Les listes en Python

I. Définition et affectation

1. Définition

Une liste est une collection mutable d'objets.

mutable: on peut changer la valeur de ses éléments.

Une liste s'écrit entre crochets.

Exemples de listes:

```python
[5, 7, 3, 8]
['a', 'g', 'j']
['a', 3, True]
```

2. affectation

On affecte une liste à une variable comme n'importe quelle autre affectation.
La liste vide est une liste (donc entre crochets) dans laquelle on ne met rien.

```python
>>> x = []
>>> x
[]
>>> y = [3, 1, 4, 1, 5]
>>> y
[3, 1, 4, 1, 5]
>>> z = ['a', 'c', 3, '7']
>>> z
['a', 'c', 3, '7']
```

Pour récupérer un élément d'une liste, on indique entre crochet le rang de l'élément voulu. Attention l'indexation des listes commence au rang 0.

```python
>>> z = ['a', 'c', 3, '7']
>>> z[2]
3
>>> z[0]
'a'
```

3. Définir une liste en compréhension

On peut définir une liste en compréhension, si les éléments de la liste sont les éléments d'une suite.

```python
>>> a = [3*n+1 for n in range(5)]
>>> a
[1, 4, 7, 10, 13]
```

II. Opérations sur les listes

1. Longueur d'une liste

La fonction `len()` retourne la longueur de la liste mise en paramètre, c'es-à-dire le nombre d'éléments de la liste.

```python
>>> y = [3, 1, 4, 1, 5]
>>> len(y)
5
>>> z = []
>>> len(z)
0
```

2. Concaténation

Pour concaténer deux listes, on utilise l'opérateur `+`.

```python
>>> x = [9, 2, 6, 5]
>>> y = [3, 1, 4, 1, 5]
>>> x + y
[9, 2, 6, 5, 3, 1, 4, 1, 5]
>>> y + x
[3, 1, 4, 1, 5, 9, 2, 6, 5]
```

Il est à noter que l'ordre des opérandes importe lors de la concaténation.

3. Ajout d'un élément en fin de liste

Pour ajouter un élément en fin de liste, on utilise la méthode `append()` . On indique juste dans la parenthèse l'élément à ajouter.

```Python
>>> y = [3, 1, 4, 1, 5]
>>> y.append(9)
>>> y
[3, 1, 4, 1, 5, 9]
```

Il est à noter que la méthode `append()` ne retourne rien. Il faut donc rappeler la variable pour en avoir la valeur.

4. Insérer un élément dans une liste

Pour insérer un élément à la position n°i d'une liste, on utilise la méthode `insert()` . On indique juste dans la parenthèse la position à laquelle on veut insérer l'élément et l'élément à ajouter.

```Python
>>> y = [3, 1, 4, 1, 5]
>>> y.insert(3, 9)
>>> y
[3, 1, 4, 9, 1, 5]
```

Il est à noter que la méthode `insert()` ne retourne rien. Il faut donc rappeler la variable pour en avoir la valeur.

5. Supprimer un élément dans une liste

Pour supprimer un élément à la position n°i d'une liste, on utilise la méthode `pop()` . On indique juste dans la parenthèse la position de l'élément à supprimer. Si aucune position n'est donnée, par défaut, `pop()` supprime le dernier élément de la liste.

```Python
>>> y = [3, 1, 4, 1, 5]
>>> y.pop(2)
4
>>> y
[3, 1, 1, 5]
>>> y.pop()
5
>>> y
[3, 1, 1]
```

Il est à noter que la méthode `pop()` retourne l'élément supprimé.

6. Supprimer un élément d'une liste - le retour

Pour supprimer un élément à la position n°i d'une liste, on utilise l'instruction `del`. On indique juste l'élément à supprimer, en donnant sa position dans la liste.

```Python
>>> y = [3, 1, 4, 1, 5]
>>> del y[2]
>>> y
[3, 1, 1, 5]
>>> del y[1:3]
>>> y
[3, 5]
>>> del y[:]
>>> y
[]
```

Il est à noter que l'instruction `del` ne retourne rien. Il faut donc rappeler la variable pour en avoir la valeur.

7. Supprimer la première occurence dans une liste

Pour supprimer la première occurence d'une valeur dans une liste, on utilise la méthode `remove()`. On indique juste dans la parenthèse la valeur à supprimer. Il faut que l'élément existe, sinon la méthode retournera une exception.

```python
>>> y = [3, 4, 1, 5, 1]
>>> y.remove(1)
>>> y
[3, 4, 5, 1]
>>> y.remove(5)
>>> y
[3, 4, 1]
>>> y.remove(0)
ValueError: list.remove(x): x not in list
```

Il est à noter que la méthode `remove()` ne retourne rien. Il faut donc rappeler la variable pour en avoir la valeur.

8. Trier une liste

Pour trier une liste, on utilise la méthode `sort()` .

```python
>>> y = [3, 1, 4, 1, 5]
>>> y.sort()
>>> y
[1, 1, 3, 4, 5]
```

Il est à noter que la méthode `sort()` ne retourne rien. Il faut donc rappeler la variable pour en avoir la valeur.

9. Remarque

Il existe évidemment d'autres méthodes applicables aux listes. Vous les trouverez sur https://docs.python.org/fr/2/tutorial/datastructures.html