Cours	**Cours**	
BTS SNIR	**Python**	
2 ème année	**Les listes**	

Dans cette leçon, nous allons découvrir un premier type de données composites Python : les listes. Nous allons comprendre l'intérêt de ce type de données et apprendre à les manipuler.

Présentation des listes Python

Jusqu'à présent, nous n'avons stocké qu'une seule valeur à la fois dans nos variables. Les listes sont un type de données très particulier au sens où elles représentent des données composées ou combinées. Une liste est en effet par définition composée d'une suite de valeur ou d'éléments.

Pour définir une nouvelle liste en Python, on va devoir utiliser une paire de crochets `[ ]`. Nous allons placer les différents éléments de notre liste dans ces crochets en les séparant par des virgules. On peut par exemple créer une liste de 5 éléments et la placer dans une variable `liste` comme ceci :

```
>>> liste1 = [2, 4, 8, 16, 32]
>>> liste1
[2, 4, 8, 16, 32]
```

Notre liste est ici composée de 5 valeurs de type numérique. On va pouvoir stocker tous types de valeurs dans une liste, comme des chaines de caractères par exemple :

```
>>> liste2 = ["Pierre", "Mathilde", "Florian", "Thomas"]
>>> liste2
['Pierre', 'Mathilde', 'Florian', 'Thomas']
```

De plus, vous devez savoir que tous les éléments d'une liste n'ont pas à être du même type, on va très bien pouvoir créer des listes composées de nombres, chaines et booléens par exemple :

```
>>> liste3 = ["Pierre", 29, True]
>>> liste3
['Pierre', 29, True]
```

Note : Les listes doivent vous faire penser à ce qu'on appelle communément dans d'autres langages des tableaux. En effet, les listes Python sont très proches des tableaux (numérotés) qu'on peut retrouver dans de nombreux autres langages.

Récupérer une ou plusieurs valeurs dans une liste

Les listes Python sont par défaut indexées ou indicées. Cela signifie que chaque valeur d'une liste est lié à un indice qu'on va pouvoir utiliser pour récupérer cette valeur en particulier.

Les listes possèdent des indices numériques qui commencent à 0. La première valeur d'une liste possède donc toujours l'indice 0, la deuxième valeur l'indice 1, la troisième valeur l'indice 2 et etc.

Pour récupérer une valeur en particulier dans une liste, on va devoir préciser le nom de la liste suivi de l'indice de cette valeur entre crochets. Notez que les indices négatifs sont acceptés; dans ce cas on partira de la fin de la liste (l'indice -1 correspond au dernier élément, -2 à l'avant dernier et etc.).

```
>>> prenoms = ["Pierre", "Mathilde", "Florian", "Thomas"]
>>> prenoms[0]
'Pierre'
>>> prenoms[2]
'Florian'
>>> prenoms[-1]
'Thomas'
```

On va également pouvoir récupérer une tranche de valeurs dans une liste, c'est-à-dire un ensemble de valeurs qui se suivent. Pour cela, on utilisera le symbole : entre les crochets avec 0, 1 ou 2 indices autour.

Si on utilise : sans indice, alors une copie superficielle de la liste sera renvoyée. Si on mentionne un indice avant : mais pas d'indice après, alors une copie superficielle partielle de la liste de départ sera renvoyée, en commençant à copier à partir de l'indice donné. Si au contraire on mentionne un indice après : mais pas d'indice avant, une copie superficielle partielle de la liste de départ sera renvoyée qui commence au début de la liste et jusqu'à l'indice donné. Enfin, si deux indice sont mentionnés de part et d'autre de :, la tranche de valeurs correspondant à ces indices sera renvoyée.

```
>>> prenoms = ["Pierre", "Mathilde", "Florian", "Thomas"]
>>> prenoms[0:2]
['Pierre', 'Mathilde']
>>> prenoms[:2]
['Pierre', 'Mathilde']
>>> prenoms[2:]
['Florian', 'Thomas']
>>> prenoms[:]
['Pierre', 'Mathilde', 'Florian', 'Thomas']
```

Vous devez également savoir que ce qu'on a vu jusqu'ici sur les listes s'applique également aux chaines de caractères. Les chaînes de caractères peuvent en effet également être indexées, ce qui signifie qu'on peut accéder aux caractères par leur position). Cela est logique après tout : les chaines de caractères sont des "séquences" de caractères tandis que les listes sont des "séquences" de valeurs.

Comme pour les listes, le premier caractère d'une chaîne possède l'indice 0, le deuxième l'indice 1 et etc; On va également pouvoir utiliser des indices négatifs et récupérer des tranches avec :.

```
>>> prenom = "Mon prénom est Pierre"
>>> prenom[2]
'n'
>>> prenom[4]
'p'
>>> prenom[4:10]
'prénom'
>>> prenom[:10]
'Mon prénom'
>>> prenom[10:]
' est Pierre'
>>> prenom[:]
'Mon prénom est Pierre'
```

Notez qu'il n'existe pas de type distinct pour les caractères en Python : un caractère est simplement une chaîne de longueur 1.

Ajouter, supprimer, modifier des éléments d'une liste

A la différence des types de données simples comme les chaines qui sont immuables, les listes sont un type de données altérable ce qui signifie qu'on va pouvoir altérer leur structure ou modifier leur contenu en ajoutant, supprimant ou remplaçant des valeurs.

En effet, vous devez bien comprendre qu'une fois qu'on définit une valeur "chaine de caractères" par exemple, celle-ci ne peut plus être modifiée par la suite. Les seules opération qu'on va pouvoir faire vont être de créer une nouvelle chaine en concaténant deux chaines d'origine (qui une nouvelle fois ne seront pas modifiées) ou de remplacer une chaine par une autre valeur en affectant une nouvelle valeur dans une variable (ce qui a pour effet d'écraser la chaine de départ).

Au contraire des chaines, on va tout à fait pouvoir ajouter, modifier ou supprimer des valeurs dans une liste. Pour cela, on va mentionner le nom de notre liste avec l'indice de la ou des éléments à ajouter / modifier et leur affecter une nouvelle valeur. Les affectations de tranches sont possibles :

```
>>> prenoms = ["Pierre", "Mathilde", "Florian", "Thomas"]
>>> prenoms[2]
'Florian'
>>> prenoms[2] = "Victor"
>>> prenoms
['Pierre', 'Mathilde', 'Victor', 'Thomas']
>>> prenoms[2:] = []
>>> prenoms
['Pierre', 'Mathilde']
>>> prenoms[2:4] = ["Julia", "Chloé", "Axelle"]
>>> prenoms
['Pierre', 'Mathilde', 'Julia', 'Chloé', 'Axelle']
>>> prenoms[:] = []
>>> prenoms
[]
```

Notez finalement également qu'on va aussi pouvoir utiliser les opérateurs de concaténation et de répétition avec des listes :

```
>>> prenoms = ["Pierre", "Mathilde"]
>>> ages = [29, 27]
>>> info = prenoms + ages #Concaténation
>>> info
['Pierre', 'Mathilde', 29, 27]
>>> prenoms3 = prenoms * 3 #Répétition
>>> prenoms3
['Pierre', 'Mathilde', 'Pierre', 'Mathilde', 'Pierre', 'Mathilde']
```