Listes en Python

Une liste est une structure permettant de stocker des éléments sans devoir forcément créer de variable au préalable.

```
# On veut faire une liste des chiffres 8,4,2,9
liste = [1,2,3,4]
```

Cette structure est composée de "deux parties" : la partie valeurs et la partie indices.

valeurs	indices
8	0
4	1
2	2
9	3

Un indice correspond à une position d'une valeur dans la liste.

> **Danger**
>
> Attention ! Les indices commencent à 0, elle correspond à la position 1.

Accéder à des éléments de listes

On peut accéder aux éléments d'une liste à l'aide de la structure des "crochets".

Par exemple, pour la liste précédente, on souhaite afficher les valeurs aux indices 1 et 3.

```
print(liste[1])
print(liste[3])
```

On peut afficher tous les éléments d'une liste à l'aide des boucles, la plus pratique étant la boucle for comme elle évolue dans une séquence.
On peut `itérer` sur les indices et les valeurs d'une boucle.

> **Danger**
>
> Il faut choisir avec parcimonie sur quelle propriété de la liste itérer. Si l'on cherche à réaliser des traitements sur les données de la liste, on itère sur les valeurs. Dans un cas où l'on cherche à inverser des valeurs, ou renvoyer des

indices précis, on itèrera sur les indices de la liste.

Exemple :

```
# Pour chacun des éléments dans la liste
for elt in liste:
    # Afficher l'élément (qui correspond à la valeur dans la liste)
    print(elt)

# Afficher les indices des valeurs supérieures à 10 avec leur valeurs
for i in range(len(liste)):
    # Si la valeur à l'indice i est supérieure à 10
    if liste[i] > 10:
        print(i, liste[i])
```

Modification de valeurs

On peut modifier les valeurs dans une liste.
Une liste est dite `non-mutable`, on peut donc en modifier les éléments la composant.

On peut réaliser cela grâce aux indices et la structures de crochets:

```
liste = [1,2,6,4]
liste[2] = 3

# Liste vaudra ainsi [1,2,3,4]

# Echanger deux valeurs aux indices 2 et 3 dans une liste:
temporaire = liste[2]
liste[2] = liste[3]
liste[3] = temporaire
```

Ajouter des valeurs

Il existe deux manières d'ajouter des valeurs dans une liste :

La concaténation

De la même manière que les chaînes de caractères, on peut ajouter des éléments à la liste, au début ou à la fin suivant où l'on positionne les variables autour du `+`.

> **Danger**
>
> Attention, pour ajouter un élément dans une liste, il faut au préalable le placer dans une liste car on ne peut pas concaténer une liste et une valeur.

Exemple:

```
def inverser_liste(liste:list)->list:
# Inverser les éléments d'une liste
a_remplir = []
for elt in liste:
    # On concatène en n'oubliant pas de mettre l'élément dans une liste
    a_remplir = a_remplir + [elt]
return a_remplir
```

La méthode append

Il existe ce que l'on appelle des `méthodes`. Une méthode est une fonction associée à un type.

La méthode append permet d'ajouter un élément à la fin de la liste.
Elle s'utilise ainsi:

```
def creer_liste_jusque_n(n:int)-> list:
    liste_retour = []
    for i in range(n+1):
        liste_retour.append(n)
    return liste_retour
```

Supprimer des valeurs

Il existe plusieurs manières de retirer une valeur dans une liste:

Suppression de valeurs dans une liste Python

- **`remove(valeur)`** : Supprime la première occurrence de la valeur spécifiée.
  Exemple : `liste.remove(5)`
- **`pop(index)`** : Supprime l'élément à l'indice donné et le retourne (sans paramètres, elle supprime le dernier).
  Exemple : `liste.pop(2)`
- **`del liste[index]`** : Supprime l'élément à l'indice donné sans le retourner.
  Exemple : `del liste[1]`