

V – Listes et fonctions d'ordre supérieur

- qu'est-ce qu'une liste?
- c'est une structure de données
- permet l'utilisation d'une collection d'objets
- collection **ordonnée** d'objets de même type quelconque
- liste particulière : la liste vide (aucun élément)
- liste+, l'ensemble des listes sauf la liste vide

Notation en Python

- liste composée des éléments a, b, ..., n est notée [a, b, ..., n]
- [2, 3, -1] est liste d'entiers
- liste vide notée []
- liste est une expression de Python
- on peut l'affecter à un symbole : l = [2, 3, 4, 5, 6]
- longueur est son nombre d'éléments
- la longueur de la liste vide est 0, en Python, len
- en Python, une liste est séquence (comme les chaînes de caractères) le type est list

Fonctions et opérations sur les listes

- construction de listes
- la fonction range :
- range : entier x entier x entier -> séquence d'entiers
- range (imin, imax, di) construit séquence de nombres entiers de imin à imax (exclue), séparés par di (1 par défaut)
- pour construire une liste à partir de range, il faut transformer le résultat à l'aide de la fonction list
- exemple :
- l = list(range(0, 10)) # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (pas = 1)
- l2 = list(range(0, 10, 2)) # 0, 2, 4, 6, 8 (pas = 2)
- range très utilisée pour les itérations sur les séquences

Fonctions et opérations sur les listes

- prendre un élément quelconque :
- les éléments d'une liste ordonnés
- accès en donnant l'index de l'élément (entre 0 et len(l)-1), comme pour les chaînes caractères
- si l'index i est négatif, on part de la fin
- si i >= len(liste) ou i < - len(liste) erreur (list index out of range)

Fonctions et opérations sur les listes

- extraire un sous-ensemble d'éléments consécutifs :
- donne index de début et fin (non compris) entre crochets séparés par ":" : l1 = l[0:4]
- si pas d'index de début : part de 0 : l[0:4] == l[:4]
- si pas d'index de fin on va jusque len(l)
- fonctionne aussi sur les chaînes de caractères

Fonctions et opérations sur les listes

- suppression d'élément(s)
- del permet de supprimer un ou plusieurs élément(s) d'une liste
- del l[0]
- peut aussi se faire en supprimant un sous-ensemble de la liste
- del l[0:3]
- del l[1:]

Fonctions et opérations sur les listes

- remplacement d'élément(s)
- affectation avec le bon index
- `l[0] = 2`
- peut aussi se faire en remplaçant un sous-ensemble de la liste (partie droite doit être une liste éventuellement de taille différente)
- `l[0:3] = []`
- `l[0:2] = [3, 5, 7, 22]`

Fonctions et opérations sur les listes

- ajout d'élément(s)
- insertion d'un ou plusieurs élément(s) dans une liste en donnant une « tranche vide » en partie gauche d'une affectation
- rappel : la partie droite doit être une liste
- `l[2:2] = [2000]`
- `l[-1:-1] = [-1, -2, -3]`

Fonctions et opérations sur les listes

- concaténation de listes
- l'opérateur `+` appliquée à deux listes les concatène en une nouvelle liste
- l'opérateur `*` appliquée à une liste et un entier `n`, concatène `n` fois la liste donnée
- `l1 = [1, 2] + [3, 4, 5]`
- `l2 = l1 * 2`

Fonctions et opérations sur les listes

- autres instructions sur les listes
- `x in l` : teste si `x` appartient à `l`
- `x not in l` : teste si `x` n'appartient pas à `l`

Itérations sur les séquences

- il est souvent nécessaire de parcourir le contenu d'une liste (ou d'une séquence)
- bien sûr il est possible de le faire avec une itération `while`
- Python propose une instruction pour le parcours d'une séquence : **for in**
- exemple : affichage de tous les éléments de la liste `l`

```
for x in l:  
    print(x, " ", end="")
```

Itérations sur les séquences

- construction d'une liste en compréhension :
- `[expr for x in seq]` construit la liste des valeurs de `expr` pour toutes les valeurs de `x` dans la séquence `seq`
- exemple :
- `[2**x for x in range(0,11)]` -> liste des puissances de 2 de 2^0 à 2^{10}
- `[expr for x in seq if cond]` construit la liste des valeurs de `expr` pour toutes les valeurs de `x` dans la séquence `seq` si la condition est vérifiée
- exemple :
- `[x**2 for x in range(0,11) if pairQ(2**x)]` -> liste des carrés pairs de 0^2 à 10^2

Opérations classiques sur les listes et fonctions d'ordre supérieur

- opérations très fréquents sur les listes :
- appliquer une opération à tous les éléments d'une liste
- réduction d'une liste (par addition par exemple)
- sélections d'éléments d'une liste
- plusieurs mise en œuvre
- une utilisant des fonctions d'ordre supérieur (fonction qui a comme paramètre une fonction)

La fonction map

- on veut, à partir de la liste $[e1, e2, \dots, en]$ obtenir la liste $[f(e1), f(e2), \dots, f(en)]$, f étant une fonction unaire
- utiliser une itération "for"
- construction d'une liste en compréhension
- fonction d'ordre supérieur "map"
- `map (f, liste)` rend la séquence obtenue en appliquant la fonction f à chacun des éléments de liste
- il faut transformer la séquence en liste : `list(...)`
- `list (map (f, [e1, e2, ..., en]))` rend $[f(e1), f(e2), \dots, f(en)]$

La fonction map

- cette fonction peut être utilisée pour combiner plusieurs listes de même longueur :
- `list(map(mul, [2, 4, 5], [6, 4, 3]))` rend
- `[mul (2, 6), mul (4, 4), mul (5, 3)]`

La fonction reduce

- à partir d'une liste $[e1, e2, e3, \dots, en]$ on veut obtenir la liste $[f(\dots f(f(e1, e2), e3), \dots, en)]$, f étant une fonction binaire
- par exemple cela permet de faire la somme des entiers d'une liste
- utilisation d'une itération "for"
- utilisation d'une fonction d'ordre supérieur `reduce` (elle fait partie de la bibliothèque `functools`)
- `functools.reduce (f, [a, b, c, d])` a pour résultat l'évaluation de la fonction $f(f(f(a, b), c), d)$

La sélection : fonction filter

- sélectionner dans une liste les éléments vérifiant une condition
- utilisation d'une itération "for"
- construction d'une liste en compréhension
- fonction d'ordre supérieur `filter`
- `filter (pred, liste)` permet de sélectionner les éléments d'une liste satisfaisant un prédicat unaire
- rend une séquence qu'il faut transformer pour obtenir une liste
- `list (filter (pairQ, [8, 5, 2, 1, 12]))` -> `[8, 2, 12]`