

Listes, boucles **for**

Création et manipulation de listes

Une liste (type "**list**") est une variable qui prend la forme d'une succession de valeurs.

📖 Créer une liste "à la main"

`L = [valeur1,valeur2,valeur3,...]`

✎ Exercice 1

Dans la console, définir la liste `L = [2,4,0,-2,3.5]`.

Que renvoient les instructions suivantes ?

`type(L) :`

`L[0] :` `L[1] :` `L[4] :`

Conclusion : À quoi sert l'instruction `L[i]` si `i` est un entier ?

.....
.....

⚠ Attention !

On notera bien que le premier élément d'une liste `L` est `L[0]` !

On peut également utiliser `L[i]` pour modifier les valeurs au sein d'une liste.

👉 Exemple

L'instruction `L[3] = -2` remplace le 4ème élément de la liste `L` par `-2`.

✎ Exercice 2

Modifier les valeurs de la liste `M` pour qu'elle devienne `[1,2,3,4,5]`.

```
>>> M = [1,-1,3,3,5]
>>>
>>>
>>> M
```

💬 Remarque 1

Bonus : • `len(L)` renvoie le nombre d'éléments ("length") de la liste `L`.

• `L[-1]` accède au dernier élément de la liste `L`.

• `L[i:j]` renvoie la "sous-liste" de `L` constituée des éléments de l'indice `i` à `j-1`.

Créer une liste d'entiers consécutifs avec **range**

📖 Instruction **range**

• `range(n)` crée une liste contenant les entiers successifs de `0` à `n-1`.

Exemple : `X = range(4)` stocke la liste `[0, 1, 2, 3]` dans la variable `X`.

• `range(a,b)` crée une liste contenant les entiers successifs de `a` à `b-1`.

(`range(n)` a donc le même effet que `range(0,n)`)

Exemple : `Y = range(5,10)` stocke la liste `[5, 6, 7, 8, 9]` dans la variable `Y`.

• `range(a,b,r)` crée une liste en progression arithmétique de raison `r`, de premier terme `a` et dont le dernier terme ne dépasse et n'atteint pas `b`.

(`range(a,b)` a donc le même effet que `range(a,b,1)`)

Autrement dit, on crée la liste : `[a,a+r,a+2r,...]` qui s'arrête juste avant `b`.

Exemple : `Z = range(4,13,3)` stocke la liste `[4, 7, 10]` dans la variable `Z`.

✎ Exercice 3

Deviner les listes produites par les instructions suivantes :

`range(8) :`

`range(1,11) :`

`range(0,17,5) :`

Pour vérifier vos réponses, on pourra taper dans la console :

`list(range(8))`, `list(range(1,11))`, etc...

(Car en fait, les variables produites par l'instruction `range` ne sont pas tout à fait du type "**list**", mais c'est un détail)

✎ Exercice 4

• Quelle instruction (avec `range`) permet de créer la liste

`L1 = [-5, -4, -3, -2, -1]` ?

• Quelle instruction (avec `range`) permet de créer la liste

`L2 = [5,7,9,11,13,15]` ?

Créer une liste plus générale

Syntaxe `[f(i) for ...]` (Très pratique!)

Si A est une liste (ou un tableau...), l'instruction `[f(i) for i in A]` crée une liste contenant les valeurs $f(i)$ pour i parcourant A .

Exemple : L'instruction `L = [1/i for i in range(1,11)]` stocke la liste `[1, $\frac{1}{2}$, $\frac{1}{3}$, ..., $\frac{1}{10}$]` dans la variable L .

Exercice 5

Quelle instruction permet de créer facilement la liste `[2^0 , 2^1 , ..., 2^{15}]` ?
(tester dans la console)

.....

Exercice 6

Dans l'éditeur de texte, créer une fonction nommée `ListeCarre` qui prend en entrée un entier n non nul et renvoie la liste `[1^2 , 2^2 , 3^2 , ..., n^2]`.

Tester ensuite cette fonction dans la console en appelant par exemple `ListeCarre(4)`, `ListeCarre(5)`, etc...

Instructions itératives : boucle `for`

Pour répéter une série d'instructions un certain nombre de fois, pour chaque valeur de k parcourant une liste L (ou un tableau...), on utilise une boucle `for`.

Boucle For

```
for k in L :  
    instructions
```

Remarques 2

- On notera à nouveau l'importance des deux points (`:`) et de l'indentation.
- La variable k est "muette" : on peut lui donner le nom que l'on souhaite. Elle n'a pas besoin d'être introduite en avance.

Exercice 7

Analyser le programme et prévoir la valeur finale affichée.

```
P = 1  
for k in range(2,6) :  
    P = P * k  
print('Valeur de P : ',P)
```

- Avant la boucle : $P = \dots$
- 1er passage : $k = \dots$, $P = \dots$
- 2ème passage : $k = \dots$, $P = \dots$
- 3ème passage : $k = \dots$, $P = \dots$
- 4ème passage : $k = \dots$, $P = \dots$

Exercice 8

Deviner la valeur affichée par le script suivant. Vérifier en l'exécutant.

```
S=0  
for i in range(1,5) :  
    S=S+2*i  
print(S)
```

La valeur affichée est :

En pratique, les boucles `for` nous serviront surtout à calculer des sommes, des produits, ou les termes successifs d'une suite : ce sera l'objet des TP suivants.