Python 20 – Indices et Tableaux

I – Connaissances de base

1 – Tableau (statique) en informatique

Un **tableau statique** (ou tableau tout court) est une structure de données ayant les propriétés suivantes :

- Les données sont stockées dans des cases possèdant un numéro nommé indice (ou index en anglais).
- Nombre de cases est constant après création.
- Même type de données dans toutes les cases

2 – Déclaration d'un tableau par extension

→ On fournit toutes les variables à la main entre [ ]

```
>>> t = [1, 10, 100, 1000]
```

→ On sépare les valeurs par une virgule

→ Tableau statique : nombre de cases fixe.

3 - Nombre d'éléments stockés avec len()

```
>>> t = [1, 10, 100, 1000]
>>> nbr = len(t)
>>> nbr
4
```

4 - Accès à l'un des éléments avec [indice]

```
>>> t = [1, 10, 100, 1000]
         0   1   2    3

>>> t[0]            >>> t[1]
1                   10
>>> t[2]            >>> t[3]
100                 1000

>>> t[4]
IndexError
```

5 - Lecture des éléments par indice

(méthode classique, numérique avec range)

```
01 t = [1, 10, 100, 1000]
02 for i in range( len(t) ):
03     print(t[i], end="-")
```

La variable de boucle **i** prend les valeurs **0,** puis **1**, puis **2** puis **3**.

La console affiche alors **: 1-10-100 1000**

6 - Lecture directe des éléments un à un

(méthode nominative, sans range)

```
01 t = [1, 10, 100, 1000]
02 for v in t:
03     print(v, end="-")
```

La variable de boucle **v** prend les valeurs **1,** puis **10**, puis **100** puis **1000**.

La console affiche alors **: 1-10-100-1000**

!!! Choix entre les deux types de for !!!

→ si on ne veut que lire : les deux sont possibles.
→ si on a besoin de connaître l'indice : avec range.
→ si on veut modifier le tableau : avec range.

Si on veut faire la moyenne des valeurs dans un tableau, il suffit de faire ceci :

```
01 t = [1, 10, 100, 1000]
02 somme = 0
03 for valeur in t:
04     somme = somme + valeur
05 moyenne = somme / len(t)
```

7 - Mutabilité des tableaux EN PYTHON

Mutable ou muable
Strucutre de données dont *le contenu* peut être modifié après son initialisation.

En Python, les tableaux de type **list** sont ***muables.*** On peut modifier le contenu des cases d'un tableau sans modifier l'adresse du tableau lui-même.

Exemple 1

```
>>> t = [1, 10, 100]
>>> type(t)
<class 'list'>
>>> t[0] = 2000
>>> t
[2000, 10, 100]
```

Exemple 2

```
01 notes = [15, 18, 8]
02 for i in range(len(notes)):
03     notes[i] = notes[i] * 2
```

Exemple QUI NE FONCTIONNERA PAS

```
01 notes = [15, 18, 8]
02 for v in notes:
03     v = v * 2
```

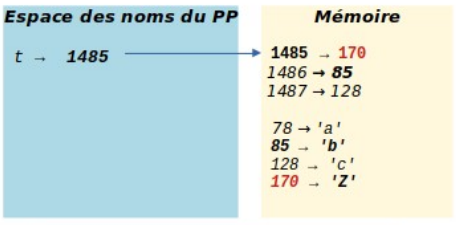
8 - Explications via l'ESPACE DES NOMS

*La variable **t** contient l'adresse de la structure qui va permettra stocker les données (l'armoire).*

*Si **t** contient initialement* **['a', 'b', 'c', 'd'],** Python affecterait, par exemple, une adresse **1485**.

L'adresse (1485+0) contient l'adresse du contenu de t[0].
L'adresse (1485+1) contient l'adresse du contenu de t[1].
L'adresse (1485+2) contient l'adresse du contenu de t[2].

t[0] = 'Z', on ne modifie pas l'adresse (1485) de **t**. On va juste faire pointer la zone-mémoire (1485+0) vers 170 qui mène alors à **'Z'**




```
t[0] = 'Z'
```

i → _____

t → _____

t[i] → _____

II – Création d'un tableau par compréhension

10 - Déclaration par compréhension avec tableau initial

```
>>> base = [10, 10, 20, 40]
>>> new = [None for element in base]
>>> new
[None, None, None, None]
```

Traduction en français : Il faut le lire de la droite vers la gauche ; Pour chaque élement qu'on trouve dans le tableau **base**, crée une nouvelle case contenant **None** à la fin du tableau **new.**

```
>>> b = [10, 10, 20, 40]
>>> dd = [v * 2 for v in b]
>>> dd
[20, 20, 40, 80]
```

Traduction en français : Il faut le lire de la droite vers la gauche ; Pour chaque valeur **v** qu'on trouve dans le tableau **b**, crée une case contenant **le double de la valeur** dans le tableau **dd.**

11 - Déclaration par compréhension avec indice

```
>>> new = [i*10 for i in range(10)]
>>> new
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Traduction en français : Il faut le lire de la droite vers la gauche ; Pour chaque indice **i** variant de 0 à 9, crée une case contenant **dix fois l'indice** dans le tableau **t**.

12 - Déclaration par compréhension avec condition

On peut également rajouter une condition, qu'on place alors à la fin.

```
>>> t = [12, 8, 1, 3, 3, 17, 8]
>>> modif = [v for v in t if v > 10]
>>> modif
[12, 17]
```

13 – Python : des tableaux dynamiques (append / pop)

Les lists Python ne sont pas réellement des tableaux statiques. Elles possèdent plus transformations possibles : on peut rajouter des cases à la fin ou supprimer des cases à la fin. C'est pour cela qu'on parle de **tableau dynamique**.

```
>>> t = []
>>> t.append("Alice")
>>> t
 ['Alice']

>>> t.append("Bob")
>>> t
  ['Alice', 'Bob']

>>> t.append("Clark")
>>> t
 ['Alice', 'Bob', 'Clark']

>>> t.pop()
 'Clark'

>>> t
 ['Alice', 'Bob']
```

III – Alias et Copie

14 - Alias d'un tableau

La variable d'un tableau contient sa référence-mémoire. On crée juste un alias permettant d'accéder au même contenu lorsqu'on tape **t2 = t.**

```
>>> t = [10, 20, 50]
>>> id(t)
 128
>>> t2 = t
>>> id(t2)
 128              ici   id(t) = id(t2)
```

Ici **t2** et **t** désignent la même zone mémoire (128). Modifier **t** va modifier **t2**, modifier **t2** va modifier **t**.

```
>>> t[0] = 100000
>>> t2
[100000, 20, 50]
```

15 Copie de tableau

Pour avoir une copie indépendante d'un tableau :
`>>> t2 = [v for v in t]`

Cette fois, t2 possède le même contenu que t au début mais les deux variables vont pouvoir suivre chacune leur propre évolution : elles pointent deux zones-mémoires différentes (**id(t)≠ id(t2)**).

On peut aussi utiliser cette fonction (mais en NSI, on utilisera plutôt la création par compréhension).

```
>>> t = [0, 10, 20, 30]
>>> t2 = t.copy()
>>> t2
[0, 10, 20, 30]
```

IV – Fonctions-Raccourci

Pour trouver le plus grand élément d'un tableau **t** :	`vmax = max(t)`
Pour trouver le plus petit élément d'un tableau **t** :	`vmin = min(t)`
Pour trouver la somme des éléments d'un tableau **t** :	`s = sum(t)`
Pour trouver la valeur moyenne des éléments d'un tableau **t** :	`m = sum(t)/len(t)`

V – Matplotlib

Voir le site pour son utilisation. Aucune connaissance exigible mais pratique dans toutes matières où on doit tracer des graphiques parfois (sciences, SES, HG...)