# Informatique commune

**Devoir surveillé n°2 - Jeudi 14 janvier 2021 (durée : 2 h) v5**

**Les réponses sont toutes à donner sur la copie.**

### Exercice A

On définit la fonction suivante :

```
def modif(ch):
    L = []
    for car in ch:
        L.append(car)
    return L
```

1. On effectue l'appel suivant à la fonction `modif` : `modif("2020")`, que renverra cet appel ?
2. Expliquer en une phrase ce que réalise la fonction `modif`.

### Exercice B Fonctions mystères.

On considère les trois fonctions suivantes, prenant en entrée, une liste de flottants.

```
def mystere_1(L):
    s = 0
    for i in range(len(L)):
        s = s + L[i]
    return s / len(L)
```

```
def mystere_2(L):
    s = 0
    ind = 0
    for i in range(len(L)):
        if L[i] > 0:
            s = s + L[i]
            ind = ind + 1
    if ind != 0:
        return ind, s / ind
```

```
def mystere_3(L):
    s = 0
    ind = 0
    for i in range(len(L)):
        if L[i] < 0:
            s = s + L[i]
            ind = ind + 1
    if ind != 0:
        return ind, s / ind
```

1. Que réalise la fonction `mystere_1` ?
2. Que renvoie l'instruction `mystere_1([0, 2, 1, 3, 4, -1, -2])` ?

3. Que réalise la fonction `mystere_2` ?
4. Que renvoie l'instruction `mystere_2([0, 2, 1, 3, 4, -1, -2])` ?

5. Que réalise la fonction `mystere_3` ?
6. Que renvoie l'instruction `mystere_3([0, 2, 1, 3, 4, -1, -2])` ?

### Exercice C Exercices sur les boucles, les chaînes de caractères.

#### Comptage de mots dans une chaîne.

1. Écrire la fonction `compte` qui renvoie le nombre de mots de la chaîne de caractères prise en argument. On appelle « mot » toute suite de caractères ne contenant pas d'espace.

```
>>> compte('Nous espérons ne pas être confinés en 2021.')
8
>>> compte(' Bonne    et  heureuse année 2021     ! ')
6
```

**2.** De nombreux logiciels ou sites n'acceptent pas les espaces en particuliers dans les noms de fichier ou de répertoire. Ces espaces sont alors remplacés par des « _ » (tiret du bas).
Écrire la fonction `remplace` qui renvoie la chaîne de caractères prises en argument, mais dont les espaces ont été remplacés par des _.

```
>>> remplace('Nous espérons ne pas être confinés en 2021.')
'Nous_espérons_ne_pas_être_confinés_en_2021.'
```

## Cubes

**3.** Écrire, **en utilisant une boucle**, une fonction `cubes1` qui prend pour argument un entier positif non nul et qui renvoie le plus petit entier dont le cube est strictement supérieur à cet entier.

```
>>> cubes1(8)
3
>>> cubes1(50)
4
```

**4.** Adapter la fonction précédente, en une fonction `cubes2` qui prend pour argument un entier positif non nul et qui renvoie le plus grand cube d'un entier inférieur ou égal à l'argument.

```
>>> cubes2(27)
27
>>> cubes2(80)
64
```

### Exercice D Recherche dans une chaîne de caractères

**1.** Écrire une fonction `recherche` qui prend pour argument une chaîne de caractères et un caractère et qui renvoie `True` si le caractère est présent dans la chaîne de caractères et `False` sinon.
Donner une implémentation, la plus concise possible, utilisant l'opérateur `in` et une seconde implémentation n'y recourant pas.

```
>>> recherche('Les MPSI et les PCSI.', 'P')
True
```

**2.** Écrire une fonction `enleve` qui prend pour argument une chaîne de caractères et un caractère et qui renvoie la chaîne de caractères précédente à laquelle le caractère a été enlevé s'il a été trouvé. L'élément pourra être présent plusieurs fois dans la chaîne.

```
>>> enleve('Les MPSI et les PCSI.', 'M')
'Les PSI et les PCSI.'
```

La fonction proposée ci-dessous, est destinée à recevoir en paramètre deux chaînes de caractères `texte` et `motif` et à renvoyer la position de la première occurrence de `motif` dans `texte`, et `-1` si `motif` n'est pas une sous-chaîne de `texte`.

```
def cherche(motif,texte) :
    """en entrée : motif et texte sont des chaines de caractères non vides
    en sortie : le premier indice auquel motif apparait dans texte
    ou -1 si texte ne contient pas motif."""
    T = len(texte)
    M = len(motif)
    for i in range(  ?  ):
        if texte[i:  ?  ] == motif:
            return i
    return -1
```

**3.** Compléter les deux zones avec le point d'interrogation afin de permettre et d'optimiser la recherche.

**4.** Écrire une fonction `compte` qui prend pour argument un motif et un texte qui renvoie le nombre d'occurrences du motif dans le texte et la liste des indices du premier élément du motif pour chaque de ses occurrences (et une liste vide si le motif n'est pas présent).

```
>>> compte('SI', 'les MPSI et les PCSI')
2, [6, 18]
```

**Exercice E Recherche par dichotomie dans une liste triée**

On considère la fonction `rechdich` définie ci-dessous, censée permettre la recherche par dichotomie d'une valeur dans une liste triée.

Cette fonction prend deux arguments, le premier est une liste dont les éléments sont classés dans l'ordre croissant et le second la valeur à rechercher dans la liste.

```
[1]     def rechdich(L, val):
[2]         while len(L) >= 1:
[3]             print(L)       #affichage de la liste L
[4]             m = len(L) // 2
[5]             if L[m] > val:
[6]                 L = L[:m]
[7]             elif L[m] < val:
[8]                 L = L[m:]
[9]             else:
[10]                return True
[11]        return False
```

On considère la liste `L` suivante, composée de 11 éléments :

```
>>> L = ['a', 'b', 'b', 'c', 'd', 'e', 'f', 'h', 'i', 'k', 'l']
```

À l'appel `rechdich(L, 'd')` (recherche de l'élément `'d'` dans la liste `L`), on obtient les 5 lignes ci-dessous :

```
>>> rechdich(L, 'd')
['a', 'b', 'b', 'c', 'd', 'e', 'f', 'h', 'i', 'k', 'l']
['a', 'b', 'b', 'c', 'd']
['b', 'c', 'd']
['c', 'd']
True
```

On a donc quatre itérations avec les valeurs suivantes affichées :

| | Affichage |
|---|---|
| Itération 1 | `['a', 'b', 'b', 'c', 'd', 'e', 'f', 'h', 'i', 'k', 'l']` |
| Itération 2 | `['a', 'b', 'b', 'c', 'd']` |
| Itération 3 | `['b', 'c', 'd']` |
| Itération 4 | `['c', 'd']` |

À la 4ème itération, la valeur `'d'` est trouvée, et la fonction renvoie `True`.

**1.** Comme dans l'exemple précédent, déterminer ce qu'affiche et renvoie la commande suivante :

```
>>> rechdich(L, 'f')
```

**2.** En fait, il s'avère que la fonction `rechdich` comporte une erreur d'implémentation.

**a.** Déterminer ce qu'affiche et renvoie la commande suivante :

```
>>> rechdich(L, 'g')
```

**b.** Proposer une modification minimale de la fonction `rechdich` qui corrige son défaut d'implémentation.

On indiquera le (ou les) numéro(s) des lignes à modifier et on écrira cette (ou ces) lignes modifiée(s), précédée(s) de leur numéro de ligne.

**Exercice F Conjugaisons**

On définit les listes suivantes

```
sujetsm = ["Je", "Tu", "Il", "Nous", "Vous", "Ils"]
sujetsf = ["Je", "Tu", "Elle", "Nous", "Vous", "Elles"]
verbes  = ["jouer", "chanter", "musarder"]
terminaisons = ["e", "es", "e", "ons", "ez", "ent"]
```

**1.** Écrire une fonction `conjugue`, prenant en paramètre un verbe du premier groupe, sous la forme d'une chaîne de caractères, et qui affiche sa conjugaison au présent de l'indicatif.
L'affichage devra être conforme à l'exemple 1 ci-dessous, et respecter l'alignement de la première lettre du verbe.

**2.** Écrire une fonction `conjugue2`, prenant, de même, en paramètre un verbe, et qui affiche la conjugaison du verbe en paramètre, comme dans l'exemple 2, ci-dessous.

- **Exemple 1** (question 1)

```
>>> conjugue("chanter")
Je   chante
Tu   chantes
Il   chante
Nous chantons
Vous chantez
Ils  chantent
```

- **Exemple 2** (question 2)

```
>>> conjugue2("chanter")
Je chante
Tu chantes
Il ou Elle chante
Nous chantons
Vous chantez
Ils ou Elles chantent
```

**3.** Compléter l'instruction suivante, permettant de définir la liste `pronoms` ci-dessous, à partir des listes `sujetsm` et `sujetsf`, en utilisant le moins de fois possible les noms `sujetsm` et `sujetsf` et sans saisir de chaîne de caractères.

```
>>> pronoms = ...
>>> pronoms
['Je', 'Tu', 'Il', 'Nous', 'Vous', 'Ils', 'Elle', 'Elles']
```

**4.** Écrire une fonction `exquis`, qui produit, en choisissant un pronom, masculin ou féminin, et un verbe au hasard dans les listes pronoms et verbes une phrase grammaticalement correcte.
On utilisera la fonction `randint`, du module `random`, et dont la syntaxe est telle que `randint(2, 5)` renvoie un entier pseudo-aléatoire compris au sens large entre 2 et 5.
Les listes définies au début seront utilisées en tant que variables globales.
**On respectera la structure de script au-dessous de l'exemple.**

*Exemple* :

```
>>> exquis()
Elles chantent
```

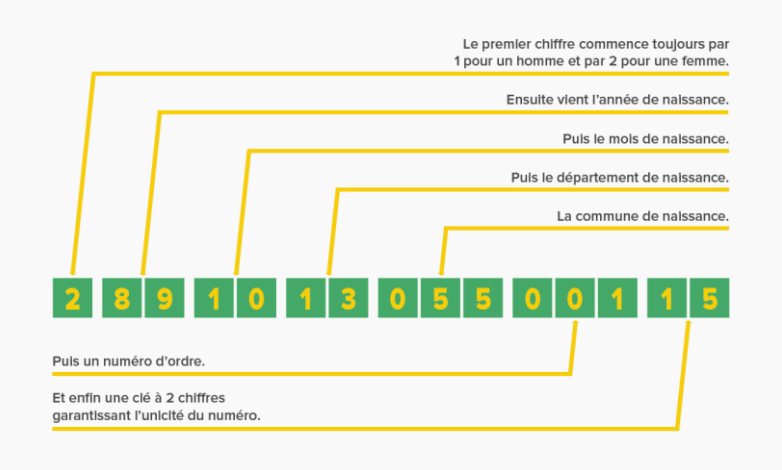
```
>>> exquis()
Nous musardons
```

**Structure à respecter :**

```
def exquis():
    pronoms = ['Je', 'Tu', 'Il', 'Nous', 'Vous', 'Ils', 'Elle', 'Elles']
    a = randint(...
    b = randint(...
    sujet = ...
    verbe = ...
    if  ... :
        terminaison = ...
    else:
        terminaison = ...
    phrase = ...
    print(phrase)
```

### Exercice G Numéro INSEE (vérifier peut-être la *crédibilité* des exemples)

Un numéro de sécurité sociale (INSEE) se décompose de la façon suivante :



La clé, appelée clé NIR, d'un numéro de sécurité sociale, encore appelé numéro INSEE est calculée avec la formule suivante :

« Clé NIR = 97 – ((valeur numérique du NIR) modulo 97) »

Pour le numéro INSEE en exemple ci-dessus, 2 89 10 13 055 001 15, la valeur numérique est l'entier 2891013055001, et la clé NIR est 15 (**note** : **ce numéro-exemple n'est pas valide**).

**1.** Écrire une fonction `verif`, prenant en paramètre un numéro INSEE, composé de 15 chiffres, au format `str`, et renvoyant un booléen, `True` ou `False`, indiquant si le numéro, vérifié par la formule indiquée ci-dessus, est valide ou non.

*Exemple :*

```
>>> verif('289101305500115')
False
```

```
>>> verif('182097523801588')
True
```

On propose de transformer ce code. Le nouveau codage consiste à sommer les 13 entiers de la partie numérique du numéro de sécurité sociale pour obtenir le code de sécurité. Si cette somme est supérieure à 99, alors le chiffre des centaines, `'1'`, est tronqué.

Par exemple, pour les parties numériques suivantes :

`'2891013055001'` → 2+8+9+1+0+1+3+0+5+5+0+0+1=35, le nouveau numéro avec le code sera : `'289101305500135'`

`'2779999988997'` → 2+7+7+9+9+9+9+9+9+8+8+9+9+7=102, le nouveau numéro avec le code sera : `'277999998899702'`.

**2.** Écrire une fonction `code`, prenant en paramètre un numéro INSEE, composé de 15 chiffres (avec l'ancien code), au format `str`, et renvoyant le numéro INSEE, composé de 15 chiffres (avec le nouveau code) au format `str`.

### Exercice H Crible d'Ératosthène

L'algorithme du crible d'Ératosthène permet d'obtenir la liste des nombres premiers inférieurs ou égaux à un entier naturel $n$, $n \geq 2$.

On rappelle qu'un nombre premier est un entier naturel supérieur ou égal à 2 admettant exactement deux diviseurs, 1 et lui-même.

L'algorithme peut être mis œuvre de la façon suivante, pour un entier $n \geq 2$ donné.

- On écrit la liste des entiers naturels, compris entre 2 et $n$.

Puis, on élimine de cette liste - ici, en les remplaçant par la valeur 0, successivement, les multiples des nombres identifiés comme premiers dans la liste (il s'agit à chaque fois le plus petit entier non nul présent dans la liste, après le dernier entier premier dont on a éliminé les multiples).

Ainsi :

- on élimine les multiples de 2. Le plus petit entier non nul suivant dans la liste est alors 3.
- on élimine les multiples de 3. Le plus petit entier non nul suivant dans la liste est alors 5.
- on élimine les multiples de 5. Le plus petit entier non nul suivant dans la liste est alors 7.

…

On arrête le processus itératif lorsque le plus petit entier non nul suivant dans la liste est inférieur ou égal à $\sqrt{n}$.

Après application de cet algorithme, les nombres premiers inférieurs ou égaux à $n$ sont les entiers non nuls encore présents dans la liste.

**1.** Écrire une fonction `Linit`, prenant en paramètre un entier $n \geq 2$, et renvoyant la liste des entiers compris entre 2 et $n$ inclus, précédés de deux valeurs nulles.

```
>>> Linit(20)
[0, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

**2.** Écrire une fonction `annule`, prenant en paramètre une liste `L` et un entier $m \geq 2$, et remplaçant par la valeur zéro tous les éléments d'une liste `L` dont les indices de position sont des multiples de $m$ strictement supérieurs à $m$.

```
>>> L = Linit(20)
>>> annule(L, 3)
>>> L
[0, 0, 2, 3, 4, 5, 0, 7, 8, 0, 10, 11, 0, 13, 14, 0, 16, 17, 0, 19, 20]
```

**3.** Écrire une fonction `divmax`, prenant en paramètre un entier naturel $n$, et renvoyant le plus grand entier $d$, inférieur ou égal à la racine carrée d'un entier $n$.

On pourra recourir à une fonction prédéfinie (`int`, `floor` (partie entière), `sqrt`, …, en effectuant si besoin les importations nécessaires) ou un opérateur (`**0.5`, `//`, …).

```
>>> divmax(19)
4
```

```
>>> divmax(25)
5
```

**4.** Écrire une fonction `elimine`, prenant en paramètre une liste `L` et une valeur `v`, et renvoyant une nouvelle liste composée uniquement des éléments de `L` non égaux à `v`, sans en changer l'ordre et sans en éliminer aucun.

```
>>> elimine([1, 0, 0, 2, 0, 4, 2], 0)
[1, 2, 4, 2]
>>> elimine([1, 1, 3, 2, 5, 4, 2], 0)
[1, 1, 3, 2, 5, 4, 2]
```

**5.** Écrire une fonction, alternative à la précédente, mais non équivalente, `elimine1`, prenant en paramètre une liste `L` et une valeur `v`, et supprimant de `L` les éléments égaux à `v`.

On devra utiliser l'instruction `del`, dont la syntaxe est rappelée ci-dessous :

```
>>> L = ['a', 'b', 'b', 'c']
>>> del L[1]
>>> L
['a', 'b', 'c']
>>> del L[-1]
>>> L
['a', 'b']
```

*Exemple* :

```
>>> L = [1, 0, 0, 2, 0, 4, 2]
>>> elimine1(L, 0)
>>> L
[1, 2, 4, 2]
```

**6.** Écrire un script dans lequel on applique l'algorithme du crible pour obtenir la liste des nombres premiers inférieurs ou égaux à 1000.

On fera impérativement usage des fonctions précédemment définies autant que faire se peut.

Les premières lignes du script seront impérativement :

```
N = 1000
L = Linit(N)
M = ...
...
while ...
...
```

- *Exemple* d'exécution du script en affichant l'état de la liste `L`

Si l'on cherche la liste des entiers premiers inférieurs ou égaux à `N = 30`, le plus grand entier inférieur ou égal à la racine carrée de 30 ($\sqrt{30} \approx 5{,}48$) est 5, et on a pour la liste `L` la suite d'états suivants :

- état initial de la liste :

```
[0, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30]
```

- après élimination des multiples de 2 :

```
[0, 0, 2, 3, 0, 5, 0, 7, 0, 9, 0, 11, 0, 13, 0, 15, 0, 17, 0, 19, 0, 21, 0, 23, 0, 25, 0,
27, 0, 29, 0]
```

- après élimination des multiples de 3 :

```
[0, 0, 2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19, 0, 0, 0, 23, 0, 25, 0, 0,
0, 29, 0]
```

- après élimination des multiples de 5 :

```
[0, 0, 2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19, 0, 0, 0, 23, 0, 0, 0, 0,
0, 29, 0]
```

La liste des entiers premiers inférieurs ou égaux à 30 que l'on obtient alors est :

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

## Exercice I Étude de trafic routier (d'après l'épreuve d'informatique concours Mines-Ponts MP, PC PSI 2017)

Ce sujet concerne la conception d'un logiciel d'étude de trafic routier. On modélise le déplacement d'un ensemble de voitures sur des files à sens unique (voir Figure 1(a) et 1(b)). C'est un schéma simple qui peut permettre de comprendre l'apparition d'embouteillages et de concevoir des solutions pour fluidifier le trafic.

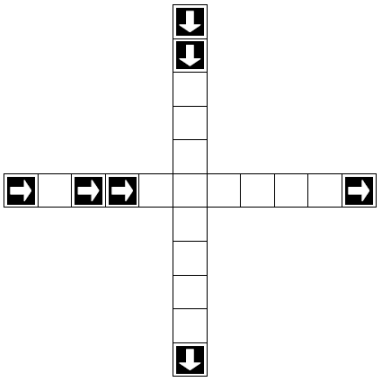
Le sujet comporte des questions de programmation. Le langage à utiliser est `Python`.

### Notations

Soit $L$ une liste,

- on note $len(L)$ sa longueur ;
- pour $i$ entier, $0 \leq i < len(L)$, l'élément de la liste d'indice $i$ est noté $L[i]$ ;
- pour $i$ et $j$ entiers, $0 \leq i < j \leq len(L)$, $L[i : j]$ est la sous-liste composée des éléments $L[i]$, ..., $L[j-1]$ ;
- $p * L$, avec $p$ entier, est la liste obtenue en concaténant $p$ copies de $L$. Par exemple, $3 * [0]$ est la liste [0, 0, 0].

(a) Représentation d'une file de longueur onze comprenant quatre voitures, situées respectivement sur les cases d'indices 0, 2, 3 et 10.



(b) Configuration représentant deux files de circulation à sens unique se croisant en une case. Les voitures sont représentées par un carré noir.

FIGURE 1 – Files de circulation

### Partie I. Préliminaires

Dans un premier temps, on considère le cas d'une seule file, illustré par la Figure 1(a). Une file de longueur $n$ est représentée par $n$ cases. Une case peut contenir au plus une voiture. Les voitures présentes dans une file circulent toutes dans la même direction (sens des indices croissants, désigné par les flèches sur la Figure 1(a)) et sont indifférenciées.

Ainsi, la file de la figure 1(a) sera représentée par la liste suivante :

```
A = [True, False, True, True, False, False, False, False, False, False, True]
```

• **Q1** – Soit $L$ une liste représentant une file de longueur $n$ et $i$ un entier tel que $0 \leq i < n$. Définir en `Python` la fonction `occupe(L, i)` qui renvoie `True` lorsque la case d'indice $i$ de la file est occupée par une voiture et `False` sinon.

• **Q2** Combien existe-t-il de files différentes de longueur $n$ ? Justifier votre réponse.

• **Q3** – Écrire une fonction `egal(L1, L2)` retournant un booléen permettant de savoir si deux listes $L1$ et $L2$ sont égales.

### Partie II. Déplacement de voitures dans la file

On identifie désormais une file de voitures à une liste. On considère les schémas de la Figure 2 représentant des exemples de files. Une *étape de simulation pour une file* consiste à déplacer les voitures de la file, à tour de rôle, en commençant par la voiture la plus à droite, d'après les règles suivantes :

- une voiture se trouvant sur la case la plus à droite de la file sort de la file ;
- une voiture peut avancer d'une case vers la droite si elle arrive sur une case inoccupée ;
- une case libérée par une voiture devient inoccupée ;
- la case la plus à gauche peut devenir occupée ou non, selon le cas considéré.

On suppose avoir écrit en `Python` la fonction `avancer` prenant en paramètres une liste de départ, un booléen indiquant si la case la plus à gauche doit devenir occupée lors de l'étape de simulation, et renvoyant la liste obtenue par une étape de simulation.
Par exemple, l'application de cette fonction à la liste illustrée par la Figure 2(a) permet d'obtenir soit la liste illustrée par la Figure 2(b) lorsque l'on considère qu'aucune voiture nouvelle n'est introduite, soit la liste illustrée par la Figure 2(c) lorsque l'on considère qu'une voiture nouvelle est introduite.

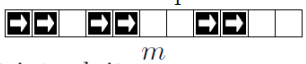
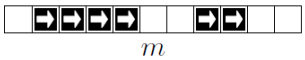
(a) Liste initiale $A$

(b) $B =$ `avancer`$(A,$ `False`$)$

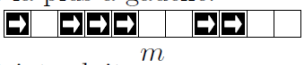
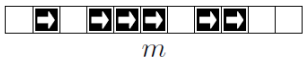
(c) $C =$ `avancer`$(A,$ `True`$)$

FIGURE 2 – Étape de simulation

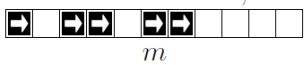
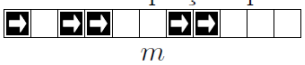
• **Q4** Etant donnée *A* la liste définie précédemment, que renvoie `avancer(avancer(A, False), True)` ?
Pour toutes les questions suivantes ci-dessous, on ne fera pas de boucle, mais on utilisera le *slicing*.

• **Q5** – On considère une liste $L$ dont la case d'indice $m > 0$ est temporairement inaccessible et bloque l'avancée des voitures. Une voiture située immédiatement à gauche de la case d'indice $m$ ne peut pas avancer. Les voitures situées sur les cases plus à gauche peuvent avancer, à moins d'être bloquées par une case occupée, les autres voitures ne se déplacent pas. Un booléen $b$ indique si une nouvelle voiture est introduite lorsque cela est possible.
Par exemple, la file  devient  lorsque aucune nouvelle voiture n'est introduite.

Définir en `Python` la fonction `avancer_debut_bloque`$(L, b, m)$ qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste.
Définir en `Python` la fonction `avancer_fin`$(L, m)$ qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste sans modifier $L$.

• **Q6** – Soient $L$ une liste, $b$ un booléen et $m$ l'indice d'une case *inoccupée* de cette liste. On considère une étape partielle où seules les voitures situées à gauche de la case d'indice $m$ se déplacent, les autres voitures ne se déplacent pas. Le booléen $b$ indique si une nouvelle voiture est introduite sur la case la plus à gauche.
Par exemple, la file  devient  lorsque aucune nouvelle voiture n'est introduite.

Définir en `Python` la fonction `avancer_debut`$(L, b, m)$ qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste sans modifier $L$.

• **Q7** – On considère $L$ une liste et $m$ l'indice d'une case de cette liste $(0 \leq m < len(L))$. On s'intéresse à une *étape partielle* où seules les voitures situées sur la case d'indice $m$ ou à droite de cette case peuvent avancer normalement, les autres voitures ne se déplaçant pas.
Par exemple, la file  devient 

## Exercice J Calcul de grandeurs statistiques

On considère des listes de $N$ ($N \geq 1$) couples de la forme $L = [[v_0,\ n_0], [v_1,\ n_1], \ldots, [v_i,\ n_i], \ldots, [v_{N-1},\ n_{N-1}]]$.
Ces listes $L$ seront implémentées sous la forme de listes de listes composées d'un flottant et d'un entier.
Une telle liste $L$ représentera une série statistique pour laquelle on a observé pour chaque valeur $v_i$ un effectif égal à $n_i$. **On supposera que ces listes sont ordonnées dans l'ordre croissant des valeurs $v_i$.**

1. Écrire une fonction `moyenne`, prenant en argument une telle liste, et renvoyant la moyenne de la série statistique qu'elle décrit.
2. Écrire une fonction `effcum`, prenant en argument une telle liste $L$ et renvoyant la liste des effectifs cumulés croissants correspondants, c'est-à-dire la liste des valeurs $(c_i)_{0 \leq i \leq N-1}$ où $c_i = \sum_{j=0}^{i} n_j$.
3. Écrire une fonction `quartile`, prenant en argument une telle liste $L$ et un entier $q$, égal à 1, 2 ou 3, et renvoyant la plus petite valeur $v_i$ pour laquelle le nombre de valeurs observées inférieures ou égales à $v_i$ est supérieure ou égale à $q \times \frac{N}{4}$.