P5

LES LISTES

I Définition

Une **liste** est une une structure de données *ordonnée* et *modifiable*. C'est une séquence d'éléments séparés par des virgules et entourés par des crochets :

Entrée [1]:
```
L = [42, 3.14, -1, 0, 7]
```

La **longueur** de la liste `L`, c'est-à-dire le nombre d'éléments qu'elle contient, est obtenue avec `len(L)` :

Entrée [2]:
```
len(L)
```

Out [2]: 5

II Accéder aux éléments

II.1 Position d'un élément

Chaque élément de la liste (et plus généralement d'une séquence) est *indicé* par un entier indiquant sa position :

Élément	42	3.14	−1	0	7
Indice positif	0	1	2	3	4
Indice négatif	-5	-4	-3	-2	-1

!! Attention !! l'indice positif du premier élément est 0 et non 1.
Le dernier élément est lui d'indice $len(L) - 1$.

Pour accéder à l'élément d'indice `i`, on utilise `L[i]` :

Entrée [3]:
```
L[2]
```

Out [3]: -1

!! Attention !! si l'indice est supérieur ou égale à la longueur de la liste, Python produit une erreur :

Entrée [4]:
```
L[len(L)]
```

Out [4]:
```
Traceback (most recent call last):
  File "<input>", line 1, in <module>
IndexError: list index out of range
```

Retenez bien cette erreur, vous l'aurez souvent !

II.2 Tranches

Pour accéder aux éléments d'indice i pour $d \leq i < f$, on utilise la syntaxe `L[d:f]` : on réalise ainsi une **tranche**. L'un ou l'autre des deux indices peut être omis, et même les deux (dans ce cas, `d` vaut 0 et `f` vaut la longueur de la liste).

Entrée [5]:
```
L[1:4]
```

Out [5]:
```
[3.14 , -1, 0]
```

Entrée [6]:
```
L[1:]
```

Out [6]:
```
[3.14, -1, 0, 7]
```

Entrée [7]:
```
L[:3]
```

Out [7]:
```
[42, 3.14, -1]
```

II.3 Savoir si un élément x est dans L

Le booléen `x in L` renvoie `True` si x est dans L et `False` sinon.

Entrée [8]:
```
5 in L
```

Out [8]:
```
False
```

Entrée [9]:
```
0 in L
```

Out [9]:
```
True
```

II.4 Parcourir une liste

Il existe deux façons de parcourir une liste :

- en itérant sur les indices à l'aide de la fonction `range` :

Entrée [10]:
```
# L liste donnée
# Affichons tous les éléments de L séparemment
for i in range(len(L)): # i parcourt tous les indices possibles
    x = L[i] # élément d'indice i
    print(x)
```

!! Attention !! C'est LA méthode à utiliser si on a besoin de connaître l'indice.

- en itérant directement sur les éléments de la liste :

Entrée [11]:
```
# L liste donnée
# Affichons tous les éléments de L séparemment
for x in L: # x parcourt directement les éléments de L
    print(x)
```

!! Attention !! Dans ce cas, on ne connaît pas l'indice (la position de x) !

III Créer ou modifier une liste

III.1 Créer une liste en une instruction

On peut créer une liste par :

- **extension** en énumérant ses éléments entre crochets et en les séparant par des virgules.

Entrée [12]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Remarque : `[]` crée une liste vide.

- **compréhension** en utilisant la syntaxe [*expression* `for e in` *séquence*]

Entrée [13]:
```
[ k**2  for  k  in range(10) ]
```

Out [13]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

III.2 Opérations sur les listes

- **concaténation** en utilisant l'opérateur `+`

Entrée [14]:
```
[2, 4, 6, 8] + [1, 3, 5, 7]
```

Out [14]:
```
[2, 4, 6, 8, 1, 3, 5, 7]
```

- **répétition** en utilisant l'opérateur `*`

Entrée [15]:
```
[0]*10
```

Out [15]:
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- **ajout d'un élément** en fin d'une liste déjà existante à l'aide de la *méthode* append

Entrée [16]:
```
L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
L.append(11)
```

Out [16]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

- **suppression de l'élément en position i** à l'aide de l'instruction `del L[i]`

Entrée [17]:
```
L = [5, 6, -1, 0, 2]
del L[3]
```

Out [17]:
```
L = [5, 6, -1, 2]
```

III.3 Modifier un élément

Une liste étant *muable*, il est possible de modifier un ou plusieurs de ses éléments.
L'instruction `L[i] = x` remplace, dans la liste `L`, l'élément d'indice `i` par la valeur `x`

Entrée [18]:
```
L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
L[3] = 20
L
```

Out [18]:
```
[0, 1, 2, 20, 4, 5, 6, 7, 8, 9, 10]
```

IV Exercice classique

On considère la suite récurrente $(u_n)_{n\in\mathbb{N}}$ 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 $L = [u_0, u_1, u_2, \ldots, u_{20}]$.

Méthode 1 : avec append - ajoute successifs d'éléments

```
u = 2
L = [u]
for k in range(20):
    u = 2*u - k
    L.append(u)        # on ajoute le nouveau u à la fin de L.
print(L)
```

Méthode 2 : modifications successives d'éléments

```
L = [0]*21    # liste [0,0,..,0] de taille 21 (taille voulue)
L[0] = 2
for i in range(20):
    L[i+1] = 2*L[i] - i
print(L)
```

ou

```
for i in range(1,21):
    L[i] = 2*L[i-1] - (i-1)
```

$L[i] = u_i$
$L[i+1] = u_{i+1}$

La boucle for sert à modifier L[1] puis L[2], etc jusqu'à L[20].