L.append(i)  
    return L
```

Question 4 Écrire une fonction de signature `generer_liste_entiers_04(deb: int, fin:int) -> list` renvoyant la liste des entiers compris entre deb (inclus) et fin (inclus). On utilisera une boucle while.

```
def cor_generer_liste_entiers_04(deb,fin):  
    L=[]  
    i = deb  
    while i <= fin :  
        L.append(i)  
        i=i+1  
    return L
```

Question 5 Écrire une fonction de signature `generer_liste_pairs(n:int) -> list` renvoyant la liste des entiers pairs compris entre 0 (inclus) et n (exclu).

```
def cor_generer_liste_pairs(n):
    L = []
    for i in range(0,n,2):
        L.append(i)
    return L
```

Question 6 Écrire une fonction de signature `generer_liste_impairs(n:int) -> list` renvoyant la liste des entiers impairs compris entre 0 (inclus) et n (exclu).

```
def cor_generer_liste_impairs(n):
    L = []
    for i in range(1,n,2):
        L.append(i)
    return L
```

Question 7 Écrire une fonction de signature `is_multiple(n:int, m:int) -> bool` renvoyant True si n est multiple de m.

```
def cor_is_multiple(n,m):
    return n%m == 0
```

Question 8 En utilisant la fonction précédente, écrire une fonction de signature `genere_liste_multiple(n:int, m:int) -> list` renvoyant les n premiers multiples de m.

```
def cor_genere_liste_multiple(n,m):
    L = []
    i = 1
    while len(L)<n :
        if cor_is_multiple(i,m) :
            L.append(i)
        i = i+1
    return L
```

Recherche d'un nombre dans une liste

Nous allons rechercher si un nombre est dans une liste.

Commençons par générer une liste de nombre aléatoires. Pour cela recopier les lignes suivantes.

```
import random as rd
def generer_alea(nb: int) -> list :
    """
    Génération d'une liste nb de nombres aléatoires compris entre 0 inclus et nb exclus.
    """
    res = []
    for i in range(nb):
        res.append(rd.randrange(0,nb))
    return res
```

Question 9 Écrire une fonction de signature `recherche_nb_01(nb: int, L: list) -> bool` qui renvoie True si nb est dans L, False sinon. On utilisera une boucle for.

```
def cor_recherche_nb_01(nb:int,L:list):
    test=False
    for x in L:
        if nb==x:
            test=True
    return(test)
```

Question 10 Écrire une fonction de signature `recherche_nb_02(nb: int, L: list) -> bool` qui renvoie True si nb est dans L, False sinon. On utilisera une boucle while.

```
def cor_recherche_nb_02(nb:int,L:list):
    n=len(L)
    i=0
    while i<=n-1 and nb!=L[i]:
        i=i+1
    return(i<n)
```

Question 11 Écrire une fonction de signature `recherche_nb_03(nb: int, L: list) -> bool` qui renvoie `True` si `nb` est dans `L`, `False` sinon. On n'utilisera pas explicitement de boucles `for` ou `while`.

```
def cor_recherche_nb_03(nb:int,L:list):
    return(nb in L)
```

Question 12 Écrire une fonction de signature `recherche_first_index_nb_01(nb: int, L: list) -> int` qui renvoie l'index de la première apparition du nombre `nb` dans la liste `L`. La fonction renverra `-1` si `nb` n'est pas dans la liste. On utilisera une boucle `for`.

```
def cor_recherche_first_index_nb_01(nb:int, L:list):
    n=len(L)
    index=-1
    for i in range(n):
        if nb==L[i] and index==-1:
            index=i
    return(index)
```

Question 13 Écrire une fonction de signature `recherche_first_index_nb_02(nb: int, L: list) -> int` qui renvoie l'index de la première apparition du nombre `nb` dans la liste `L`. La fonction renverra `-1` si `nb` n'est pas dans la liste. On utilisera une boucle `while`.

```
def cor_recherche_first_index_nb_02(nb:int, L:list):
    n=len(L)
    i=0
    while i<=n-1 and nb!=L[i]:
        i=i+1
    if i<n:
        index=i
    else:
        index=-1
    return(index)
```

Question 14 Écrire une fonction de signature `recherche_last_index_nb_01(nb: int, L: list) -> int` qui renvoie l'index de la dernière apparition du nombre `nb` dans la liste `L`. La fonction renverra `-1` si `nb` n'est pas dans la liste.

```
def cor_recherche_last_index_nb_01(nb:int, L:list):
    n=len(L)
    i=0
    while i<=n-1 and nb!=L[i]:
        i=i+1
    if i<n:
        index=i
    else:
        index=-1
    return(index)
```

Question 15 Écrire une fonction de signature `recherche_index_nb_01(nb: int, L: list) -> list` qui renvoie la liste des index du nombre `nb` dans la liste `L`. La fonction renverra une liste vide si `nb` n'est pas dans la liste.

```
def cor_recherche_index_nb_01(nb:int, L:list):
    n=len(L)
    i=0
```

```
while i<=n-1 and nb!=L[i]:
    i=i+1
if i<n:
    index=i
else:
    index=-1
return(index)
```

Recherche d'un caractère dans une chaîne (de caractères)

Question 16 Écrire une fonction de signature `is_char_in_str_01(lettre: str, mot: str) -> int` qui renvoie True si lettre est dans mot, False sinon. On utilisera une boucle for ou while.

```
def cor_is_char_in_str_01(lettre:str,mot:str):
    n=len(mot)
    i=0
    while i<=n-1 and lettre!=mot[i]:
        i=i+1
    return(i<n)
```

Question 17 Écrire une fonction de signature `is_char_in_str_02(lettre: str, mot: str) -> int` qui renvoie True si lettre est dans mot, False sinon. On n'utilisera ni boucle for ni while explicite.

```
def cor_is_char_in_str_02(lettre:str,mot:str):
    return(lettre in mot)
```

Question 18 Écrire une fonction de signature `compte_lettre_01(lettre: str, mot: str) -> int` qui renvoie le nombre d'occurrences de lettre dans le mot.

```
def cor_compte_lettre_01(lettre:str, mot:str):
    nb=0
    for x in mot:
        if x==lettre:
            nb=nb+1
    return(nb)
```

Question 19 Écrire une fonction de signature `compte_lettre_02(lettre: str, mots: list) -> int` qui renvoie le nombre d'occurrences de lettre dans une liste de mots mots.

```
def cor_compte_lettre_02(lettre:str, mots:list):
    nb=0
    for mot in mots:
        if lettre in mot:
            nb=nb+1
    return(nb)
```

Question 20 Quelle consonne apparaît le plus souvent? Quelle consonne apparaît le moins souvent? Indiquer le nombre d'occurrences dans chacun des mots

Question 21 Écrire une fonction de signature `mots_plus_long(mots: list) -> str` qui renvoie le mot le plus long.

```
def cor_mots_plus_long(mots:list):
    M=0
    for mot in mots:
        if len(mot)>M:
            max=mot
            M=len(mot)
    return(max)
```

Question 22 Écrire une fonction `init_dictionnaire(chaine:str) -> dict` que crée, initialise et renvoie le dictionnaire dont les clés sont les lettres de l'alphabet et les valeurs sont initialisées à 0.

```
def cor_init_dictionnaire(chaine:str) -> dict :
    alphabet = 'abcdefghijklmnopqrstuvwxyz'
    dico = {}
    for lettre in alphabet :
        dico[lettre]=0
    return dico
```

Question 23 Écrire une fonction `remplir_dictionnaire(dico:dict, liste_mots:list) -> dict` qui compte le nombre de lettres de chacun des mots de la liste `liste_mots` chargée précédemment en incrémentant chacune des valeurs du dictionnaire.

```
def cor_remplir_dictionnaire(dico,mots):
    for mot in mots :
        for lettre in mot :
            dico[lettre]=dico[lettre]+1
    return dico
```

Question 24 Écrire une fonction `cherche_podium(dico:dict) ->list` qui renvoie la liste des trois lettres les plus utilisées. La liste renvoyée sera sous la forme `['a' ,5] , ['b' ,4] , ['c' ,3]` .

```
def cor_cherche_podium(dico: dict) -> list:
    L = list(dico.items())
    L.sort(key=lambda f: f[1], reverse=True)
    res = []
    for i in range(0, 3):
        res.append(list(L[i]))
    return res
```

Activité 2 – La conjecture de Syracuse