Chapitre 1 Généralités sur les suites

Fiche 6 : Les listes en Python

Une liste en Python est une succession finie ordonnée de nombres ou de caractères.

Ce n'est donc pas une suite qui est infinie mais plutôt les premiers termes d'une suite.

En Python, on déclare une liste avec les crochets **[]**, les éléments sont séparés par une virgule **,** .

Exemple :

```
>>> #liste des nombres nuturelles
>>> nombre=[12,78,79,15,14,11,187,15]
```

```
>>> #creation d'une liste vide
>>> liste1=[]
```

1. Générer une liste

Génération par ajouts successifs : On utilise la fonction **append()** qui ajoute un élément en fin de liste.

```
L=[5]
for i in range(6,13):
    L.append(i)
```

Génération par compréhension :
liste=[expression for x in Liste] .
Il faut que la liste soit déjà créée.

Exemple liste en compréhension sans condition:

```
>>> #Creation liste de nombres avec liste par compréhension
>>> list_A=[i for i in range(10)]
>>> list_A #afficher la liste
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list_A=[i * i for i in range(1,10)]#liste des carrés
>>> list_A #afficher la liste
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Génération par concaténation :

On peut fusionner deux listes avec l'opérateur +

Exemple :

```
>>> #Concaténation de deux listes
>>> list_A=[1,2,3]
>>> list_B=[4,5,6,7]
>>> list_C=list_A + list_B
>>> print(list_C)
[1, 2, 3, 4, 5, 6, 7]
>>> #ajouter un élément à la list_C
>>> list_C=list_C+ [56]
>>> print(list_C)
[1, 2, 3, 4, 5, 6, 7, 56]
```

2. Manipuler des éléments d'une liste

On peut accéder aux éléments d'une liste en mettant l'indice souhaité entre les crochet :

Exemple :

```
>>> nombre=[12,78,79,15,14,11,187,15]#creation d'une liste de nombres
>>> print(nombre[0]) #récupérer l'élément d'indice 0
12
>>> print(nombre[-1]) #récupérer le dernier élément
15
>>> print(nombre[2:6])#récupérer la sous-liste de l'indice 2 à l'indice 6-1
[79, 15, 14, 11]
```

Pour modifier un élément, on affecte une nouvelle valeur entre crochets à L[i]

Pour insérer un élément à l'indice i, on affecte la nouvelle valeur entre crochets à L[i :i]

Pour supprimer un élément d'indice i, on affecte [ ] à L[i] ou on utilise la fonction del(L[i])

3. Parcourir une liste

Le parcours d'une liste c'est l'accès séquentiel (1 par 1) à ses éléments du début jusqu'à la fin de la liste.

Les fonctions **range(), len()** et la boucle **for** permettent le parcours de liste.

Exemple 1:

```
>>> #parcours les éléments d'une liste
>>> ville=["Laayoune","Fes","Casablanca","Tanger","Agadir"]
>>> nb_elements=len(ville)
>>> for i in range(nb_elements):
        print("l'élement d'indice :",i," est :",ville[i])

l'élement d'indice : 0  est : Laayoune
l'élement d'indice : 1  est : Fes
l'élement d'indice : 2  est : Casablanca
l'élement d'indice : 3  est : Tanger
l'élement d'indice : 4  est : Agadir
```

Exemple 2:

```
>>> #parcours les éléments d'une liste element par element
>>> ville=["Laayoune","Fes","Casablanca","Tanger","Agadir"]
>>> for elem in ville:
        print("l'élement est :",elem)

l'élement est : Laayoune
l'élement est : Fes
l'élement est : Casablanca
l'élement est : Tanger
l'élement est : Agadir
```

Remarque : Itérer les éléments d'une liste, c'est obtenir tous les éléments d'une liste à l'aide de l'instruction x for x in L .

Les exercices du livre : Ex 102 p 34, 124 p 37