

Python 102



Plus loin avec Python



Squelette d'un module

- Même un fichier destiné à être utilisé comme un script doit être importable
- Un fichier important un module ne devrait subir d'effet de bord en exécutant les fonctionnalités du module importé
- En Python des outils comme **pydoc** ou ceux de tests unitaires ont besoin d'importer vos modules
- Votre code doit toujours vérifier `if __name__ == '__main__'` avant d'exécuter votre programme principal
- Ainsi, ce programme principal ne sera pas exécuté lorsque le module est importé
- [Un squelette de module](#)

Complément sur les listes

- Le type liste dispose de méthodes supplémentaires
- Voici la liste complète des méthodes des objets de type liste
 - **`list.append(x)`**
 - Ajoute un élément à la fin de la liste. Équivalent à `a[len(a):] = [x]`
 - **`list.extend(iterable)`**
 - Étend la liste en y ajoutant tous les éléments de l'iterable. Équivalent à `a[len(a):] = iterable`
 - **`list.insert(i, x)`**
 - Insère un élément à la position indiquée. Le premier argument est la position de l'élément courant avant lequel l'insertion doit s'effectuer, donc `a.insert(0, x)` insère l'élément en tête de la liste, et `a.insert(len(a), x)` est équivalent à `a.append(x)`
 - **`list.remove(x)`**
 - Supprime de la liste le premier élément dont la valeur est x. Une exception est levée s'il existe aucun élément avec cette valeur

Complément sur les listes

➤ `list.pop([i])`

- Enlève de la liste l'élément situé à la position indiquée, et le renvoie en valeur de retour. Si aucune position n'est indiquée, `a.pop()` enlève et renvoie le dernier élément de la liste (les crochets autour du `i` dans la signature de la méthode indiquent bien que ce paramètre est facultatif, et non que vous devez placer des crochets dans votre code)

➤ `list.clear()`

- Supprime tous les éléments de la liste, équivalent à `del a[:]`

➤ `list.index(x[, start[, end]])`

- Renvoie la position du premier élément de la liste ayant la valeur `x` (en commençant par zéro). Une exception **`ValueError`** est levée si aucun élément n'est trouvé.
- Les arguments optionnels `start` et `end` sont interprétés de la même manière que dans la notation des tranches, et sont utilisés pour limiter la recherche à une sous-séquence particulière. L'index renvoyé est calculé relativement au début de la séquence complète, et non relativement à `start`

➤ `list.count(x)`

- Renvoie le nombre d'éléments ayant la valeur `x` dans la liste

Complément sur les listes

- `list.sort(key=None, reverse=False)`
 - Trie les éléments sur place, (les arguments peuvent personnaliser le tri, voir `sorted()` pour leur explication)
- `list.reverse()`
 - Inverse l'ordre des éléments de la liste, sur place
- `list.copy()`
 - Renvoie une copie superficielle de la liste. Équivalent à `a[:]`
- Un exemple utilisant la plupart de ces méthodes

Utiliser les listes comme des piles

- Les méthodes des listes rendent très facile leur utilisation comme des piles, où le dernier élément ajouté est le premier récupéré (« dernier entré, premier sorti », ou LIFO pour « last-in, first-out »)
- Pour ajouter un élément sur la pile, utilisez la méthode `append()`
- Pour récupérer l'objet au sommet de la pile, utilisez la méthode `pop()`, sans indicateur de position

```
lists-as-piles.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4
5  def main():
6      """Lists as piles
7      """
8
9      # Create a stack
10     my_stack = [1, 2, 3, 4]
11     print("my_stack", my_stack)
12
13     # Push values on the stack
14     my_stack.append(5)
15     my_stack.append(6)
16     my_stack.append(7)
17     print("my_stack", my_stack)
18
19     # Pop values from the stack
20     print("Poped value", my_stack.pop())
21     print("my_stack", my_stack)
22     print("Poped value", my_stack.pop())
23     print("my_stack", my_stack)
24     print("Poped value", my_stack.pop())
25     print("my_stack", my_stack)
26     print("Poped value", my_stack.pop())
27     print("my_stack", my_stack)
28
29
30  if __name__ == '__main__':
31     main()
32
```


Références

- Python.org : <https://www.python.org/>
- Learning Python : <https://github.com/thierrydecker/learning-python>
- ...

Outils

- IDE Pycharm Community : <https://www.jetbrains.com/pycharm/>
- Analyse en ligne de code Python : <http://www.pythontutor.com/>
- ...