Introduction à l'informatique

Cours 7.5

kn@lmf.cnrs.fr
https://usr.lmf.cnrs.fr/~kn

université PARIS-SACLAY

But du cours

Le but est de consolider les connaissances en Python, particulièrement sur les tableaux et dicionnaires. On va se donner une série de problème concrets (ex: « trouver l'indice de la valeur maximale d'un tableau ») et on va essayer:

1. de le résoudre en Python
2. de reconnaître les problèmes similaires ≡ dont le code aura la même structure
3. on admet que les solutions données sont efficaces (cours Algo et Structure de données S2)

Pour pouvoir faire le point (3.) on évitera certaines opérations Python non vue dans le cadre de ce cours, en particulier celles qui s'écrivent simplement mais cachent des opérations coûteuses.

Les supports de cours seront minimaux: il faudra aussi apprendre le fichier Python associé!

Plan

1 Présentation du cours ✓
2 Le système Unix ✓
3 Le système Unix (2) ✓
4 Python (1) : expressions, types de bases, if/else ✓
5 Python (2) : boucles, tableaux, exceptions ✓
6 Python (3) : Textes, chaînes de caractères, entrées/sorties ✓
7 Python (4) : Fonctions ✓
8 Python (5) : Concepts avancés ✓
9 Python (6) : Algorithmes sur les tableaux
  9.1 Recherche d'un élément dans un tableau
  9.2 Agrégat d'un tableau
  9.3 Groupements de valeurs

Appartenance d'un élément

Énoncé :
écrire une fonction `appartient(t, e)` qui renvoie `True` si `e` est dans le tableau `t` et `False` sinon.

On écrit la fonction ainsi qu'une fonction de test.

Existence d'un entier pair

Énoncé :
écrire une fonction `contient_pair(t)` qui renvoie `True` si `t` contient un entier pair.

On écrit la fonction ainsi qu'une fonction de test.

Algorithme général

Si on généralise un peu, le code ci-dessous permet de répondre à la question:

$$\exists\ x \in t, \text{ tel que } P(x)$$

où P est un prédicat ($\equiv$ un test) booléen : x est positif, x est égal à une valeur donnée, ...

```
def exists(t):
    for x in t:
        if P(x):   #remplacer par le vrai test
            return True
    return False
```

On parcours au pire tout le tableau (quand aucun élément ne vérifie le prédicat et qu'on doit renvoyer `False`)

Variante : renvoyer l'indice de l'élément tel que...

- On fait une boucle sur les indices (pas les éléments)
- Mais que faire si aucun élément vérifie ?:
- Renvoyer `-1` ⇒ pas terrible car `-1` est un indice valide de tableau en Python (ce qui est une mauvaise idée)
- Renvoyer `None` ⇒ Ok, à condition de bien documenter
- Lever une exception ⇒ Ok, à condition de bien documenter

Recherche de plusieurs éléments dans un tableau ?

Les fonctions précédentes permettent de renvoyer un élément du tableau.

Comment faire si on veut renvoyer *toutes* les valeurs qui vérifient un prédicat ?

Problème, on ne sait pas, a priori, combien de valeurs on va renvoyer

Ajout d'élément en fin de tableau

On peut ajouter un élément en fin de tableau avec l'opération append

```
>>> tab = [10, 11, 42]
>>> len(tab)
3
>>> tab.append(100)
>>> tab
[10, 11, 42, 100]
>>> len(tab)
4
>>> tab2 = tab
>>> tab2.append(18)
>>> tab2
[10, 11, 42, 100, 18]
>>> tab
[10, 11, 42, 100, 18]
```

Attention au partage !

Exemples d'application

- Renvoyer tous les éléments strictement positifs d'un tableau
- Renvoyer toutes les lettres minuscules d'une chaîne (dans un tableau)

```
def filtrer(t):
    res = []
    for x in t:
        if P(x):    #remplacer par le prédicat
            res.append(x)
    return res
```


Plan

1 Présentation du cours ✓
2 Le système Unix ✓
3 Le système Unix (2) ✓
4 Python (1) : expressions, types de bases, if/else ✓
5 Python (2) : boucles, tableaux, exceptions ✓
6 Python (3) : Textes, chaînes de caractères, entrées/sorties ✓
7 Python (4) : Fonctions ✓
8 Python (5) : Concepts avancés ✓
9 Python (6) : Algorithmes sur les tableaux
  9.1 Recherche d'un élément dans un tableau ✓
  9.2 Agrégat d'un tableau
  9.3 Groupements de valeurs

Exemples

- Renvoyer la somme des éléments d'un tableau (d'entiers)
- Renvoyer le maximum (ou le minimum) d'un tableau (d'entiers)
- Renvoyer le point le plus éloigné de l'origine (dans un tableau de points données par des coordonnées flottantes `(x,y)`

Structure générale du code

```
def agregat(t):
    accu = ...      #valeur initiale
    for x in t:
        if test(x):    #l'élément courant est utile (optionnel)
            accu = comb(x, accu)  #on le combine
    return accu
```

- `accu` est initialisé à une valeur par défaut, qui est renvoyée en particulier pour le tableau vide
- `test` est utilisé si tous les éléments n'ont pas vocation à être pris en compte dans le calcul
- `comb` est la fonction de combinaison

Attention, certaines fonctions suivent ce motif mais ne sont définies que pour des tableaux non vide (ex: moyenne).

Plan

1 Présentation du cours ✓
2 Le système Unix ✓
3 Le système Unix (2) ✓
4 Python (1) : expressions, types de bases, if/else ✓
5 Python (2) : boucles, tableaux, exceptions ✓
6 Python (3) : Textes, chaînes de caractères, entrées/sorties ✓
7 Python (4) : Fonctions ✓
8 Python (5) : Concepts avancés ✓
9 Python (6) : Algorithmes sur les tableaux
  9.1 Recherche d'un élément dans un tableau ✓
  9.2 Agrégat d'un tableau ✓
  9.3 Groupements de valeurs

Problème

- Le tableau de départ contient des éléments de la forme (clé, valeur)
- Des clés peuvent être répétées
- On veut calculer une fonction d'agregat *toutes les valeurs de chaque clé*

Exemples:

- On a un tableu ("Nom", note) (un nom peut apparaître plusieurs fois), on veut calculer la note maximale *pour chaque nom*.
- On a une chaîne de caractères, on veut compter combien de fois chaque caractère apparaît.

⇒ Ici, on peut imaginer que la chaîne `"ABCABD"` est vue comme le tableau:

```
[ ('A',1), ('B',1), ('C',1), ('A',1), ('B',1), ('D',1) ]
```

et pour chaque lettre, on veut la somme des `1`.

Solution

On parcourt le tableau avec un dictionnaire auxiliaire où on va calculer l'aggregat pour chaque clé

.

On peut ensuite faire ce qu'on veut avec le dictionnaire (par exemple le renvoyer tel quel ou le filtrer, prendre le maximum etc...)

.