

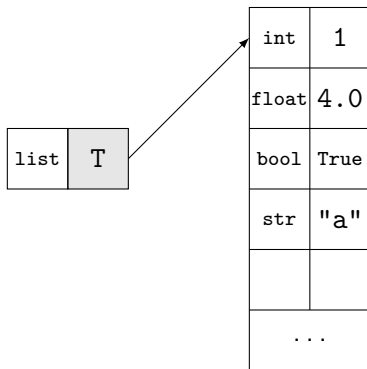
II – Listes, dictionnaires

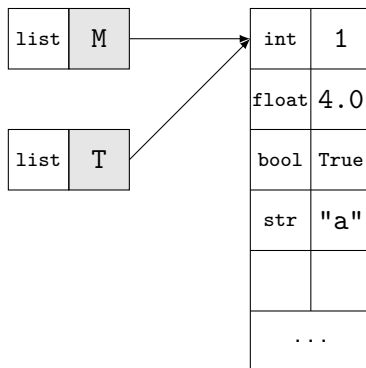
M. Solnon

PT, Lycée Gustave Eiffel, Dijon

6 octobre 2026

Création d'une liste : `T = [1, 4.0, True, "a"]`



Alias : $M = T$ 

A-t-on copié T ?

Pour ceux qui ont un ordinateur

```
1 T = [1, 4.0, True, "a"]  
2 M = T  
3 T[0] = 42  
4 print(M)
```

A-t-on copié T ?

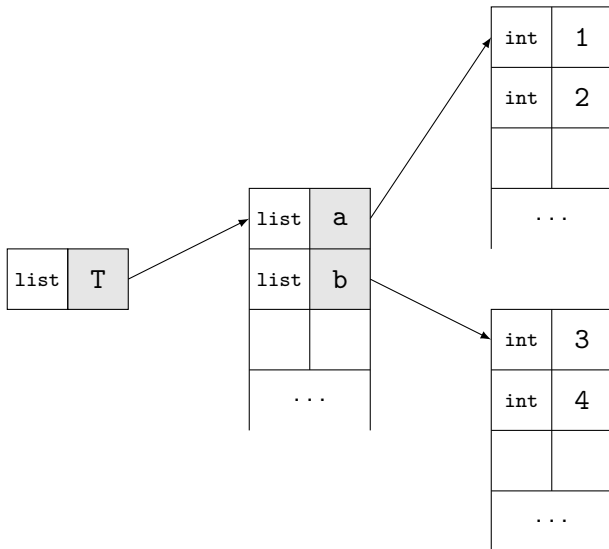
Pour ceux qui ont un ordinateur

```
1 T = [1, 4.0, True, "a"]
2 M = T
3 T[0] = 42
4 print(M)
```

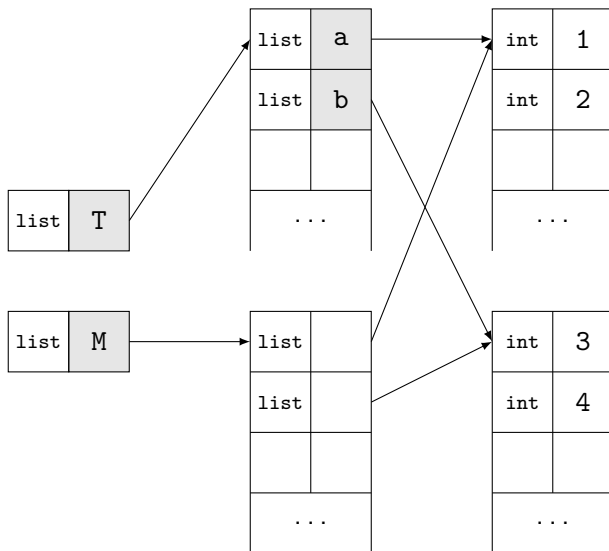
Solution : la méthode `copy`

```
1 T = [1, 4.0, True, "a"]
2 M = T.copy()
3 T[0] = 42
4 print(M)
```

Matrice : $a = [1,2]$; $b = [3,4]$; $T = [a,b]$;



Copie superficielle (*shallow*) : $M = T.copy()$



Solution : copie profonde

```
1 from copy import deepcopy
2 T = [[1,2],[3,4]]
3 M = deepcopy(T)
```

Parcours

Parcours sur les éléments

```
1 for element in L :  
2     # instructions en fonction de element
```

Parcours sur les indices

```
1 for i in range(len(L)) :  
2     # instructions en fonction de i
```

Somme des éléments

Code python

Somme des éléments

Code python

```
1 def somme(L) :  
2     """Préc. : L = liste de nombres"""  
3     S = 0  
4     for x in L :  
5         S = S + x  
6     return S
```

Complexité

Somme des éléments

Code python

```
1 def somme(L) :  
2     """Préc. : L = liste de nombres"""  
3     S = 0  
4     for x in L :  
5         S = S + x  
6     return S
```

Complexité

$O(n)$ où n est le nombre d'éléments de L .

Minimum/maximum d'une liste

Code python

Minimum/maximum d'une liste

Code python

```
1 def minimum(L) :  
2     """Préc.: L=liste de nombres non vide"""  
3     mini = L[0]  
4     for x in L :  
5         if x < mini :  
6             mini = x  
7     return mini
```

Complexité

$O(n)$ où n est le nombre d'éléments de L .

Exercice

Exercice 1.3.1

Écrire une fonction calculant le maximum d'une liste.

Lieu du minimum/maximum d'une liste

Code python

Lieu du minimum/maximum d'une liste

Code python

```
1 def argmin(L) :  
2     """Préc. : L = liste de nombres non vide"""  
3     imin = 0  
4     for i in range(1, len(L)) :  
5         if L[i] < L[imin] :  
6             imin = i  
7     return imin
```

Complexité

$O(n)$ où n est le nombre d'éléments de L .

Question

Quelle est la différence avec le code suivant ?

```
1 def argmin(L) :  
2     """Préc. : L = liste de nombres non vide"""  
3     imin = 0  
4     for i in range(1, len(L)) :  
5         if L[i] <= L[imin] :  
6             imin = i  
7     return imin
```

Exercices

Exercice 1.3.2

Écrire une fonction déterminant un indice du lieu du maximum d'une liste.

Exercice 1.3.3

Écrire une fonction prenant en argument une liste de nombres non vide L et renvoyant la liste des indices i vérifiant $L[i] = \min(L)$.

Recherche séquentielle d'un élément

Code python

Recherche séquentielle d'un élément

Code python

```
1 def recherche(L, e):  
2     for x in L :  
3         if x == e :  
4             return True  
5     return False
```

Complexité

$O(n)$ où n est le nombre d'éléments de L .

Question

Quelle est la différence avec le code suivant (méthode du drapeau) ?

```
1 def recherche(L, e):  
2     drapeau = False  
3     for x in L :  
4         if x == e :  
5             drapeau = True  
6     return drapeau
```

Nombre d'occurrences d'un élément

Code python

Nombre d'occurrences d'un élément

Code python

```
1 def compte(L, e):  
2     nb = 0 # nb d'occurrences de e dans L  
3     for x in L :  
4         if x == e :  
5             nb = nb + 1  
6     return nb
```

Complexité

$O(n)$ où n est le nombre d'éléments de L .

Tableaux multidimensionnels (matrices)

Principe (convention usuelle)

- Représentation comme liste de listes.
- Convention usuelle : chaque ligne est représentée par une ligne

Exemple

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

$$\begin{array}{l} 1 \quad M = [[1, 2, 3], \\ 2 \quad \quad 4, 5, 6]] \end{array}$$

Parcours

Parcours sur les éléments

```
1 for ligne in M :  
2     for coefficient in ligne :  
3         # instructions en fonction de coefficient
```

Parcours sur les indices

```
1 for i in range(len(M)) :  
2     for j in range(len(M[0])) :  
3         # instructions en fonction de M[i][j]
```

Exercice

Exercice 1.4.2

On suppose qu'une image en niveau de gris est représentée par une matrice. Un pixel est codé par un entier (0 : blanc, 255 : noir).

Écrire une fonction `image_blanche(n,p)` représentant une image blanche (attention aux alias!) de n lignes et p colonnes.

Écrire une fonction `nb_pixels_blancs(M)` prenant en argument une matrice M représentant une telle image et renvoyant le nombre de pixels blancs de l'image.

Création

Dictionnaire vide

```
1 dico_vide = {}
```

Avec quelques éléments

```
2 d = { 3:[],  
3      'clef':42,  
4      (15,15):[1,2,3,4]}
```

En compréhension

```
5 dico_comp = {str(i) : i for i in range(42)}
```

Manipulation

Accès à une valeur

```
1 d['clef']  
2 d[42]
```

Réaffectation, ajout d'un élément

```
3 d['clef'] = 1515  
4 d[42] = 7
```

Test d'appartenance d'une clef

```
5 'clef' in d  
6 1515 in d
```

Alias, copies superficielles et profondes

Complètement identique aux listes.

Parcours

Officiellement au programme

- `for clef in dict.keys()`
- `for clef, val in dict.items()`

Syntaxe la plus pratique

- `for clef in dict`

Exercice

Exercice 1.3.4

Écrire une fonction `enumeration` qui prend en argument une liste L d'objets non mutables et qui renvoie le dictionnaire dont les éléments sont les couples de la forme $e : nb$, où

- e désigne un élément de L (sans répétition) ;
- nb désigne le nombre d'occurrences de e dans L .

Par exemple, un appel de `enumeration([42, "", 42, True, 42, ""])` renverra le dictionnaire suivant.

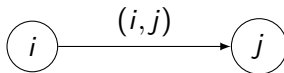
```
{42 : 3,  
 "" : 2,  
 True : 1}
```

Rappels : graphe orienté

Formellement

- S : ensemble des sommets
- $A \subset S \times S$: ensemble des arêtes

Représentation

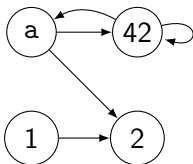


Vocabulaire

- i : un *antécédent* de j
- j : un *successeur* de i

Dictionnaire d'adjacence

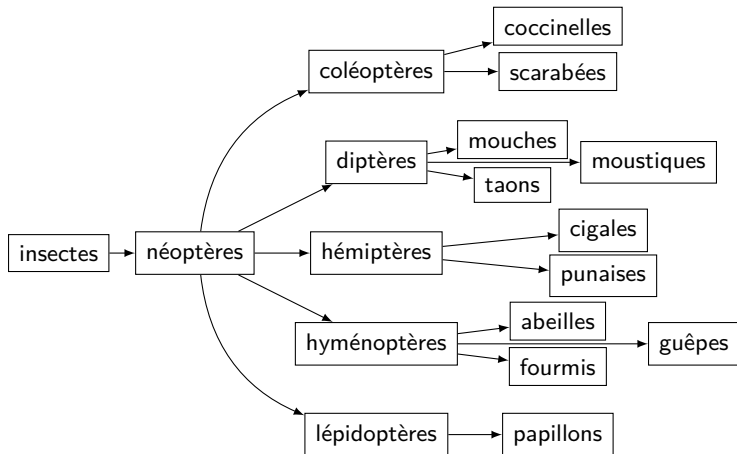
Notre exemple de graphe



Le dictionnaire associé

```
1 d = {1 : [2] ,
2       2 : [] ,
3       'a' : [2, 42] ,
4       42 : ['a', 42]}
```

Exercice : arbre phylogénique des insectes (1)



Exercice : arbre phylogénique des insectes (2)

Exercice 2.4.3

Créer un dictionnaire représentant ce graphe.

Exercice : arbre phylogénique des insectes (3)

Écrire des instructions ou des fonctions pour :

1. Déterminer le nombre de sommets du graphe.
2. Déterminer le nombre d'arêtes du graphe.
3. Déterminer la moyenne du nombre de successeurs par sommet. Que dire de la moyenne du nombre d'antécédents ?
4. Déterminer la liste des sommets du graphe.
5. Déterminer la liste des arêtes du graphe (une arête pourra être représentée par un couple de sommets).
6. Déterminer la liste des boucles (*i.e.* des sommets qui sont leurs propres successeurs).
7. Déterminer la liste des sommets sans successeurs.
8. Déterminer la liste des sommets sans antécédents.

Fonction de hachage

Définition

Fonction qui prend en argument un objet (informatique) et qui renvoie un entier en sortie.

Collision

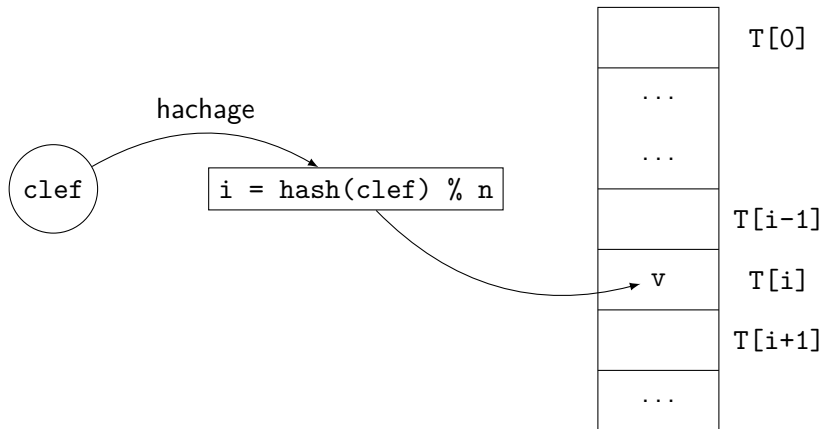
Quand deux objets ont une même image par une fonction de hachage

En python

- Fonction déjà fournie : `hash`
- Limitation : ne s'applique qu'à un objet non mutable.

Fonctionnement d'un dictionnaire : $d[\text{clef}] = v$

Tableau T (caché) : $n = \text{len}(T)$



De nombreux mécanismes cachés

Ce dont vous n'avez pas à vous soucier

- Fonctionnement de `hash`
- Gestion et extension du tableau
- Gestion des collisions lors de l'affectation
- Mécanisme pour retrouver une valeur malgré les collisions

Ce dont vous avez à vous soucier

- Limitation sur les clefs : appliquer `hash` dessus
- Conséquence : clefs = objets non mutables

Complexités : tableau à retenir

Pour un dictionnaire d de n éléments.

Opération	Commande	Complexité
Création vide	<code>d={}</code>	$\mathcal{O}(1)$
Création avec n valeurs	<code>d={c₁:v₁, ..., c_n:v_n}</code>	$\mathcal{O}(n)$
Accès	<code>d[c]</code>	$\mathcal{O}(1)$
Insertion ou modification	<code>d[c]=v</code>	$\mathcal{O}(1)$
Parcours	<code>for c in d:</code>	$\mathcal{O}(n)$
Copie superficielle	<code>d.copy()</code>	$\mathcal{O}(n)$
Test de présence (clef)	<code>c in d</code>	$\mathcal{O}(1)$
Nombre d'éléments	<code>len(d)</code>	$\mathcal{O}(1)$
Liste des clefs	<code>d.keys()</code>	$\mathcal{O}(n)$
Liste des couples clefs/valeurs	<code>d.items()</code>	$\mathcal{O}(n)$

Table de hachage

En Python, on dispose de la fonction `ord`, qui prend en argument un caractère et renvoie son point de code unicode.

Proposer une fonction de hachage prenant en objet une chaîne et renvoyant une valeur dans $\llbracket 0, 2^8 \rrbracket$ (de sorte que l'on puisse la coder sur un octet).

Matrices

Comment peut-on faire pour manipuler des matrices avec Python en utilisant un dictionnaire ?

Matrices parcimonieuses (1)

On s'intéresse ici aux matrices *parcimonieuses*, i.e. dont la plupart des coefficients sont nuls. Une telle matrice M de dimensions n, p pourra être codée par un dictionnaire ayant pour couples clefs/valeurs :

- 'dim' : (n, p)
- $(i, j) : M_{i,j}$ pour chaque couple (i, j) tel que $M_{i,j} \neq 0$.

Par exemple, la matrice

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \end{pmatrix}$$

sera codée par le dictionnaire

$$\{\text{'dim'} : (2, 4) \ , \ (1, 2) : 4\}.$$

Matrices parcimonieuses (2)

Proposer alors une fonction d'addition de deux telles matrices (de même dimension).

Si la première matrice contient c coefficients non nuls, et la seconde c' , quelle est la complexité temporelle de cet algorithme ?