1 Problématique

Construire et remplir un tableau demande plusieurs étapes.

```
1 tab = []
2 for i in range(100):
3       tab.append(i)
4
5 tab2 = [0] * 100
6 for i in range(len(tab2)):
7       tab2[i] = i
```

Code 1 – Construire un tableau

Pourtant Python est un langage de haut niveau et devrait pouvoir faciliter ces constructions.

Comment construire des tableaux en utilisant les propriétés d'un langage de haut niveau ?

2 Construction par compréhension

2.1 Un tableau simple

La méthode ne diffère pas énormément du code 1. La boucle est exécutée directement pendant la construction du tableau.

```
1 tab = [i for i in range(100)]
```

Code 2 – Construction par compréhension

Ce code exécute d'abord la boucle, garde les valeurs en mémoire à chaque tour de boucle (ici i) puis construit le tableau avec ces valeurs. Enfin il affecte ce tableau à la variable *tab*.
Cet ordre à une réelle importance. Il est ainsi possible de construire un tuple par compréhension.

```
1 t = tuple(i for i in range(100))
```

Code 3 – Tuple par compréhension

mettre **tuple** sinon on construit un générateur ; avec une boucle on ne peut pas remplir un tuple

Activité 1 :

1. Construire par compréhension le tableau des carrés des nombres de 0 à 10.
2. Construire par compréhension le tableau des nombres impairs de 0 à 20.
3. Construire par compréhension le tableau des multiples de 3 compris entre 0 et 50.

Christophe Viroulaud

2.2 Tableau de tableaux

Un tableau peut stocker tous types d'objets même d'autres tableaux. Ceci peut s'avérer très utiles pour représenter une matrice mathématique ou le plateau d'un jeu.
Le jeu du *puissance 4* comporte six lignes de sept colonnes. Une représentation en mémoire possible est :

```
1 grille = [[False for col in range(7)] for ligne in range(6)]
```

Code 4 – Puissance 4

Activité 2 : Tester le code 4 sur http://pythontutor.com/ et visualiser la représentation en mémoire.

2.3 Une erreur classique

Il pourrait être tentant de coder la grille de cette manière.

```
1 grille = [[False] * 7] * 6
```

Code 5 – Puissance 4

Activité 3 : Tester le code 5 sur *Pythontutor* et observer la représentation en mémoire.

Il faut rappeler que la variable grille ne contient pas réellement le tableau mais uniquement l'adresse vers celui-ci. Ainsi le même tableau contenant sept valeurs *False* est crée une fois puis pointé six fois. D'une manière générale nous construirons les tableaux par compréhension en utilisant la méthodologie vue en 2.2.

3 Slice (hors programme)

Les *slices* (tranches) sont des expressions qui permettent d'extraire facilement des éléments d'un tableau.

```
1 tab = [0, 1, 2, 3, 4, 5]
2 tranche = tab[1:3] # La variable tranche contient [1, 2]
```

Code 6 – Extraire une tranche

Le code 6 montre la syntaxe d'un *slice*. La variable *tranche* contient les valeurs entre l'indice 1 *inclus* et 3 *exclus*.
Les exemples ci-après montrent la puissance d'un langage de haut-niveau comme Python.

```
1 tab = [0, 1, 2, 3, 4, 5]
2 debut = tab[:3] # [0, 1, 2]
3 fin = tab[3:] # [3, 4, 5]
```