Les listes

1. Définition

Une **liste** est une structure de données qui contient une série de valeurs. Python autorise la construction de liste contenant des valeurs de types différents (par exemple entier et chaîne de caractères), ce qui leur confère une grande flexibilité. Une liste est déclarée par une série de valeurs (n'oubliez pas les guillemets, simples ou doubles, s'il s'agit de chaînes de caractères) séparées par des **virgules**, et le tout encadré par des **crochets**.

Exemples

```
>>> animaux = ['girafe', 'tigre', 'singe', 'souris']
>>> tailles = [5, 2.5, 1.75, 0.15]
>>> mixte = ['girafe', 5, 'souris', 0.15]
>>> animaux
['girafe', 'tigre', 'singe', 'souris']
>>> tailles
[5, 2.5, 1.75, 0.15]
>>> mixte
['girafe', 5, 'souris', 0.15]
```

Lorsque l'on affiche une liste, Python la restitue telle qu'elle a été saisie.

2. Utilisation

Un des gros avantages d'une liste est que vous pouvez appeler ses éléments par leur position de la même manière que pour une chaîne de caractères (voir TP2 1ers programmes – chaînes de carctère).
Ce numéro est appelé **indice** (ou *index*) de la liste.

```
liste  : ['girafe', 'tigre', 'singe', 'souris']
indice :        0        1        2         3
```

Exemple

```
>>> animaux = ['girafe','tigre','singe','souris']
>>> animaux[0]
'girafe'
>>> animaux[1]
'tigre'
>>> animaux[3]
'souris'
```

Remarque si on appelle l'élément d'indice 4 de notre liste, que peut-il se passer ?

(a) Opération sur les listes

Tout comme les chaînes de caractères, les listes supportent l'opérateur + de concaténation, ainsi que l'opérateur * pour la duplication.

Exemples

```
>>> ani1 = ['girafe', 'tigre']
>>> ani2 = ['singe', 'souris']
>>> ani1 + ani2
['girafe', 'tigre', 'singe', 'souris']
>>> ani1 * 3
['girafe', 'tigre', 'girafe', 'tigre', 'girafe', 'tigre']
```

Ajout d'un élément à une liste

Pour ce faire nous avons deux méthodes :

- On crée une liste vide

```
>>> a = []
>>> a
[]
```

- puis :

Ajout d'un élément avec l'opérateur + de concaténation	Ajout d'un élément avec la méthode **`.append()`**

`>>> a = a + [15]`
`>>> a`
`[15]`
`>>> a = a + [-5]`
`>>> a`
`[15, -5]`
	`>>> a.append(13)`
	`>>> a`
	`[15, -5, 13]`
	`>>> a.append(-3)`
	`>>> a`
	`[15, -5, 13, -3]`

(b) Listes et chaînes de caractères

On parcourt les listes comme les chaînes de caractères

i. Indiçage négatif

Comme pour les chaînes de caractères, on peut avoir un indiçage négatif (voir TP2).

Exemple

```
liste          : ['girafe', 'tigre', 'singe', 'souris']
indice positif :         0         1         2         3
indice négatif :        -4        -3        -2        -1
```

ii. Slicing

Comme pour les chaînes de caractères, on peut extraire une portion de la liste (voit TP2).

Prévoir les sorties **`Out[]`**

Exemples

```
In [25]: animaux = ['girafe','tigre','singe','souris','chien']

In [26]: animaux[0:2]
Out[26]:

In [27]: animaux[:3]
Out[27]:

In [28]: animaux[:]
Out[28]:

In [29]: animaux[-2:]
Out[29]:

In [30]: animaux[:-1]
Out[30]:

In [31]: animaux[1:2:4]
Out[31]:


In [32]: x = [0,1,2,3,4,5,6,7,8,9]

In [33]: x[::1]
Out[33]:

In [34]: x[::3]
Out[34]:

In [35]: x[::-1]
Out[35]:

In [36]: x[11]
```

iii. Fonction len()

Tout comme pour les chaînes de caractère, l'instruction len() vous permet de connaître la longueur d'une liste, c'est-à-dire le nombre d'éléments que contient la liste.

Exemple

```
>>> animaux = ['girafe', 'tigre', 'singe', 'souris']
>>> len(animaux)
4
>>> len([1, 2, 3, 4, 5, 6, 7, 8])
8
```

3. Les fonctions range() et list()

La fonction **`range()`** est une fonction spéciale en Python qui génère des nombres entiers compris dans un intervalle. Lorsqu'elle est utilisée en combinaison avec la fonction **`list()`**, on obtient une liste d'entiers.

Exemple

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

La commande **`list(range(n,p))`** génère une liste contenant tous les nombres entiers de **n inclus** à **p exclus**.

Remarque **`list(range(0,10))`** renvoie la même sortie que **`list(range(10))`**

Exemple

Expliciter le rôle de chaque argument utilisé (nombres utilisés par la fonction range()).

`In [1]: list(range(0,5))`
`Out[1]: [0, 1, 2, 3, 4]`
 	
`In [2]: list(range(15,20))`
`Out[2]: [15, 16, 17, 18, 19]`
 	
`In [3]: list(range(0,1000,200))`
`Out[3]: [0, 200, 400, 600, 800]`
 	
`In [4]: list(range(2,-2,-1))`
`Out[4]: [2, 1, 0, -1]`
	commentaires

4. Listes de listes

Il est tout à fait possible de construire des listes de listes.

Cette fonctionnalité peut parfois être très pratique.

Exemple prévoir la sortie **`Out[13]`**.

```
In [9]: enclos1=['girafe',4]

In [10]: enclos2=['singe',2]

In [11]: enclos3=['tigre',5]

In [12]: zoo=[enclos1,enclos2,enclos3]

In [13]: zoo
```

`out[13] :`

5. Applications

(a) Jours de la semaine

i. Constituez une liste semaine contenant les 7 jours de la semaine.

ii. À partir de cette liste, proposez deux lignes de commande différentes permettant de récupérer :

- seulement les 5 premiers jours de la semaine
- ceux du week-end
- le dernier jour de la semaine.

iii. Inversez les jours de la semaine en une commande

(b) Saisons

i. Créez 4 listes hiver, printemps, été et automne contenant les mois correspondants à ces saisons.

ii. Créez ensuite une liste saisons contenant les listes hiver, printemps, été et automne.

iii. Prévoyez ce que renvoient les instructions suivantes, puis vérifiez-le dans l'interpréteur.

```
1. saisons[2]
2. saisons[1][0]
3. saisons[1:2]
4. saisons[:][1].
```

Comment expliquez ce dernier résultat ?

(c) Table de multiplication par 9

Affichez la table de multiplication par 9 de 0 à 270 en une seule commande avec les fonctions `range()` et `list()`.

(d) Nombres pairs

Répondez à la question suivante en une seule ligne de commande en n'utilisant pas les opérations : +, -, * et /.

Combien y a-t-il de nombres pairs dans l'intervalle [2,10000] inclus ?

(e) Mélange de Monge (bis)

Généraliser le script melange_monge afin que l'on puisse obtenir un mélange de Monge d'un paquet de cartes (1, 2, 3, . . . , 2n) pour n quelconque.

(f) n^2

Ecrire un script qui affiche la liste des n premiers carrés.

```
saisir le nombre de carrés : 5
la liste des 5 premiers carrés est :  [0, 1, 4, 9, 16]
```

(g) Mot de passe

Écrire un script qui prend en paramètre un mot de passe et retourne « mot de passe invalide » s'il vérifie les propriétés suivantes :

- il a 4 caractères
- au moins un chiffre

On pourra utiliser l'opérateur **in** qui teste si un élément fait partie d'une liste et renvoie un booléen : True ou False

```
In [30]: L
Out[30]: [3, 5, 7, 8, 5, 2, 9, 5]

In [31]: 8 in L
Out[31]: True

In [32]: -1 in L
Out[32]: False
```