1 Langage de haut niveau

Nous l'avons déjà évoqué, Python est un langage de haut niveau c'est à dire qu'il met à disposition des outils optimisés facilitant la vie du développeur. Par exemple la méthode *len* renvoie la taille de la séquence donnée en paramètre (code 1).

```
>>> t = (1, 4, 8)
>>> len(t)
>>> 3
>>> l = [2, 6, 9, 10]
>>> len(l)
>>> 4
>>> texte = "bonjour"
>>> len(texte)
>>> 7
```

Code 1 – La méthode *len* est utilisable avec plusieurs types de données

Comparativement au langage machine que nous avons utilisé en début d'année nous n'avons par exemple pas besoin de gérer les appels mémoires.
Il est illusoire et de toute façon inutile de connaître toutes les fonctionnalités par cœur. Par contre il faut savoir les retrouver. La documentation est donc un outil précieux. Concentrons-nous sur la documentation des tableaux (les *list* en Python).

https://docs.python.org/fr/3/tutorial/datastructures.html

2 Méthodes courantes

Pour fixer les idées il est utile de visualiser ce qui se passe en mémoire. Le site http://pythontutor.com/ simule l'exécution d'un programme Python. Nous utiliserons ce site pour la suite des activités. Il faut cliquer sur *Start visualizing your code now*.

2.1 Comprendre la documentation

La documentation donne le nom de la méthode et les *paramètres* éventuels.

list.pop([i])
Enlève de la liste l'élément situé à la position indiquée et le renvoie en valeur de retour. Si aucune position n'est spécifiée, a.pop() enlève et renvoie le dernier élément de la liste.

Dans cet exemple, il faut remarquer que :

— La méthode *renvoie* la valeur dans le programme principal. Il est donc possible de la stocker dans une variable.

— Le paramètre *i* est facultatif. La notation par crochets indique ce caractère.

Activité 1 :

1. Créer une liste **l** de cinq entiers.
2. Enlever le dernier élément et le récupérer dans une variable **dernier**.
3. Enlever le premier élément et le récupérer dans une variable **premier**.

Christophe Viroulaud

2.2 Modifier un tableau

Certaines opérations sont très courantes sur un tableau. Python offre des méthodes pour les réaliser.

Activité 2 :

1. Créer un tableau vide nommés **disciplines**.
2. Ajouter (une à une) les noms des disciplines étudiées en première.
3. Trier le tableau par ordre alphabétique.
4. Trouver l'indice de la discipline "NSI".
5. Supprimer une des trois spécialités.

Observation : Il faut remarquer que derrière ces méthodes simples d'utilisation se cachent des algorithmes parfois complexes. Nous y reviendrons plus tard dans l'année.

2.3 Copier un tableau

Activité 3 :

1. Tester le code suivant. Le comportement a déjà été observé dans les cours précédents.

```
a = 3
b = 4
a = b
a = 5
```

2. Tester le code suivant.

```
l1 = [1, 2, 3]
l2 = l1
l1[0] = 4
```

   En première approche nous pouvons expliquer ce comportement ainsi : la variable *l1* ne contient pas le tableau mais *l'adresse mémoire* de ce dernier. La variable *l2 pointe* alors vers le même espace mémoire que *l1*.

3. Dans la documentation, trouver une méthode qui permet de créer une vraie copie de *l1*.

2.4 Itérer sur un tableau

C'est une opération courante sur un tableau. Il existe plusieurs manières de réaliser cette itération.

Activité 4 :

1. Écrire une boucle qui affiche les disciplines entrées dans le tableau *disciplines*.
2. Tester le code suivant. Bien prendre le temps de comprendre cette implémentation.

```
for matiere in disciplines:
        print(matiere, end=" ")
```

3 Des tableaux plus complexes

Nous avons crée des tuples et des tableaux d'entiers (*int*) ou de chaînes de caractère (*string*) mais il est possible de stocker des objets plus complexes. Imaginons un programme qui implémente le jeu *Puissance 4*. Le plateau de jeu (figure 1) est composé de six lignes de sept colonnes.

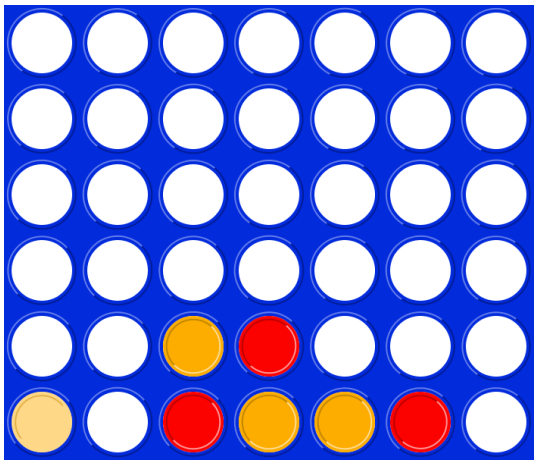

FIGURE 1 – Grille du puissance 4

Chaque ligne peut être représentée par un tableau en mémoire.

```
ligne1 = [0, 0, 0, 0, 0, 0, 0]
ligne2 = [0, 0, 0, 0, 0, 0, 0]
ligne3 = [0, 0, 0, 0, 0, 0, 0]
ligne4 = [0, 0, 0, 0, 0, 0, 0]
ligne5 = [0, 0, 0, 0, 0, 0, 0]
ligne6 = [0, 0, 0, 0, 0, 0, 0]
```

Mais tout comme le stockage des anomalies de température dans le chapitre précédent, l'accès aux différentes lignes peut vite s'avérer fastidieux. Il est possible de créer un tableau de tableau comme le montre le code suivant.

```
grille = [ [0, 0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 0, 0, 0],
           [0, 0, 0, 0, 0, 0, 0] ]
```

Il faut noter que l'interpréteur Python ne comprend pas ici les retours à la ligne comme une indentation. Cela facilite la visualisation de la grille.
L'accès au deuxième trou de la sixième ligne s'effectue de la manière suivante.

```
# Rappel: la numérotation commence par zéro
trou = grille[5][1]
```

Activité 5 :

1. Créer la variable *grille*.
2. Modifier la grille pour représenter la situation de la figure 1. Les pions jaunes seront représentés par le chiffre 1, les rouges par le chiffre 2.

Christophe Viroulaud