Proposition de corrigé

Activité 1 :

Recherche séquentielle

Exercice 1 – Exercices d'échauffement

Objectif Rechercher séquentiellement un élément dans un tableau unidimensionnel ou dans un dictionnaire.

Recherche d'un nombre dans une liste

Nous allons commencer par rechercher si un nombre est dans un tableau.
Commençons par définir la liste des entiers pairs compris entre 0 et `nb` exclus.

```
def generate_pair_01(nb: int) -> list :
    """
    Génération d'une liste de nombres pairs compris entre 0 (exclus) et nb (exclus).
    """
    res = []
    for i in range(1,nb//2):
        res.append(2*i)
    return res
```

Recopier la fonction dans un fichier.

Question 1 *Vérifier que la fonction* `generate_pair_01` *fonctionne pour* `nb=0`, `nb=9`, `nb=10`.

Question 2 *Écrire une fonction de signature* `generate_pair_02(nb: int) -> list` *en utilisant une boucle* `while`.

Question 3 *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`.

Question 4 *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`.

Question 5 *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`.

Question 6 *Écrire une fonction de signature* `recherche_first_index_nb_01(nb: int, L: list) -> int` *qui renvoie l'index de la première appartion du nombre* `nb` *dans la liste* `L`. *La fonction renverra* `-1` *si* `nb` *n'est pas dans*

la liste. On utilisera une boucle `for`.

Question 7 *Écrire une fonction de signature* `recherche_first_index_nb_02(nb: int, L: list) -> int` *qui renvoie l'index de la première appartion du nombre* `nb` *dans la liste* `L`. *La fonction renverra* `-1` *si* `nb` *n'est pas dans la liste. On utilisera une boucle* `while`.

Question 8 *Écrire une fonction de signature* `recherche_last_index_nb_01(nb: int, L: list) -> int` *qui renvoie l'index de la dernière appartion du nombre* `nb` *dans la liste* `L`. *La fonction renverra* `-1` *si* `nb` *n'est pas dans la liste. On utilisera une boucle* `for`.

Question 9 *Écrire une fonction de signature* `recherche_last_index_nb_02(nb: int, L: list) -> int` *qui renvoie l'index de la dernière appartion du nombre* `nb` *dans la liste* `L`. *La fonction renverra* `-1` *si* `nb` *n'est pas dans la liste. On utilisera une boucle* `while`.

Question 10 *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.*

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

Question 11 *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`.

Question 12 *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.*

Question 13 *Écrire une fonction de signature* `compte_lettre_01(lettre: str, mot: str) -> int` *qui renvoie le nombre d'occurrences de* `lettre` *dans le* `mot`.

Les instructions suivantes permettent de charger l'ensemble des mots du dictionnaire dans la variable `dictionnaire`. `dictionnaire` est une liste de mots. Chacune des lettres de l'alphabet sont stockées dans la variable `alphabet`.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
def load_fichier(file):
    fid = open(file,'r')
    mots = fid.readlines()
    fid.close()
    return mots
dictionnaire = load_file('liste_francais.txt')
```

Question 14 *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`.

Question 15 *Quelle consonne apparaît le plus souvent ? Quelle consonne apparaît le moins souvent ? Indiquer le nombre d'occurrences dans chacun des mots*

Question 16 *Écrire une fonction de signature* `mots_plus_long(mots: list) -> str` *qui renvoie le mot le plus long.*

Question 17 *Écrire une fonction de signature* `cherche_mot_in_chaine_01(mot: str, chaine: str) -> int` *qui renvoie* `True` *si* `mot` *est dans* `chaine`, `False` *sinon. On utilisera des boucles* `for` *ou* `while`.

Question 18 *Écrire une fonction de signature* `cherche_mot_in_chaine_02(mot: str, chaine: str) -> int` *qui renvoie* `True` *si* `mot` *est dans* `chaine`, `False` *sinon. On n'utilisera ni boucle* `for` *ni* `while`.

Question 19 *Écrire une fonction de signature* `cherche_mot_in_dico(nb: int, dico: lst) -> str` *qui permet de trouver le mot de* `nb` *lettres qui est le plus contenu dans d'autres mots.*

Recherche dans un dictionnaire

Un dictionnaire (`dict`) est un type composite (au même titre que les chaînes, les listes ou les tuples). Les éléments d'un dictionnaire sont constitués d'une **clé** (alphabétique ou numérique par exemple) et d'une valeur. À la différence d'une liste par exemple, les éléments ne sont pas ordonnés.

▪ **Exemple** Dans le but de faire un comptage du nombre de lettres des mots du dictionnaire, nous allons créer un dictionnaire constitué des lettres de l'alphabet (clés) et de leur nombre d'apparitions (valeurs).

```
nb_lettres = {}
nb_lettres['a']=0
nb_lettres['b']=0
print(nb_lettres)
    {'a': 0, 'b': 0}
```

```
nb_lettres = {}
for lettre in alphabet :
    nb_lettres[lettre]=0
print(nb_lettres["a"])
    0
```

Tester l'appartenance d'une clé à un dictionnaire : `"a" in nb_lettres`.
Supprimer une clé d'un dictionnaire : `del nb_lettres["a"]`.
Parcourir un dictionnaire :

```
for clef in nb_lettres :
    print(clef)
    print(clef,nb_lettres[clef])

for clef,valeur in nb_lettres.items() :
    print(clef,valeur)
```

▪