Types mutables et immuables
API des séquences

Types mutables et immuables

Classe	Description	Immuable ?
bool	Valeur booléene	X
int	Valeur entière	X
float	Valeur réele	X
list	Séquence d'objets	
tuple	Séquence d'objets	X
str	Chaîne de caractères	X
set	Ensemble d'objets	
frozenset	Ensemble d'objet	X
dict	Table associative	

API (Interface de Programmation d'Applications)

L'API définit comment on utilise un type ou une structure de données. Elle décrit le constructeur et ses arguments, par exemple pour le type *list* :

class **`list`**([iterable])

les opérations applicables sur les données du types ou de la structure de données, par exemple :

	Size	Add	Remove	Contains
List `l`	`len(l)`	`insert` `append`	**`del`**, `pop`, `remove`	**`in`** `index`
Tuple `t`	`len(t)`	–	–	**`in`**
Dictionary `d`	`len(t)`	**`d[k]=v`**	**`del`**, `pop`	**`in`**
Set `s`	`len(s)`	`add`	`remove`	**`in`**

Table prise dans Heineman 2015 Designing Data Structures in Python, O'Reilly.

ainsi que les fonctions qui s'y appliquent. Par exemple, les listes implémentent toutes les opérations de séquences mutables. Les listes fournissent également la méthode supplémentaire suivante :

`sort`(*, key = None, reverse = False)

sort(*, key = None, reverse = False)

sorted(*iterable, *, key=None, reverse=False*)

```
l = [ ('a', 1,10,100), ('b', 100, 10, 1), ('c', 10, 1, 100) ]

# sorted( l, *, key = None, reverse = False )
print( l )
print( sorted( l, key = lambda x:x[1] ) )
print( sorted( l, key = lambda x:x[2], reverse = True ) )

/*
output:
 [('a', 1, 10, 100), ('b', 100, 10, 1), ('c', 10, 1, 100)]
 [('a', 1, 10, 100), ('c', 10, 1, 100), ('b', 100, 10, 1)]
 [('a', 1, 10, 100), ('b', 100, 10, 1), ('c', 10, 1, 100)]
*/
```

Cette méthode trie une *list* en utilisant uniquement des comparaisons < entre les éléments. Les exceptions ne sont pas supprimées - si des opérations de comparaison échouent, l'opération de tri complète échouera (et la liste sera probablement laissée dans un état partiellement modifié).

Pour tous les détails sur la méthode **sort**, consultez la documentation docs.python.org/3.

Opérations communes aux séquences mutables et immuables

Operation	Result
`x in s`	`True` if an item of *s* is equal to *x*, else `False`
`x not in s`	`False` if an item of *s* is equal to *x*, else `True`
`s + t`	the concatenation of *s* and *t*
`s * n` or `n * s`	equivalent to adding *s* to itself *n* times
`s[i]`	*i*th item of *s*, origin 0
`s[i:j]`	slice of *s* from *i* to *j*
`s[i:j:k]`	slice of *s* from *i* to *j* with step *k*
`len(s)`	length of *s*
`min(s)`	smallest item of *s*
`max(s)`	largest item of *s*
`s.index(x[, i[, j]])`	index of the first occurrence of *x* in *s* (at or after index *i* and before index *j*)
`s.count(x)`	total number of occurrences of *x* in *s*

Ce tableau répertorie les opérations de séquence triées en priorité ascendante. Dans la table, s et t sont des séquences du même type, n, i, j et k sont des entiers et x est un objet arbitraire qui satisfait toutes les restrictions de type et de valeur imposées par s.

⚠ `slice` comme `range`, ne prend pas le dernier élément.

Opérations des séquences mutables

Operation	Result
`s[i] = x`	item *i* of *s* is replaced by *x*
`s[i:j] = t`	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*
`del s[i:j]`	same as `s[i:j] = []`
`s[i:j:k] = t`	the elements of `s[i:j:k]` are replaced by those of *t*
`del s[i:j:k]`	removes the elements of `s[i:j:k]` from the list
`s.append(x)`	appends *x* to the end of the sequence (same as `s[len(s):len(s)] = [x]`)
`s.clear()`	removes all items from `s` (same as `del s[:]`)
`s.copy()`	creates a shallow copy of `s` (same as `s[:]`)
`s.extend(t)` or `s += t`	extends *s* with the contents of *t* (for the most part the same as `s[len(s):len(s)] = t`)
`s *= n`	updates *s* with its contents repeated *n* times
`s.insert(i, x)`	inserts *x* into *s* at the index given by *i* (same as `s[i:i] = [x]`)
`s.pop([i])`	retrieves the item at *i* and also removes it from *s*
`s.remove(x)`	remove the first item from *s* where `s[i] == x`
`s.reverse()`	reverses the items of *s* in place

Dans la table, s est une instance d'un type de séquence mutable, t est un objet itérable et x est un objet arbitraire qui satisfait toutes les restrictions de type et de valeur imposées par s.

Tous les types et structures de données intégrés du langage Python sont ainsi documentés sur docs.python.org/3.

Il est utile de voir comment les choses ont été conçues en Python de manière à ce que lorsque nous concevrons nos propres types et structures de données nous pourrons nous en inspirer.

Un des objectifs lors de l'implémentation d'un type ou d'une structure de données est d'optimiser les temps d'exécution du plus grand nombre d'opérations possible.

Parfois, pour optimiser une opération qui sera exécutée un grand nombre de fois dans une application particulière, il faut changer la structure de données, ce qui peut entraîner des modifications sur d'autres opérations et sur l'utilisation mémoire.

Implémenter des structures de données adéquates est un Art de faire des compromis.