Conseils :

- Prenez le temps de bien faire les choses et rendez une copie propre. Des points seront mis pour le soin (environ 1,5 points sur 20)..
- Aucun ordre n'est imposé pour la résolution, par contre, rendez les problèmes dans l'ordre. Numérotez avec rigueur les questions que vous traitez.
- Même si vous n'avez pas sû faire une question, vous pouvez utiliser la fonction que vous étiez censé coder pour répondre aux questions suivantes.
- L'usage des **calculatrices est interdit**.

I. À propos des dictionnaires

Q1 1. Rappeler comment créer un dictionnaire vide en python.

Q2 2. Si l'on dispose d'un dictionnaire d, comment ajouter dans ce dictionnaire une paire «clé-valeur» ? Par exemple la clé "j'aime" associée à la valeur "l'info" ?

Q3 3. Comment tester l'existence d'une clé dans un dictionnaire ? Par exemple comment tester si la clé "j'aime" est présente dans le dictionnaire d ? Quel est «l'avantage» de cette recherche par rapport à la recherche d'un élément dans une liste ?

Q4 4. Écrire une fonction `traduction(phrase:list, dico:dict)->list` qui prend en argument une "phrase" sous forme d'une liste de mot et un dictionnaire dico qui contient les traduction mot à mot ; votre fonction renverra la traduction mot à mot de la phrase (sous forme d'une liste de mot). Par exemple si un dictionnaire `fr_en` contenait des associations entre des chaines en français et des chaines en anglais, alors `traduction(["je","aime",'physique'], fr_en)` devrait renvoyer `["I",'love','physic']`.

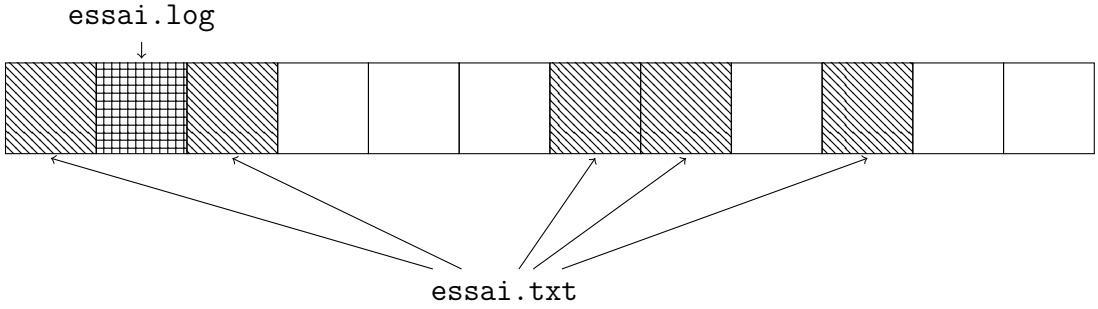
Q5 5. Écrire une fonction `inventaire` qui prend en argument une liste et qui renvoie un dictionnaire qui indique le nombre d'occurrences (valeurs dans le dictionnaire) pour chaque élément de la liste (clés). Par exemple si `L = [3.5, "a", 3.5, 3.5]`, alors `inventaire(L)` renvoie un dictionnaire contenant deux clés : 3.5 et "a", associées aux valeurs respectives 3 et 1 (3.5 est présent 3 fois et "a" est présent 1 fois dans la liste L).

II. Organisation des fichiers

Dans ce sujet nous allons étudier le principe de sauvegarde des fichiers.

Nous décrivons ici une version simplifiée de l'organisation d'un disque selon le système FAT (file allocation table).

Un disque est divisé en secteurs de taille fixe, par exemple chaque secteur occupe 512 octets. Quand on enregistre un fichier sur le disque il occupe un nombre entiers de secteurs : un fichier de 12540 octets occupera donc 25 secteurs soit un espace total de 12800 octets. Un fichier de 10 octets occupera un secteur entier. Voici un exemple d'occupation d'un disque très petit (6144 octets) ; chaque rectangle représente un secteur.



La fin du disque est occupée par deux tableaux de taille fixe. L'un décrit l'occupation de chaque secteur, l'autre indique les noms des fichiers présents sur le disque ainsi que leur emplacement.

Dans le deuxième tableau chaque fichier est décrit par son nom, la position du premier secteur et d'autres renseignements.

1 Modélisation

Les secteurs sont indicés par un entier `i` : le premier secteur est indicé par 0.

Le tableau des fichiers sera représenté dans ce problème par une liste **`TF`** dont les éléments sont des couples formés du nom du fichier sous forme d'une liste de caractères et de l'entier indiquant le premier secteur où est stocké le fichier.

La première composante du couple, le nom du fichier, est accessible par `TF[i][0]` et le second, le secteur de départ, par `TF[i][1]`.

Dans l'exemple ci-dessus on pourra avoir

```
TF = [("essai.log", 1), ("essai.txt", 0)]
```

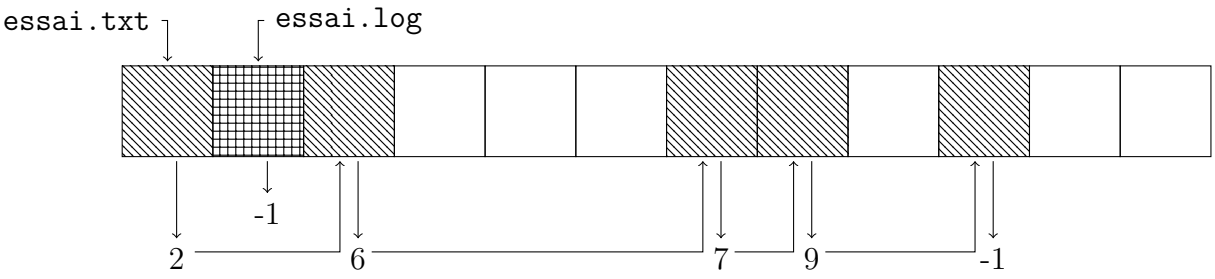
ainsi `TF[0][0]` renvoie "essai.log" et `TF[1][1]` renvoie 0.

Le tableau des secteurs sera représenté dans ce problème par une liste d'entiers **`TS`**.

- ➢ Si le secteur d'indice i n'est pas occupé alors `TS[i]` contient la valeur 0.
- ➢ Si le secteur d'indice i est occupé par un fichier alors deux cas peuvent se produire
  - — s'il reste des données à stocker lorsque le fichier a rempli le secteur i alors `TS[i]` contient la valeur du secteur suivant,
  - — si le fichier est complètement stocké, c'est-à-dire si le secteur d'indice i est le dernier secteur contenant le fichier, alors `TS[i]` contient la valeur -1.

Dans l'exemple ci-dessus on aura

```
TS = [2, -1, 6, 0, 0, 0, 7, 9, 0, -1, 0, 0 ]
```



Ainsi "essai.txt" commence à être stocké au secteur 0, la suite est stockée dans le secteur `2 = TS[0]`, puis dans le secteur `6 = TS[2]`, puis dans les secteurs 7 et 9 où stockage est terminé car `TS[9]` vaut -1.

"essai.log" est stocké dans le secteur 1 et uniquement celui-ci car `TS[1]` vaut -1.

Dans le problème on supposera définie une variable globale

```
1 taille_secteur = 512
```

Quand on aura besoin de connaître la taille d'un secteur on utilisera cette variable et non la valeur 512 qui dépend du disque.

2 Espace libre

Q6 ➢ Que fait la fonction suivante ?

```
def inconnue(TS):
    """ Entrée : une table des secteurs
        Sortie : ???."""
    n = len(TS)
    compteur = 0
    for i in range(n):
        if TS[i] == 0:
            compteur = compteur +1
    return compteur*taille_secteur
```

Dans la mesure du possible, pour améliorer la vitesse d'accès aux secteurs, le système essaie d'écrire un fichier dans des secteurs contigus c'est-à-dire dans une suite de secteurs libres placés les uns après les autres. On souhaite donc connaître la plus grande suite de secteurs libres, c'est-à-dire la plus longue suite de 0 dans la liste `TS`.

Q7 ➢ Compléter la fonction suivante : son objectif est de prendre comme paramètres un entier `i` et une liste `TS` et de renvoyer le nombre de cases contiguës du tableau contenant 0 à droite de la case d'indice i, cette case étant comprise.

```
def nombreZerosDroite(i, liste):
    """ Entrée : un indice et une table des secteurs
        Sortie : le nombre de secteurs libres à droite de i"""
    n  = len(liste)
    nombre = 0
    position = i
    while position ????? and ?????:
        nombre = ?????
        position = ?????
    return nombre
```

Dans l'exemple précédent `nombreZerosDroite(1, TS)` renverra 0, `nombreZerosDroite(3, TS)` renverra 3 et `nombreZerosDroite(10, TS)` renverra 2.

Q8 ➢ Dans le cas où `TS` est une liste de taille n ne contenant que des 0, combien de tests comparant à 0 un élément de la liste `TS` sont faits dans `nombreZerosDroite(i, TS)` ?

On remarque que le nombre maximal de zéros consécutifs est le résultat de `nombreZerosDroite(i, TS)` quand i est le premier indice de la portion de la liste qui contient le maximum de zéros.
On en déduit une fonction du calcul de la taille maximale en secteurs contigus.

```
def libreMax(liste):
    """ Entrée : une table des secteurs
        Sortie : la taille maximale libre en un seul bloc"""
    n  = len(liste)
    nbMax = 0
    for i in range(n):
        l = nombreZerosDroite(i, liste)
        if l > nbMax:
            nbMax = l
    return nbMax*taille_secteur
```

Q9 ➢ Modifier la fonction précédente en `libreMaxInd(TS)` qui renvoie, en plus de la taille maximale, l'indice du premier secteur libre dans une portion maximale. Dans le cas où tous les secteurs sont occupés cet indice peut être quelconque. Dans l'exemple précédent `libreMaxInd(TS)` renverra 1536, 3.

Q10 ➢ Dans le cas où `TS` est une liste de taille n ne contenant que des 0, combien de tests comparant à 0 un élément de la liste `TS` sont faits dans `libreMaxInd(TS)` ?

3 Position des fichiers

On peut remarquer que chaque fichier est écrit dans plusieurs secteurs mais, pour chaque fichier, il existe un, et un seul secteur pour lequel la table des secteurs renvoie -1, le dernier secteur correspondant à ce fichier.

Q11 ➢ Écrire une fonction qui reçoit comme paramètre une table des secteurs et renvoie le nombre de fichiers indiqués comme occupant au moins un secteur.

Q12 ➢ En déduire une fonction `coherence(TS,TF)` qui renvoie `True` ou `False` selon que les les deux tables (des secteurs et des fichiers) indiquent le même nombre de fichiers.
Dans la mesure du possible, on valorisera une solution courte, de la forme

```
def coherence(TS,TF):
    return ....
```

Q13 ➢ Écrire une fonction `depart(nom, TF)` qui renvoie le secteur de départ d'un fichier donné par son nom (une chaîne de caractères) en explorant la liste `TF`.
La fonction devra renvoyer -1 si le fichier n'est pas présent.
Dans l'exemple `depart("essai.log", TF)` renverra 1, alors que `depart("perdu.txt, TF)` renverra -1.

Q14 ➢ Écrire une fonction `places(nom, TS, TF)` qui renvoie la liste des secteurs occupés par un fichier donné par son nom.
La fonction devra renvoyer la liste vide si le fichier n'est pas présent.
Dans l'exemple `places("essai.txt", TS, TF)` renverra `[0, 2, 6, 7, 9]`.

4 Une amélioration

On se place de nouveau dans le cadre de la recherche d'une portion maximale de secteurs libres contigus.

Q15 ➢ Proposer une fonction `libreMaxInd1(TS)` qui renvoie la taille maximale et l'indice du premier secteur libre dans la portion maximale mais qui ne fera jamais plus de n tests pour une liste de taille n.

I. À PROPOS DES DICTIONNAIRES

Q1 1. Créer un dictionnaire vide en python `nom_dico = {}`

Q2 2. Création d'une nouvelle entrée (ou modification d'une entrée existante) `d["j'aime"] = "l'info"`

Q3 3. test d'appartenance : `"j'aime" in $d$` (renvoie True ou False) «L'avantage» de cette recherche par rapport à la recherche d'un élément dans une liste est le temps d'exécution qui ne dépend pas de la taille du dictionnaire alors qu'il est en général proportionnel à la taille de la liste. Voir le cours de 2e année pour comprendre le mécanisme derrière.

Q4 4. Écrire une fonction `traduction(phrase:list, dico:dict)->list` (cf tp)

```
#exemple de dictionnaire
#fr_en = {"je":"I","aime":"love","physique":"physic"} #non demandé ici
def traduction(phrase,dico):
    res = []
    for mot in phrase:
        res.append(dico[mot])
    return res
```

Q5 5. Écrire une fonction `inventaire` (cf tp)

```
def inventaire_dico(L:list)->dict:
    res = {}
    for x in L:
        if x in res: #déjà rencontré
            res[x] += 1 #on incrémente le nombre
        else:
            res[x] = 1 # non rencontré, on crée l'entrée
    return res
```

II. ORGANISATION DES FICHIERS

Merci à Eric Detrez d'avoir partagé ce sujet.

1 Espace libre

Q6 ➢ La fonction compte le nombre de secteurs marqués 0, les secteurs libres. Elle renvoie alors ce nombre multiplié par la taille d'un secteur : c'est la taille libre dans le disque (comme l'indique le titre).

Q7 ➢ Compléter la fonction suivante : son objectif est de prendre comme paramètres un entier `i` et une liste `TS` et de renvoyer le nombre de cases contiguës du tableau contenant 0 à droite de la case d'indice i, cette case étant comprise.
On poursuit tant qu'on peut, `position < n` et que le terme lu est nul, `and liste[position] == 0`. Dans ce cas on a trouvé une case de plus, `nombre = nombre + 1` et on passe à la position suivante `position = position +1`. Attention, c'est bien $< n$ et non $\leqslant$! Les indices valident vont de 0 à $n-1$.

```python
def nombreZerosDroite(i, liste):
    """ Entrée : un indice et une table des secteurs
        Sortie : le nombre de secteurs libres à droite de i"""
    n  = len(liste)
    nombre = 0
    position = i
    while position < n and liste[position] == 0:
        nombre = nombre + 1
        position = position +1
    return nombre
```

Q8 ➢ Dans le cas où `TS` est une liste de taille n ne contenant que des 0, combien de tests comparant à 0 un élément de la liste `TS` sont faits dans `nombreZerosDroite(i, TS)` ? On fait un test pour chaque entier de i à $n-1$: il y en a $n-i$.

Q9 ➢ Modifier la fonction précédente en `libreMaxInd(TS)` qui renvoie, en plus de la taille maximale, l'indice du premier secteur libre dans une portion maximale. Dans le cas où tous les secteurs sont occupés cet indice peut être quelconque.

```python
def libreMax(liste):
    """ Entrée : une table des secteurs
        Sortie : la taille maximale libre en un seul bloc
                 et l'indice du début"""
    n  = len(liste)
    nbMax = 0
    ind = 0
    for i in range(n):
        l = nombreZerosDroite(i, liste)
        if l > nbMax:
            nbMax = l
            ind = i
    return nbMax*taille_secteur, ind
```

Q10 ➢ Dans le cas où `TS` est une liste de taille n ne contenant que des 0, combien de tests comparant à 0 un élément de la liste `TS` sont faits dans `libreMaxInd(TS)` ?

Pour i variant de 0 à $n-1$ on fait $n-i$ tests. Le nombre total est donc $\sum_{i=0}^{n-1}(n-i) = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$.

2 Position des fichiers

Q11 ➢ On parcourt la liste en notant les termes -1 ; attention, on voulait le nombre de fichier et non le nombre de secteur occupés.

```python
def nombreFichiers(TS):
    """ Entrée : une table des secteurs
        Sortie : le nombre de fichiers inclus."""
    n = len(TS)
    nbFichiers = 0
    for i in range(n):
        if TS[i] == -1:
```

```
            nbFichiers = nbFichiers +1
    return nbFichiers
```

Q12 ➢ En déduire une fonction `coherence(TS,TF)` qui renvoie `True` ou `False` selon que les les deux tables (des secteurs et des fichiers) indiquent le même nombre de fichiers.
Le nombre de fichiers est aussi la longueur de `TF`.

```
def coherence(TS,TF):
    return nombreFichiers(TS) == len(TF)
```

== renvoie un booléen, il n'est pas utile de mettre "if ... return True else : return False".

Q13 ➢ Écrire une fonction `depart(nom, TF)` qui renvoie le secteur de départ d'un fichier donné par son nom (une chaîne de caractères) en explorant la liste `TF`.
On parcourt la liste en cherchant le nom dans la première composante, on renvoie alors la 2e composante. Attention, on veut le secteur de départ et non la position dans la liste TF.

```
def depart(nom, TF):
    """ Entrée : un nom et une table des fichiers
        Sortie : l'adresse du début du fichier"""
    n = len(TF)
    for i in range(n):
        couple = TF[i]
        if couple[0] == nom:
            return couple[1]
    return -1
```

Q14 ➢ Écrire une fonction `places(nom, TS, TF)` qui renvoie la liste des secteurs occupés par un fichier donné par son nom.
À partir du départ on suit les fichiers dans `TS` tant que le suivant n'est pas -1.

```
def places(nom, TS, TF):
    position = depart(nom, TF)
    liste = []
    while position != -1:
        liste.append(position)
        position  = TS[position]
    return liste
```

Vous avez été nombreux à faire une boucle for sur TS, mais ce n'est vraiment pas une idée optimale : vous faites de nombreux test et pour peu qu'un fichier soit stocké dans des secteurs qui ne sont pas dans l'ordre croissant, ça ne marcherait pas (je ne sais pas si c'est possible ou non, l'énoncé n'était pas explicite sur ce point).

3 Une amélioration

Q15 ➢ Proposer une fonction `libreMaxInd1(TS)` qui renvoie la taille maximale et l'indice du premier secteur libre dans la portion maximale mais qui ne fera jamais plus de n tests pour une liste de taille n.

Il suffit d'avancer la position à droite du dernier indice testé, on évite de retester les 0 que l'on vient de parcourir. Ainsi, chaque élément sera lu dans la liste une fois au plus. Si le résultat de `nombreZerosDroite(position, liste)` est `largeur` on a tout testé jusqu'à `position + largeur` donc on passe à `position + largeur + 1`

```python
def libreMaxIndMieux(liste):
    """ Entrée : une table des secteurs
        Sortie : la taille maximale libre en un seul bloc
                 et l'indice du début"""
    n  = len(liste)
    nbMax = 0
    ind = 0
    position = 0
    while position < n:
        largeur = nombreZerosDroite(position, liste)
        if largeur > nbMax:
            nbMax = largeur
            ind = position
        position  = position + largeur + 1
    return nbMax*taille_secteur, ind
```