Python-ECG1-08-Listes1-cor

October 28, 2021

1 TP8 - Introduction aux listes

1.1 I) Créer une liste explicitement

Pour créer une liste, on écrit ses éléments **entre crochets**, séparés par des vrigules.

[1]:
```
L1 = [1,2,3,4]
L1
```

[1]:
```
[1, 2, 3, 4]
```

1) Créer une liste L2 contenant les éléments 4, 7, 12, 11 et 8 dans cet ordre.

[2]:
```
L2 = [4,7,12,11,8]
L2
```

[2]:
```
[4, 7, 12, 11, 8]
```

La commande **range(n,m)** utilisée dans les boucles for permet de créer des listes d'entiers consécutifs.

Attention, dans range(n,m) les éléments vont de n à m-1.

[3]:
```
L3 = range(2,8)
L3
```

[3]:
```
range(2, 8)
```

[4]:
```
# Pour afficher les éléments d'un range, il faut faire une petite manipulation
list(L3)
```

[4]:
```
[2, 3, 4, 5, 6, 7]
```

On peut également créer des listes d'entiers séparés d'un pas constant (suite arithmétique) avec **range(n,m,pas)**.

2) Créer la liste L4 = [4,6,8,10,...,20] avec un range.

[5]:
```
L4 = range(4,21,2)
```

1.2 II) Récupérer/modifier les éléments d'une liste.

Les éléments d'une liste sont numérotées avec une numérotation **commençant à zéro**.

Ainsi, le premier élément est l'élément numéro 0, le second est l'élément numéro 1, etc.

Pour obtenir l'élément numéro i d'une liste L, on utilse **L[i]**.

3) Afficher le premier élément de L3.

[6]:
```
L3[0]
```

[6]:
```
2
```

Il existe aussi une **numérotation négative**, en partant de la fin.

Le dernier élément est celui de numéro -1, l'avant-dernier est celui de numéro -2, etc.

4) Afficher le quatrième élément de L2 (le 11) de deux façons différentes.

[7]:
```
print(L2[3])
print(L2[-2])
```

```
11
11
```

On peut alors modifier tout élément d'une liste L en écrivant **L[i] = valeur**.

5) Modifier le 7 de L2 en un -1. Vérifier en affichant L2.

[8]:
```
L2[1] = -1
L2
```

[8]:
```
[4, -1, 12, 11, 8]
```

Il est également possible d'accéderà une sous-liste d'une liste L :

- sous-liste des éléments numérotés de 0 à m-1 : **L[:m]**
- sous-liste des éléments numérotés de n à m-1 : **L[n:m]**
- sous-liste des éléments numérotés de n à la fin : **L[n:]**
- sous-liste des éléments numérotés de n à m-1 avec un pas : **L[n:m:pas]**

6) Afficher la liste [3,4] à partir de L1.

[9]:
```
L1[2:] # ou L2[2:4]
```

[9]:
```
[3, 4]
```

1.3 III) Ajouter/supprimer des éléments

Pour ajouter un élément x à la fin d'une liste L on utilise : **L.append(x)**

Pour **concaténer** deux listes L1 et L2, on utilise **L1+L2**.

Pour répéter plusieurs une liste, on utilise **L*n**

7) Ajouter un 13 à la fin de L2.

[10]:
```
L2.append(13)
L2
```

[10]:
```
[4, -1, 12, 11, 8, 13]
```

8) Concaténer L1 et L2 dans cet ordre.

[11]:
```
L1+L2
```

[11]:
```
[1, 2, 3, 4, 4, -1, 12, 11, 8, 13]
```

9) En utilisant une concaténation, ajouter un 0 au début de L1.

[12]:
```
L1 = [0] + L1
L1
```

[12]:
```
[0, 1, 2, 3, 4]
```

10) Créer alors la liste [0,1,2,3,4,0,1,2,3,4] à partir de L1.

[13]:
```
L1*2
```

[13]:
```
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4]
```

Pour supprimer un élément d'une liste L, on utilise **del L[i]**.

11) Supprimer le 13 dans L2.

[14]:
```
del L2[-1]
L2
```

[14]:
```
[4, -1, 12, 11, 8]
```

1.4 IV) Autres commandes

La longueur d'une liste L (nombre d'éléments) est obtenue avec **len(L)**.

On peut aussi tester l'appartenance d'un élément x à une liste L avec **x in L**. Ceci renvoie True si x est dans L et False sinon (booléen).

12) Afficher la longueur de L2.

[15]:
```
len(L2)
```

[15]:
```
5
```

13) Vérifier que 13 n'est plus dans L2.

[16]:
```
13 in L2
```

[16]:
```
False
```

1.5 V) Créer une liste avec un for.

1.5.1 a) En compréhension (suites explicites).

La commande [**element_k for k in range(n,m)**] permet de créer la liste des terme de la forme element_k, pour k allant de n à m-1.

14) Créer la liste L4 = [4,6,8,10,...,20] avec cette méthode.

[17]:
```
L4 = [2*k for k in range(2,11)]
L4
```

[17]:
```
[4, 6, 8, 10, 12, 14, 16, 18, 20]
```

1.5.2 b) Suites récurrentes

15) On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = 2u_n - n \end{cases}$$

Écrire un programme qui crée la liste $L = [u_0, u_1, u_2, \ldots, u_{20}]$.

[18]:
```
u = 2
L = [2]
for k in range(20):
    u = 2*u - k
    L.append(u)
print(L)
```

```
[2, 4, 7, 12, 21, 38, 71, 136, 265, 522, 1035, 2060, 4109, 8206, 16399, 32784,
65553, 131090, 262163, 524308, 1048597]
```