Comparaisons

	p-uplet	Tableau	Dictionnaire	Chaîne de caractères
type Python	`tuple`	`list`	`dict`	`str`
Suite d'éléments	ordonnés	ordonnés	sans ordre	ordonnés
Eléments modifiables	immuable	muable	muable	immuable
Création/notation	`t = (1, 2, 3, 4)` ou `t = 1, 2, 3, 4`	`l = [1, 2, 3, 4]`	`d = {'one':1, 'two':2, 'three':3}`	`s='abcd'`

Création par compréhension	impossible	`l1 = [x**2 for x in range(5)]`
		`l2 = [3*i for i in [1, 2, 3]]`
		`l3 = [x for x in range(9) if x!=2]`
		`l4 = [f(x) for x in l3]`
		`d1 = {x: x**2 for x in range(5)}`
			`d2 = {str(x): x for x in range (5)}`
			`d3 = {x: x for x in range(5) if x=2}`
			`d4 = {x: f(x) for x in [1, 2, 3]}`
				Impossible

Nombre d'éléments	`len(t)`	`len(l)`	`len(d)`	`len(s)`
Indices	indexés de `0` à `len(t) - 1`		indexés de `0` à `len(l) - 1`	
Accès à un élément	`t[index]`	`l[index]`	`d[clé]`	`s[index]`
Tranches	`t[i1:i2]` de `i1` (inc.) à `i2` (exclus)	`l[i1 : i2]` de `i1` (inc.) à `i2` (exclus)	impossible	`s[i1:i2]` de `i1` (inc.) à `i2` (exclus)

	p-uplet	Tableau	Dictionnaire	Chaîne de caractères

Mot-clé `in`	`>>> 1 in (1, 2, 3)`
	`True`
	`>>> for i in (1, 2 ,3):`
	`  print(i, end='-')`
	`1-2-3-`
		`>>> 1 in [1, 2, 3]`
		`True`
		`>>> for i in [1, 2 ,3]:`
		`  print(i, end='-')`
		`1-2-3-`
			Le mot-clé `in` teste la presence d'une clé, pas d'une valeur
			`>>> 'one' in {'one':1, 'two':2}`
			`True`
			`>>> for key in {'one':1, 'two':2}:`
			`  print(key)`
				`>>> 'z' in 'hello world'`
				`False`
				`for car in 'hello':`
				`  print(car, end='-')`
				`h-e-l-l-o-`

Modifier un élément	impossible	`l[index]= nouvelle_valeur` avec `index < len(l)` sinon erreur	`d[clé] = nouvelle_valeur` la clé est créée si elle n'existe pas	Impossible

Opérations	`>>> 3*(1, 2)`
	`(1, 2, 1, 2, 1, 2)`
	`>>> (1, 2) + (3, 4)`
	`(1, 2, 3, 4)`
		`>>> 3*[1, 2]`
		`[1, 2, 1, 2, 1, 2]`
		`>>> [1, 2] + [3, 4]`
		`[1, 2, 3, 4]`
			impossible	`>>> 3*'abc'`
				`'abcabcabc'`
				`>>> 'abc'+'def'`
				`'abcdef'`

Ajouter de nouveaux éléments	Impossible (mais on peut créer un nouveau p-uplet par concaténation et écraser l'ancien :
	`t = (1, 2, 3)`
	`t = t + (4, 5, 6)`
		`>>> [1,2,3].append(4)`
		`[1, 2, 3, 4]`
		`>>> ['a','b','d'].insert(2,'c')`
		`['a', 'b', 'c', 'd']`
		`>>> [1,2,3].extend([4,5,6])`
		`[1, 2, 3, 4, 5, 6]`
			`d[nouvelle_clé] = valeur` L'ajout de nouveaux éléments est automatique quand la clé n'existe pas déjà	Impossible (mais on peut créer une nouvelle chaîne par concaténation et écraser l'ancienne
				`>>>ch = 'hello'`
				`>>> ch = ch + 'world'`

Supprimer des éléments	Impossible de supprimer un élément seul. On peut supprimer le	Avec le mot-clé `del` `>>> l= ['a', 'b', 'c']`	Avec le mot-clé `del` `>>> d={'one':1, 'two':2,`	Impossible de supprimer un caractère seul. On peut supprimer la

	p-uplet	Tableau	Dictionnaire	Chaîne de caractères
	p-uplet avec `>>> t= ['a', 'b', 'c']` `>>> del t`	`>>> del l[2]` `>>> del l` Avec les methodes `>>> l= [1, 2, 3, 4, 5]` `>>> l.pop()`	`'three':3}` `>>> del d['two']` `>>> del d` Avec les methodes `>>> d = {'one':1, 'two':2, 'three':3}` `>>> d.pop('two')`	chaîne avec `>>> s= 'hello world'` `>>> del s`
Quelques méthodes associées	Aucune méthode disponibles	`>>> l = [5, 8, 2, 1.]` `>>> l.sort()` `>>> l` `[1, 2, 5, 8]` `>>> l.index(5)` `2`	`>>> d = {'one':1, 'two':2, 'three':3}` `>>> d.keys()` `dict_keys(['one', 'two', 'three'])` `>>> d.values()` `dict_values([1, 2, 3])` `>>> d.items()` `dict_items([('one', 1), ('two', 2), ('three', 3)])`	`>>> s = "hello world"` `>>> s.upper()` `'HELLO WORLD'` `>>> s.split()` `['hello', 'world']` `>>> s.find("lo")` `3`
Conversion (cast)	`>>>tuple("abc")` `('a', 'b', 'c')` `>>> tuple(range(5))` `(0, 1, 2, 3, 4)`	`>>> list('abc')` `['a', 'b', 'c']` `>>> list(range(5))` `[0, 1, 2, 3, 4]`	`>>> t = [("one", 1), ("two", 2), ("three", 3)]` `>>> d = dict(t)` `>>> d` `{'one': 1, 'three': 3, 'two': 2}`	`>>> l = ["h", "e", "l", "l", "o"]` `>>> "".join(l)` `hello`
Copier	`t1 = t2` ou `a, b, c, … = t` pour disperser tuple `t`	`l2 = l1` créé deux noms pour la même liste. Pour copier les valeurs de `l1` dans `l2` il faut utiliser : `l2 = l1[:]` ou `l2 = list(l1)` ou `l2 = l1.copy()`	`d2 = d1` créé deux noms pour le même dictionnaire. Pour copier les valeurs de `d1` dans `d2` il faut utiliser : `d2 = dict(d1)` ou `d2 = d1.copy()`	`s1 = s2`
Argument de	inchangé par la	peut-être modifié	peut-être modifié	inchangé par la

	p-uplet	Tableau	Dictionnaire	Chaîne de caractères
fonctions	fonction	par la fonction	par la fonction	fonction
Utilisation notable	fonction renvoyant plusieurs valeurs	tableau de tableaux (matrices)	p-uplet nommé	manipulations de textes