

TP de Langages de script n° 3 : mutations de listes

Documents à télécharger

Tous les énoncés et les documents à télécharger seront disponibles sur le site

<http://didel.script.univ-paris-diderot.fr/>

Les mutations de listes ou créations fonctionnelles de listes (*list comprehension* en Anglais, souvent traduit de façon assez moche par *compréhension de listes*) sont un des puissants outils de python qui simplifient la vie du programmeur. Les formes les plus simples des mutations de listes sont les suivantes :

```
[transformation(element) for element in liste]
[element for element in liste if propriete(element)]
```

Mais on peut parfaitement les combiner en :

```
[transformation(element) for element in liste if propriete(element)]
```

Par exemple :

```
>>> liste = [1, 34, 5, 4, 7, 8, 93]
>>> [2*i for i in liste]
[2, 68, 10, 8, 14, 16, 186]
>>> [i for i in liste if i%2 == 0]
[34, 4, 8]
>>> [2*i for i in liste if i%2 == 0]
[68, 8, 16]
```

On peut même faire encore plus fort :

```
>>> liste = 'ceci est une liste de mots'.split()
>>> liste
['ceci', 'est', 'une', 'liste', 'de', 'mots']
>>> [u+v for u in liste for v in liste if u != v]
['ceciest', 'ceciune', 'ceciliste', 'cecide', 'cecimots', 'estceci', 'estune',
'estliste', 'estde', 'estmots', 'unececi', 'uneest', 'uneliste', 'unede',
'unemots', 'listececi', 'listeest', 'listeune', 'listede', 'listemots',
'dececi', 'deest', 'deune', 'deliste', 'demots', 'motsceci', 'motsest',
'motsune', 'motsliste', 'motsde']
>>> alphabet = 'ab'
>>> mots_longueur_3 = [i+j+k for i in alphabet for j in alphabet for k in alphabet]
>>> mots_longueur_3
['aaa', 'aab', 'aba', 'abb', 'baa', 'bab', 'bba', 'bbb']
```

Exercice 1 : Premières mutations de listes.

Grâce à des mutations de listes, créez les listes suivantes :

1. Liste des entiers pairs entre 0 et 10.
2. Sous-liste des éléments de rang pair d'une liste donnée.
3. Reprenez le module `aquarium` de l'exercice 5 du TP2.

- (a) Ajoutez un menu déroulant qui permette de choisir parmi une des couleurs de poissons de l'aquarium (classe `Menu` du module `Tkinter`).
- (b) Ajoutez un bouton "pêcheur" qui supprime tous les poissons dont la couleur a été choisie dans le menu déroulant précédent.

Exercice 2 : Retours vers le passé : TP1.

1. Reprenez la question 4 de l'exercice 4 du TP1 et réécrivez la fonction `grand_message` à l'aide d'une mutation de liste.
2. Réécrivez les questions de l'exercice 5 du TP1 grâce à des mutations de listes.
3. Ecrivez une fonction `prefixes` qui renvoie la liste des préfixes d'un mot. Reprenez l'exercice 7 du TP1.

Exercice 3 : Retours vers le passé : TP2.

Toujours à l'aide de mutations de listes.

1. Réécrivez la fonction `sans_e` de l'exercice 3 du TP2.
2. La fonction `zip` sert prendre la diagonale du produit cartésien de deux listes (par abus de langage on dira qu'on *zippe* les deux listes). Plus clairement par un exemple :

```
>>> nombres = [1, 2, 3, 4, 5]
>>> lettres = ['a', 'b', 'c']
>>> zip(nombres, lettres)
[(1, 'a'), (2, 'b'), (3, 'c')]
```

On veut réécrire la fonction `anti_begue` de l'exercice 4 du TP2. Pour cela on remarque la chose suivante : si on zippe la liste des mots de la phrase avec la liste des mots décalée de 1, les mots répétés se retrouvent ensemble.

```
>>> begue
['ceci', 'est', 'un', 'un', 'test']
>>> begue_decale
['est', 'un', 'un', 'test']

>>> zip(begue, begue_decale)
[('ceci', 'est'), ('est', 'un'), ('un', 'un'), ('un', 'test')]
```

Il suffit alors de choisir les bons mots (attention au dernier mot de la phrase...).