I.S.N. TP 4 – Les tableaux

1 Introduction

Ce TP a pour objectif d'aborder la notion de tableau en informatique. Il est basé sur les deux ressources suivantes, qu'on pourra consulter le moment venu pour avoir davantage d'explications, de vocabulaire, d'exemples.

- Dowek et coauteurs, Informatique et sciences du numérique, page 50 du livre, c.-à-d. page 64 du fichier. Lien ici, 22.78 Mio
- Apprendre à programmer avec Python 3, de Gérard Swinnen : livre disponible en format pdf (licence CC) `http://inforef.be/swi/download/apprendre_python3_5.pdf`, 6.12 Mio
  - Les listes (première approche), page 44 du livre, c.-à-d. page 64 du fichier,
  - Le point sur les listes, page 141 du livre, c.-à-d. page 161 du fichier.

2 La notion de tableau (c.-à-d. de liste en python)

2.1 Types simples, types composites

En Python, nous avons déjà rencontré les types entier (`int`), nombre à virgule flottante (`float`), booléen (`bool`). On les appelle des *types simples*.

Nous avons aussi rencontré le type chaîne de caractères (`str`). On dit que c'est un *type composite* (une chaîne de caractère est composée de plusieurs caractères [1]).

Dans ce TP, on introduit un nouveau type composite. Ce type est appelé **tableau** dans de nombreux langages de programmation, mais en python on l'appelle **liste**.

2.2 Définition abstraite d'une liste

En python, on peut considérer qu'une liste est une *collection d'éléments séparée par des virgules, l'ensemble étant enfermé par des crochets.*

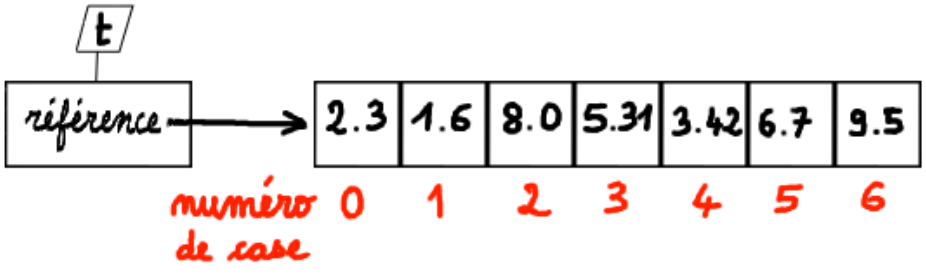
2.3 Exemple

```
# creation d'une liste, affectation a la variable t
t = [2.3, 1.6, 8.0, 5.31, 3.42, 6.7, 9.5]
type(t)
```

►► Question A : Qu'affiche l'instruction `type(t)` ?

2.4 Le contenu en mémoire de l'ordinateur

En fait, l'ordinateur a réservé de la place dans la mémoire pour 7 valeurs, il a stocké ces 7 valeurs, comme schématisé par la boîte à 7 cases ci-dessous, puis il a créé une référence pour accéder au contenu de cette « boîte », et cette référence a été enregistrée dans la variable `t`.



1. Dans d'autres langages, il existe un autre type simple : le type caractère `char`.

2.5 Utilisation des cases d'une liste

Les cases d'une liste sont numérotés à partir de 0. Si `t` est une liste, et si `i` est le numéro d'une case de la liste, alors la valeur contenue dans la case est `t[i]`. On peut donc :

- accéder à cette valeur, et par exemple l'afficher avec `print(t[i])`
- modifier cette valeur, avec `t[i]= nouvelle_valeur`

```
t = [2.3, 1.6, 8.0, 5.31, 3.42, 6.7, 9.5]
print(t[0])  # 2.3
print(t[2])  # 8.0
print(t[6])  # 9.5
print(t[7])  # IndexError: list index out of range
t[1]=1.0
t[5]=5.0
print(t)
```

►► Question B : Expliquer pourquoi `print(t[7])` renvoie une erreur.

►► Question C : Qu'affiche l'instruction `print(t)` à la fin ?

2.6 Créer une liste de grande taille

On peut utiliser une instruction comme par exemple `t = [0 for i in range(0,10)]` ce qui donne : `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

On peut aussi remplacer `10` par `100`, ou par une variable.

►► Question D : Comment créer une liste dont la longueur est déterminée par l'utilisateur d'un programme ?

2.7 Utilisation d'une boucle `pour` avec une liste

```
lst = [0 for i in range(0,7)]
for i in range(0,7):
    lst[i] = i**2
print(lst)
```

►► Question E : Qu'affiche l'instruction `print(lst)` ?

2.8 Autres instructions

L'instruction `len(ma_liste)` donne le nombre de cases de la liste (`len` correspond à length qui signifie longueur).

```
lettres = ['a', 'b', 'c', 'd', 'e', 'f']
print(len(lettres))
```

►► Question F : Qu'affiche l'instruction `print(len(lettres))` ?

►► Question G : Pour une liste `lst`, quel est le numéro de la dernière case ? (Répondre en utilisant `len`)

3 Exercices sur les listes

1. Écrire un programme qui compte le nombre d'éléments supérieurs à 10 dans une liste d'entiers.
2. Écrire un programme qui trouve l'élément maximal dans une liste d'entiers.
3. Écrire un programme qui affiche les éléments d'une liste, du dernier au premier. Par exemple si la liste donnée est `[1, 2, 3, 4, 5, 6]`, alors le programme affiche `6 5 4 3 2 1`.
4. Écrire un programme qui remplace les éléments d'une liste par les éléments permutés de façon circulaire comme ceci : si `t=[1, 2, 3, 4, 5]` alors à la fin `t` vaudra `[2, 3, 4, 5, 1]`.

5. Écrire un programme qui analyse tous les éléments d'une liste de nombres entiers pour générer deux nouvelles listes. L'une contiendra seulement les nombres *pairs* de la liste initiale, et l'autre les nombres *impairs*. Par exemple, si la liste initiale est `[32, 5, 12, 8, 3, 75, 2, 15]`, alors le programme devra construire une liste `pairs` qui contiendra `[32, 12, 8, 2]` et une liste `impairs` qui contiendra `[5, 3, 75, 15]`.
6. Exercices du site **France IOI - Découverte des tableaux**
   - 2. Préparation de l'onguent – Accès à une case d'une liste
   - 4. Liste de courses – Accès à une case d'une liste
   - 5. Grand inventaire – Modifier les éléments d'un tableau
   - 6. Étude de marché – Tableaux de taille variable
   - 7. Répartition du poids – Tableaux de taille variable
   - 8. Visite de la mine – Tableaux de taille variable

4 Tableaux à deux dimensions

On cherche à représenter de façon informatique le tableau à double entrée suivant :

120	145	87	8
12	67	89	24
90	112	83	47

4.1 Ligne par ligne

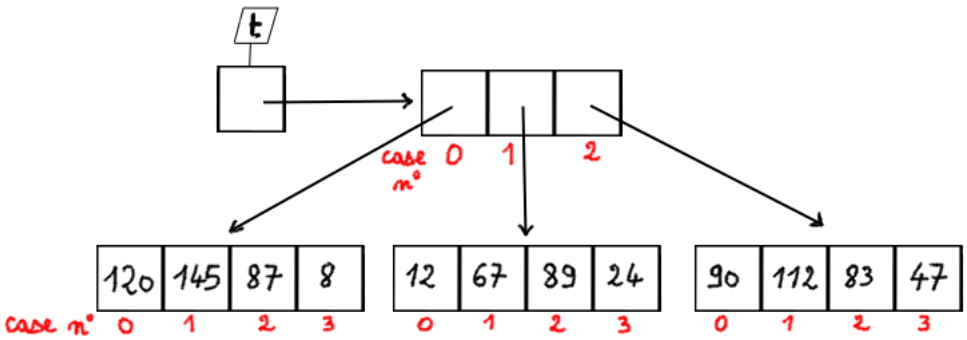
Une première idée est d'utiliser la liste `[120, 145, 87, 8, 12, 67, 89, 24, 90, 112, 83, 47]`. Mais dans certains cas, cette idée n'est pas pratique.

On préfère donc généralement considérer chaque ligne comme une liste, comme ceci :
`[120, 145, 87, 8]`, puis `[12, 67, 89, 24]`, puis `[90, 112, 83, 47]`.
Ensuite, on crée une « liste de listes » comme ceci :
`t = [[120, 145, 87, 8], [12, 67, 89, 24], [90, 112, 83, 47]]`.
En Python, on dit que l'on a fait une liste à deux dimensions (et dans d'autres langages on parlera de tableaux à 2 dimensions).

On a donc, par exemple :

- `t[0]=[120, 145, 87, 8]`, `t[1]=[12, 67, 89, 24]`, `t[2]=[90, 112, 83, 47]`
- `t[0][0]=120`, `t[0][1]=145`, `t[0][2]=87`, `t[0][3]=8`
- `t[1][0]=12`, `t[1][1]=67`, `t[1][2]=89`, `t[1][3]=24`
- `t[2][0]=90`, `t[2][1]=112`, `t[2][2]=83`, `t[1][1]=47`
- Cas général : `t[i][j]` est le contenu de la case à la ligne `i` et à la colonne `j`.

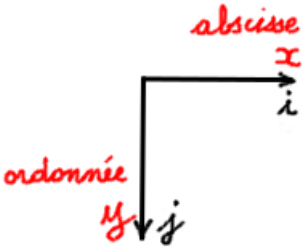
On peut schématiser le contenu de la mémoire de l'ordinateur de la façon suivante :



La variable `t` contient une référence vers une « boîte de 3 cases » contenant chacune une référence vers une « boîte de 4 cases » contenant chacune un nombre.

4.2 Colonne par colonne

Il est tout à fait possible de faire un autre choix : considérer chaque colonne comme une liste. Dans ce cas, `t[0]=[120, 12, 90]`, `t[1]=[145, 67, 112]`, etc.
Alors `t[i][j]` est le contenu de la case à la colonne `i` et à la ligne `j`.
On utilise parfois les notations `t[x][y]` par analogie avec les abscisses et les ordonnées (pour les images informatiques, les ordonnées « augmentent vers le bas »).



4.3 Générer un tableau à deux dimensions

Exemple : on veut initialiser un tableau rempli de zéros comme ci-contre :

0	0	0	0
0	0	0	0
0	0	0	0

On peut saisir les instructions suivantes :

- En pensant « ligne par ligne » :

```
t = [[0 for colonne in range(0, 4)] for ligne in range(0, 3)]
print(t)  # [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

# formule generale
t = [[0 for colonne in range(0, nbColonnes)] for ligne in range(0, nbLignes)]
```

- En pensant « colonne par colonne » :

```
t = [[0 for ligne in range(0, 3)] for colonne in range(0, 4)]
print(t)  # [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]

# formule generale
t = [[0 for ligne in range(0, nbLignes)] for colonne in range(0, nbColonnes)]
```

4.4 Exercices

4.4.1 Moyennes des élèves d'une classe

Créer une liste de `3x5` éléments appelée `notes`, qui respecte les conditions suivantes :

- Pour `i=0..2`, `notes[i][0]` est le prénom de l'élève (une chaîne de caractères)
- Pour `i=0..2` et `j=1..4`, `notes[i][j]` est la note numéro `j` de l'élève.
- Notes d'Arthur : 12 ; 8 ; 15 ; 16. Notes de Basile : 7 ; 11 ; 14 ; 10. Notes de Camille : 18 ; 15 ; 6 ; 13.

4.4.2 Jeu du Tic-tac-toe

On souhaite utiliser un tableau `jeu` de dimensions `3x3` pour suivre l'avancement d'une partie de Tic Tac Toe.

Chaque case du tableau contient un caractère `'.'` ou bien `'O'` ou bien `'X'`.

Compléter le programme, composé des instructions suivantes :

- Créer le tableau, et l'initialiser avec des `'.'`
- Écrire 5 instructions de type `jeu[ligne][colonne]=...` qui correspondent à la partie déjà commencée ci-dessous :

```
..X
OOX
..O
```

- Écrire des instructions qui permettent d'afficher le tableau de façon lisible dans la console (comme ci-dessus par exemple).

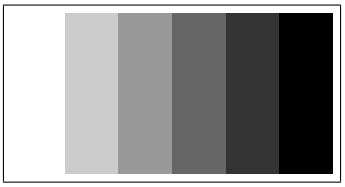
Voici le squelette du programme à compléter :

```
# ici creation de la liste 3x3
print(jeu) # affichage pas vraiment lisible

# ici les 5 instructions de modification du tableau pour les coups joues
#
#
#
#
#

# ici les instructions pour l'affichage du jeu
#
#
#
#
```

4.4.3 Une image

On s'intéresse à une image de dégradé du Blanc vers le Noir, au format `pnm`, de largeur 6 pixels, et de hauteur 3 pixels. On rappelle que `BLANC=255`, et `NOIR=0`.

On décide ensuite de traiter les données sous la forme d'un tableau à deux dimensions, appelé `pixels`, de sorte que `pixels[compteurX][compteurY]` est la valeur de gris à utiliser pour le pixel de la colonne `compteurX` et de la ligne `compteurY` (en commençant à compter à partir de 0).



Déterminer le contenu du tableau `pixels`, et stocker ce tableau dans une variable sous python.