This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Formation NSI

luc.vincent@ac-bordeaux.fr

Déroulement

- Matin
  - Données structurées
    - réflexion et échange sur les notions (les listes, les tableaux, les dictionnaires …
  - Parcours de listes et dictionnaires
    - index, élément, énumération
  - Muable ou immuable

- Après midi
  - Projet

    - Puissance4 Présentation
    - Light_out (Mise en Œuvre)
    - Grille d'évaluation

Ressources

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les programmes

- En première

2. Représentation des données types construits

Tableau indexé, tableau donné en compréhension	Lire et modifier les éléments d'un tableau grâce à leurs index. Construire un tableau par compréhension. Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i] [j]. Itérer sur les éléments d'un tableau.	Seuls les tableaux dont les éléments sont du même type sont présentés. Aucune connaissance des tranches (slices) n'est exigible. L'aspect dynamique des tableaux de Python n'est pas évoqué. Python identifie listes et tableaux. Il n'est pas fait référence aux tableaux de la bibliothèque NumPy.
Dictionnaires par clés et valeurs	Construire une entrée de dictionnaire. Itérer sur les éléments d'un dictionnaire.	Il est possible de présenter les données EXIF d'une image sous la forme d'un enregistrement. En Python, les p-uplets nommés sont implémentés par des dictionnaires. Utiliser les méthodes keys(), values () et items ().

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les programmes

- En terminale

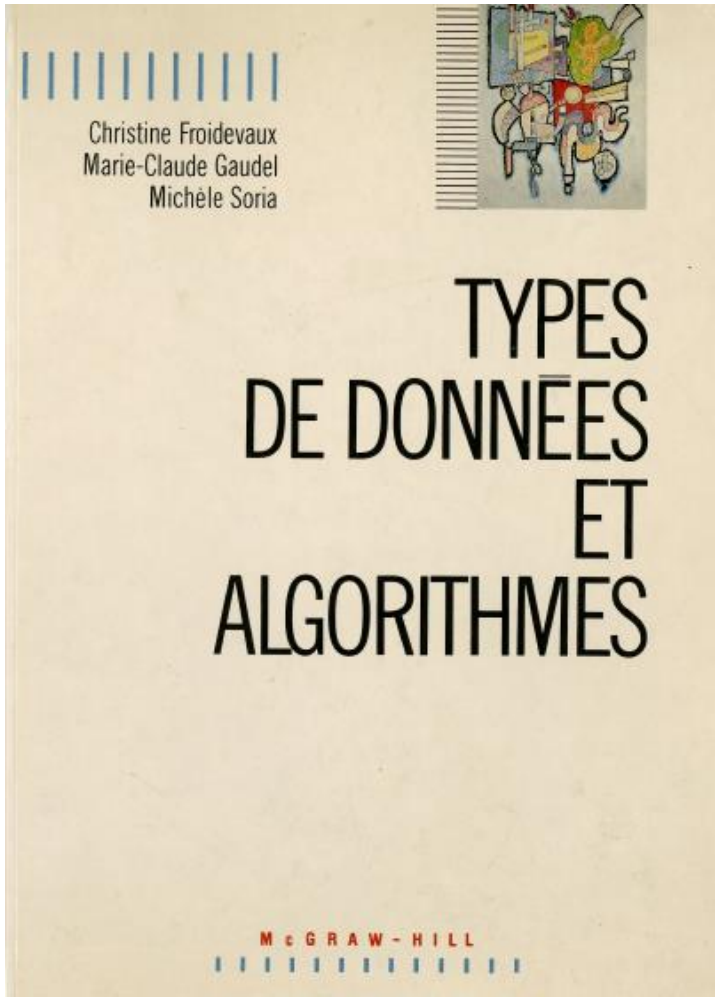
1. Structures de données

Listes, piles, files : structures linéaires. **Dictionnaires, index et clé.**	Distinguer des structures par le jeu des méthodes qui les caractérisent. Choisir une structure de données adaptée à la situation à modéliser. Distinguer la recherche d'une valeur dans une liste et dans un dictionnaire.	On distingue les modes FIFO (first in first out) et LIFO (last in first out) des piles et des files.

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les listes



luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les listes

- Les listes linéaires sont la forme la plus commune d'organisation de données
- Une liste linéaire λ est une suite finie, éventuellement vide, d'éléments repérés selon leur rang dans la liste

$$\lambda = <e_1, e_2, \ldots e_n>$$

- L'ordre des éléments est déterminé par l'ordre des places des éléments.

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Le type abstrait liste itérative

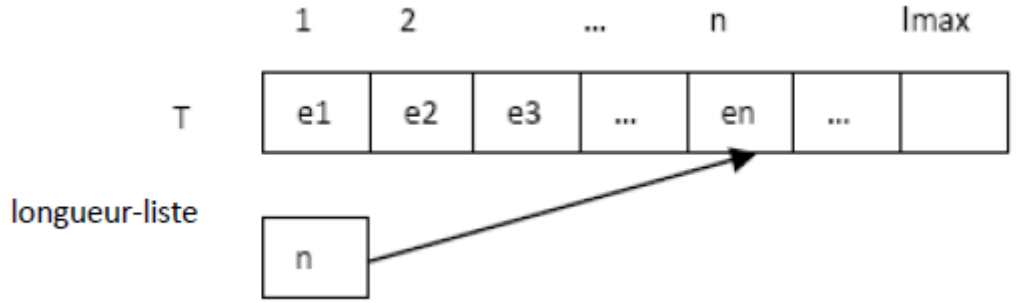
Opérations	Signature	Description
liste_vide	→ liste	*Définir la liste vide*
accéder	liste x position → élément	*Accéder à un élément de la liste; l'élément se trouve à une place (position) donnée dans la liste*
modifier	liste x position x élément → liste	*Modifier un élément de la liste; Le nouvel élément se trouve à une place (position) donnée dans la liste*
longueur	liste → entier	*Calculer la longueur d'une liste*
supprimer	liste x position → liste	*Supprimer un élément à la liste; l'élément à supprimer se trouve à une place donnée dans la liste*
inserer	liste x position x élément → liste	*Ajouter un élément à la liste; l'élément ajouté se trouvera (après ajout) à une place (position) donnée dans la liste*
est-elle vide ?	liste → booléen	*Tester si une liste est vide*

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Représentation des listes

- Représentation contiguë



- Représentation chainée

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau

- Un tableau est une séquence ordonnée et contigüe d'éléments de même type.
- Le contenu d'un tableau est modifiable.

Un tableau en mémoire

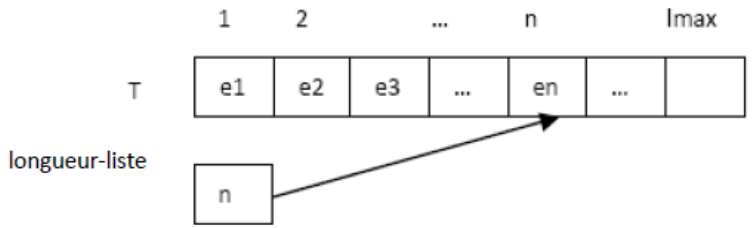
h	e	l	l	o	!					
	3						9			
								6		
h	e	y	8	5	3	9	1	0	2	!
	3	4								

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau

- La liste est représentée par un tableau dont la $i_{ème}$ case est à la $i_{ème}$ place de la liste
- La longueur de la liste ne peut pas dépasser la taille du tableau qui est surdimensionné
- La longueur de la liste est représenté par un entier
- La liste est donc représentée par un couple <tableau, entier>



Liste

Tableau T	**indice**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	valeur	8	5	3	9	1	0	2		

Longueur	**7**

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau

- Il est simple d'accéder au $k_{ième}$ élément d'une liste
- On accède à un élément en temps constant grâce à son indice.

tableau	indice	0	1	2	3	4	5	6	7	8
	valeur	8	5	3	9	1	0	2		

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau

- Il est simple d'insérer ou de supprimer le dernier élément de la liste
- Si l'élément à insérer ou supprimer est à la $k_{ième}$ place de la liste il faut déplacer tous les éléments d'indice k à l'indice Liste(long) dans le tableau.

indice	0	1	2	3	4	5	6	7	8
valeur	8	5	3	9	1	0	2		

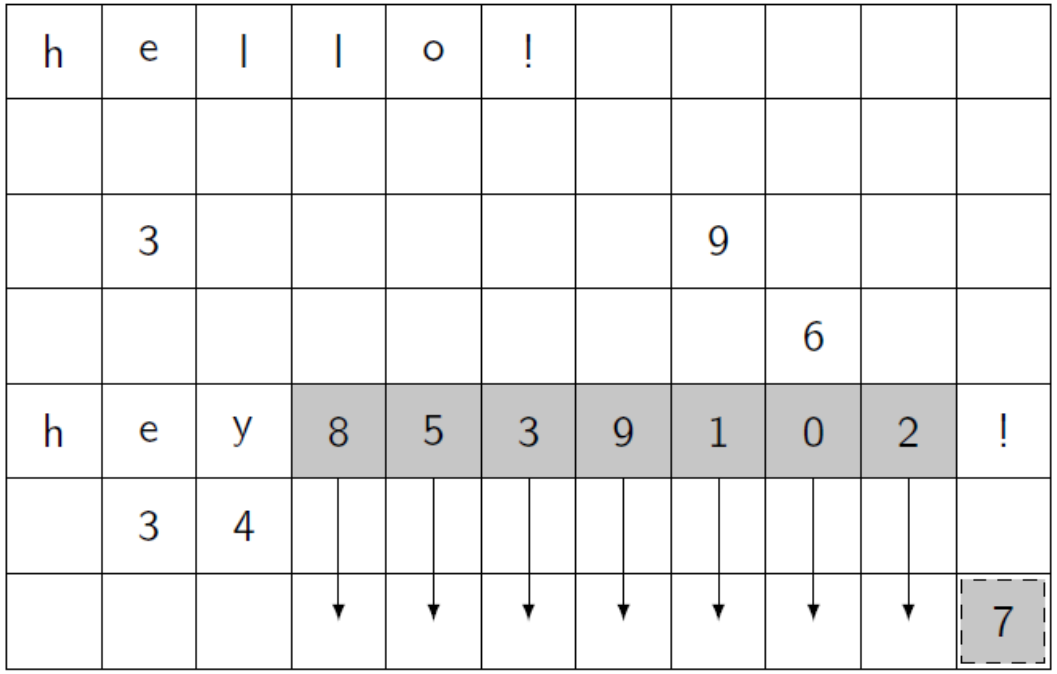
indice	0	1	2	3	4	5	6	7	8
valeur	8	5	3	9	6	1	0	2	

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau

- Agrandir un tableau peut prendre un temps linéaire



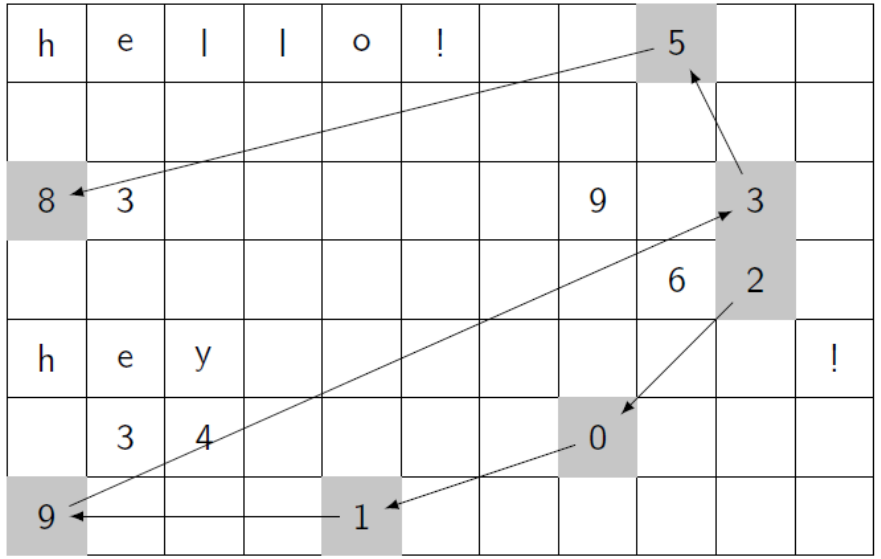
luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Liste chainée

- Une liste chaînée est une séquence ordonnée où chaque élément possède une référence vers le suivant.

Une liste chainée en mémoire

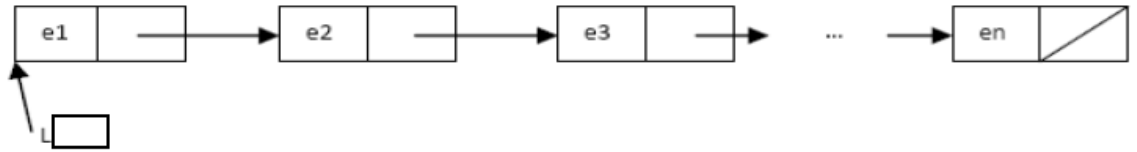


luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Liste chainée

- On utilise des pointeurs pour chainer les éléments successifs
- La liste est déterminée par l'adresse de son premier élément



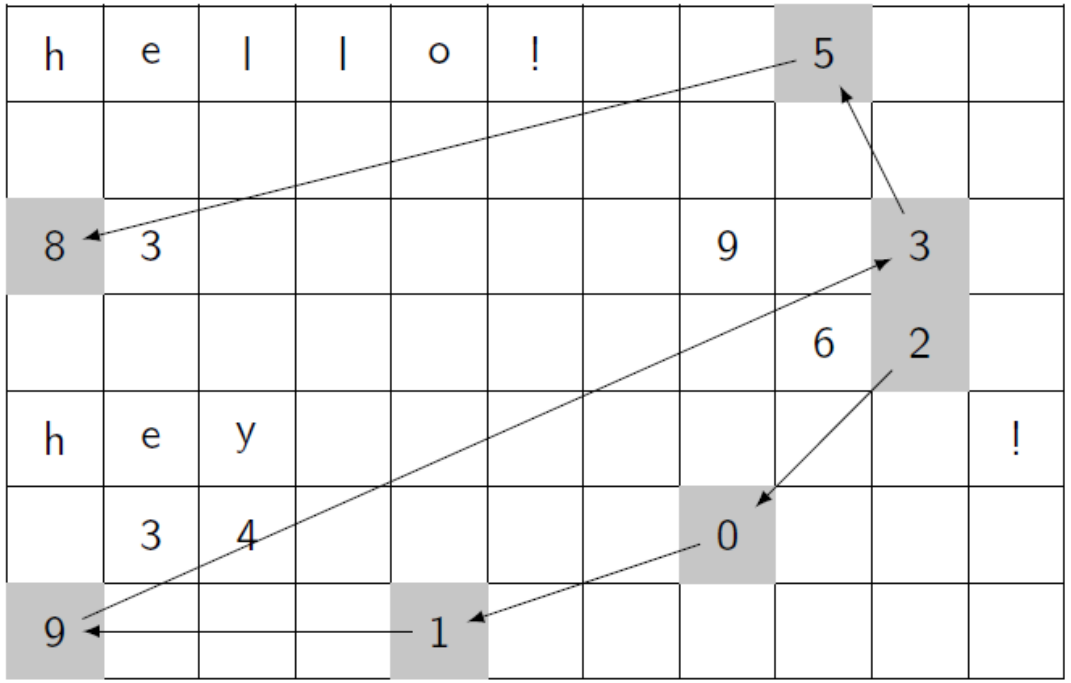
- Remarques :
  - Cela nécessite de la place supplémentaire en mémoire
  - Il n'y a pas de taille maximum pour la liste
  - La suppression ou l'insertion sont simples

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Liste chainée

- On accède à l'élément de rang n en temps linéaire.

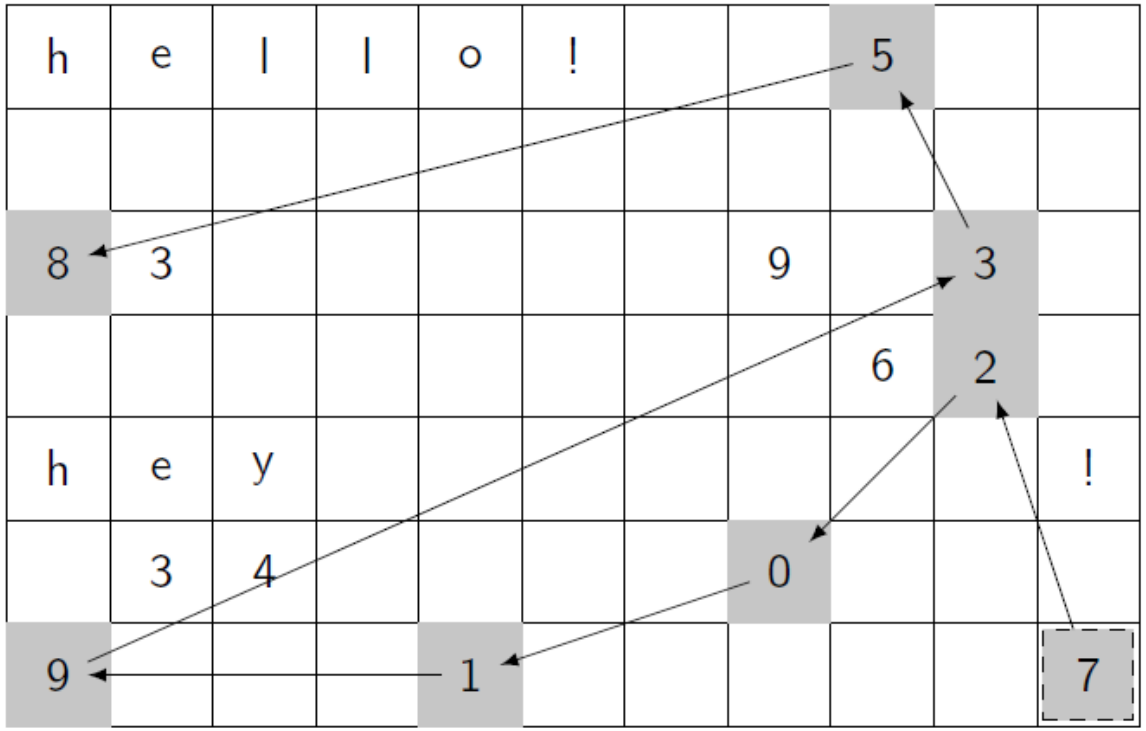


luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Liste chainée

- On ajoute un élément en tête de liste en temps constant.



luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les list python

- Python est un langage de haut-niveau.
- Le type list essaie de prendre les avantages des deux structures précédentes.
- Les list vont plus loin : les éléments peuvent être de types différents.

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les piles et les files

- Pour beaucoup d'applications les seules opérations à effectuer sur les listes sont des insertions et des suppressions aux extrémités.
- Piles (LIFO): les insertions et les suppressions se font à une seule extrémité de la file appelée sommet.
- Files (FIFO): On fait les insertions à une extrémité, les accès et les suppressions à l'autre extrémité.

luc.vincent@ac-bordeaux.fr

Le type abstrait Pile

Opérations	Signature	Description
pile_vide	→ pile	*Définir la pile vide*
empiler	pile x élément → pile	*Ajouter un élément à la pile; l'élément ajouté se trouvera (après ajout) au sommet de la pile*
depiler	pile → pile	*Supprimer l'élément au sommet de la pile*
sommet	pile → élément	*Accéder au sommet de la pile*
est-elle vide ?	pile → booléen	*Tester si une pile est vide*

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

luc.vincent@ac-bordeaux.fr

Le type abstrait File

Opérations	Signature	Description
file_vide	→ file	*Définir la file vide*
enfiler	file x élément → file	*Ajouter un élément à la file; l'élément ajouté se trouvera (après ajout) en queue de la file*
defiler	file → file	*Supprimer l'élément en tête de la file*
premier	file → élément	*Accéder à l'élément en tête de la file (présent depuis le plus longtemps)*
est-elle vide ?	pile → booléen	*Tester si une file est vide*

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Les dictionnaires

- Dans les listes, il y a un ordre sur les places et les éléments sont donc ordonnés.
- Dans un dictionnaire, la place d'un élément est calculée uniquement à partir de sa clé.
- Ce calcul est réalisé grâce à une fonction dite fonction de hash
- Ces méthodes ont la particularité de rendre la complexité des opérations de recherche, d'ajout ou de suppression en moyenne constante.

luc.vincent@ac-bordeaux.fr

Le type abstrait dictionnaire

Opérations	Signature	Description
dictionnaire_vide	→ dictionnaire	*Définir le dictionnaire vide*
accéder	dictionnaire x clé → élément	*Accéder à un élément du dictionnaire; l'élément se trouve à une position définie par sa clé*
modifier	dictionnaire x clé x élément → dictionnaire	*Modifier un couple clé:valeur en remplaçant la valeur courante par une autre valeur (la clé restant identique)*
supprimer	dictionnaire x clé → dictionnaire	Supprimer une clé (et donc la valeur qui lui est associée)
ajouter	dictionnaire x clé x élément → dictionnaire	*Associer une nouvelle valeur à une nouvelle clé*

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Fonction de hashe

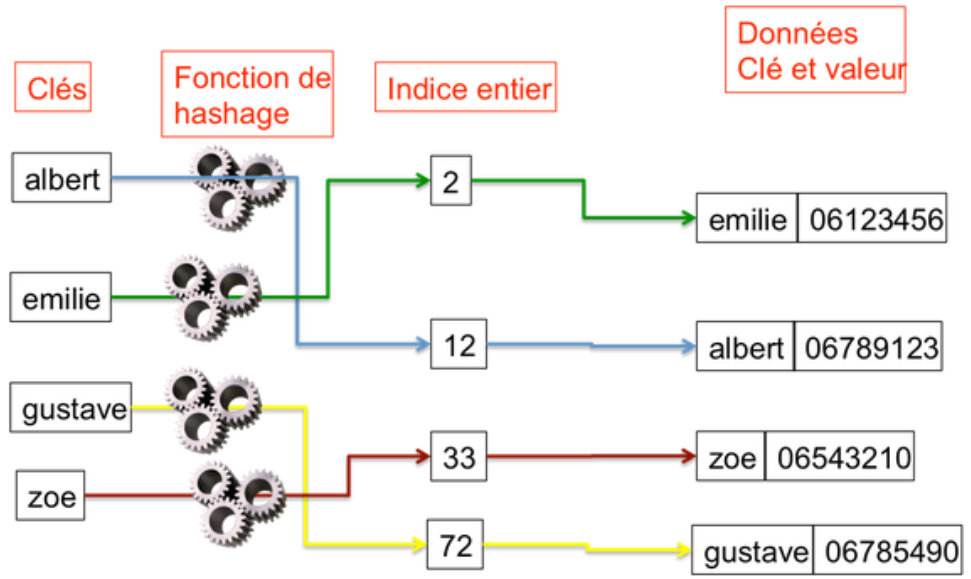
- Nous cherchons à indexer un ensemble de valeurs (i.e les numéros de téléphones) par un ensemble de clés (i.e les noms).
- Ceci n'est bien sûr pas possible avec un tableau dont les indices sont obligatoirement des entiers (afin d'être capable de calculer l'adresse d'un élément).

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Fonction de hashage

- Le premier principe essentiel des tables de hashage est de transformer une clé en un entier, qui sera alors l'indice du tableau contenant les données. C'est le rôle de la fonction de hashage, qui assure une correspondance entre notre clé et l'indice entier du tableau.



luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Table d'association

- Ajouter un couple clé, valeur c’est
  - Calculer l'indice que ce couple occupera dans la table, c'est à dire exécuter la fonction de hashage pour récupérer la valeur entière qu'elle retourne ;
  - Mettre ce couple [clé, valeur] dans la table à l'indice qui vient d'être calculé

La table initiale est vide

	Clé	Valeur
0		
1		
....		
6		
...		
31		
...		
99		

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Table d'association

- Rechercher un couple clé, valeur c'est
  - Calculer l'indice que ce couple occupe s'il existe dans la table, à l'aide de la fonction de hashage ;
  - S'ils existent, la clé et la valeur sont dans la table à l'indice qui vient d'être calculé.

on calcule l'indice de hashage

hash("bac")=(2+1+3)%100=6

	Clé	Valeur
0		
....		
6	bac	examen passé en terminale
...		
31	lame	partie coupante d'un couteau
...		
42		
...		
99		

La case d'indice *6* de la table n'est pas vide et contient la clé *bac* et sa définition

on calcule l'indice de hashage

hash("trac")=(20+18+1+3)%100=42

	Clé	Valeur
0		
....		
6	bac	examen passé en terminale
...		
31	lame	partie coupante d'un couteau
...		
42		
...		
99		

La case d'indice *42* de la table est vide et la table ne contient aucune information sur le mot "truc".

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Parcours de listes et dictionnaires

- Difficultés rencontrées

```
1  # confusion indice / élément
2  tab = [3, 5, 7]
3  for i in tab:
4      print(tab[i])
5
```

Python Tutor

IndexError: list index out of range

```
1  # indice dans un dictionnaire
2  dico = {"a":0, "b":1, "c":2}
3  for i in range(len(dico)):
4      print(dico[i])
5
```

Python Tutor

KeyError: 0

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau parcours par indice

Pour variable entière variant de valeur de début à valeur de fin par pas de taille faire …

```python
# Parcours par indice
tab = [3, 5, 7]
for i in range(0, len(tab), 1):
    print(tab[i])

```

Python Tutor

```java
public class Main
{
    public static void main(String[] args) {
        int[] tab = {3, 5 ,7};
        for (int i=0; i<tab.length; i=i+1){
          System.out.println(tab[i]);
        }
    }
}
```

Java Tutor

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Tableau parcours par éléments

Pour chaque élément dans le tableau …

```python
# Parcours par éléments
tab = [3, 5, 7]
for e in tab:
    print(e)

```

Python Tutor

```java
public class Main
{
    public static void main(String[] args) {
        int[] tab = {3, 5 ,7};
        for (int e:tab){
          System.out.println(e);
        }
    }
}
```

Java Tutor

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Dictionnaire parcours par clé

Pour chaque clé du dictionnaire faire …

```python
# clé dans un dictionnaire
dico = {"a":0, "b":1, "c":2}
for k in dico.keys():
    print(dico[k])

```

Python Tutor

```java
import java.util.*;

public class Main
{
    public static void main(String[] args) {
      Hashtable dic=new Hashtable();
      dic.put("a",0);
      dic.put("b",1);
      dic.put("c",2);
      Set<String> setOfKeys = dic.keySet();
      for (String key: setOfKeys){
        System.out.println(dic.get(key));
      }
    }
}

```

Java Tutor

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Dictionnaire parcours par éléments

Pour chaque élément du dictionnaire faire …

```python
# éléments dans un dictionnaire
dico = {"a":0, "b":1, "c":2}
for e in dico.values():
    print(e)

```

Python Tutor

```java
import java.util.*;
import java.util.Map.Entry;

public class Main
{
    public static void main(String[] args) {
      Hashtable dic=new Hashtable();
      dic.put("a",0);
      dic.put("b",1);
      dic.put("c",2);
      Set<Entry<String, Integer> > entrySet = dic.entrySet();
      for (Entry<String, Integer> entry : entrySet){
        System.out.println(entry.getValue());
      }
    }
}

```

Java Tutor

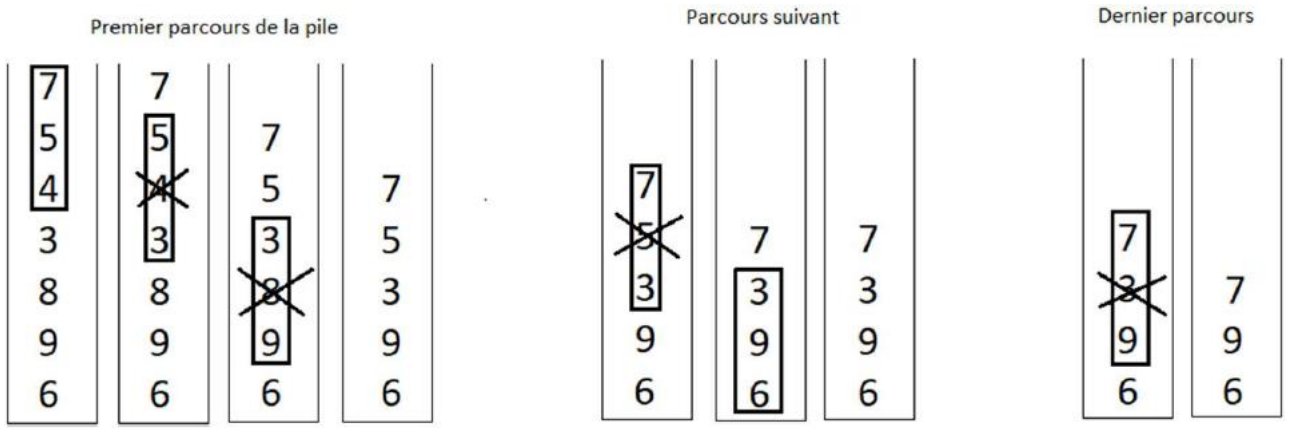
luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Muable ou immuable

- Baccalauréat 22_NSIJ2ME1

La poussette est un jeu de cartes en solitaire. Cet exercice propose une version simplifiée de ce jeu basée sur des nombres.
On considère une pile constituée de nombres entiers tirés aléatoirement. Le jeu consiste à réduire la pile suivant la règle suivante : quand la pile contient du haut vers le bas un triplet dont les termes du haut et du bas sont de même parité, on supprime l'élément central.

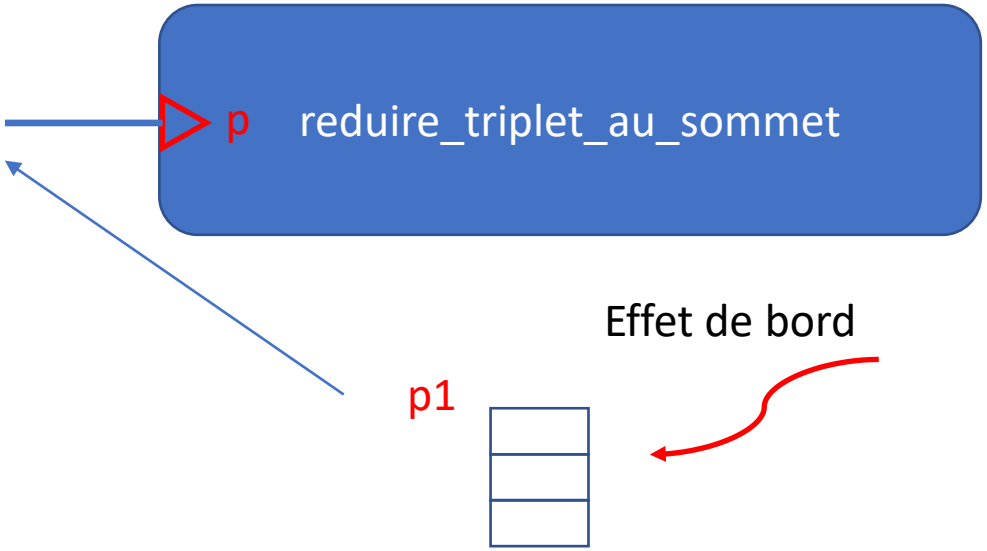


luc.vincent@ac-bordeaux.fr

Muable ou immuable

- Baccalauréat

- « Recopier et compléter sur la copie le code de la fonction reduire_triplet_au_sommet prenant une pile p en paramètre et qui la modifie en place. Cette fonction ne renvoie donc rien. »




```
def reduire_triplet_au_sommet(p):
    """Reduction de la pile p

    permet de supprimer l'élément central des trois premiers éléments
    en partant du haut de la pile,
    si l'élément du  bas et du haut sont de même parité.
    Les éléments dépilés et non supprimés sont replacés
    dans le bon ordre dans la pile.
    """
    # la pile p étant une liste c'est un objet muable
    # Cette fonction réduit le sommet de la pile en place
    a = depiler(p)
    b = depiler(p)
    c = sommet(p)
    if a % 2 != c % 2:
        empiler(p, b)
    empiler(p, a)
```

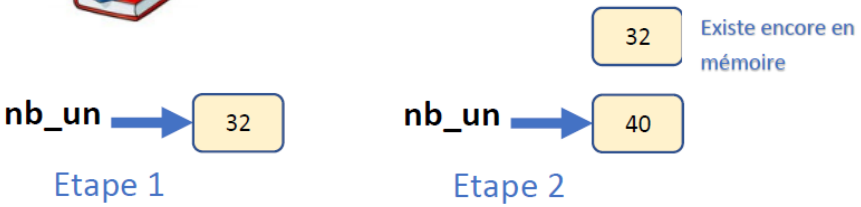
This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

luc.vincent@ac-bordeaux.fr

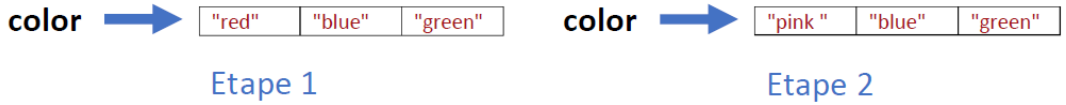
Muable ou immuable

- « An immutable object is an object whose value cannot change. »
- « In Python, 'mutable' is the ability of objects to change their values »

Python Tutor



Python Tutor



fonction

Se comporte comme « passage par valeur »

Python Tutor

« passage par référence »

Python Tutor

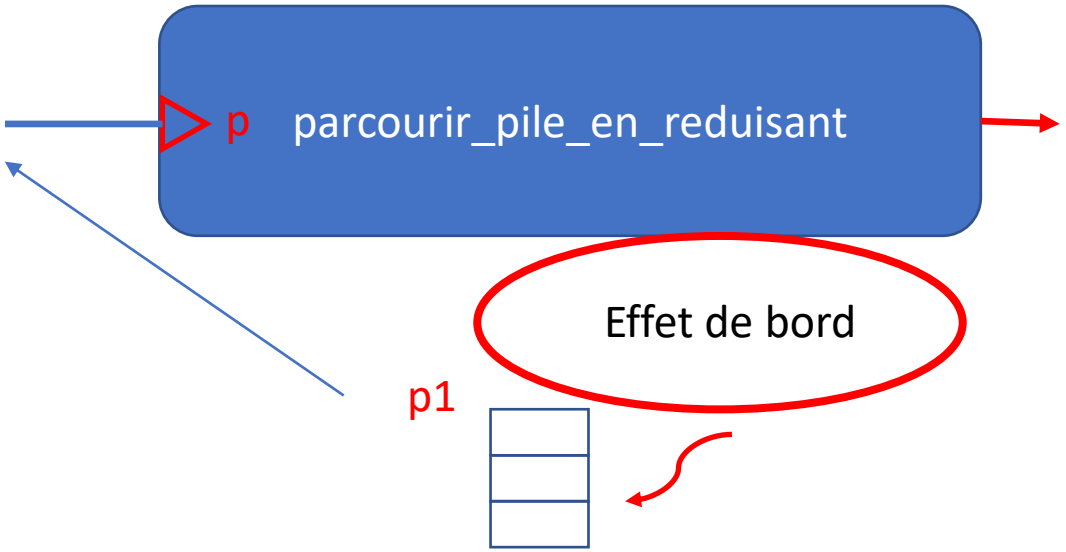
luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Muable ou immuable

- Baccalauréat

- «On se propose maintenant d'écrire une fonction parcourir_pile_en_reduisant qui parcourt la pile du haut vers le bas en procédant aux réductions pour chaque triplet rencontré quand cela est possible.»



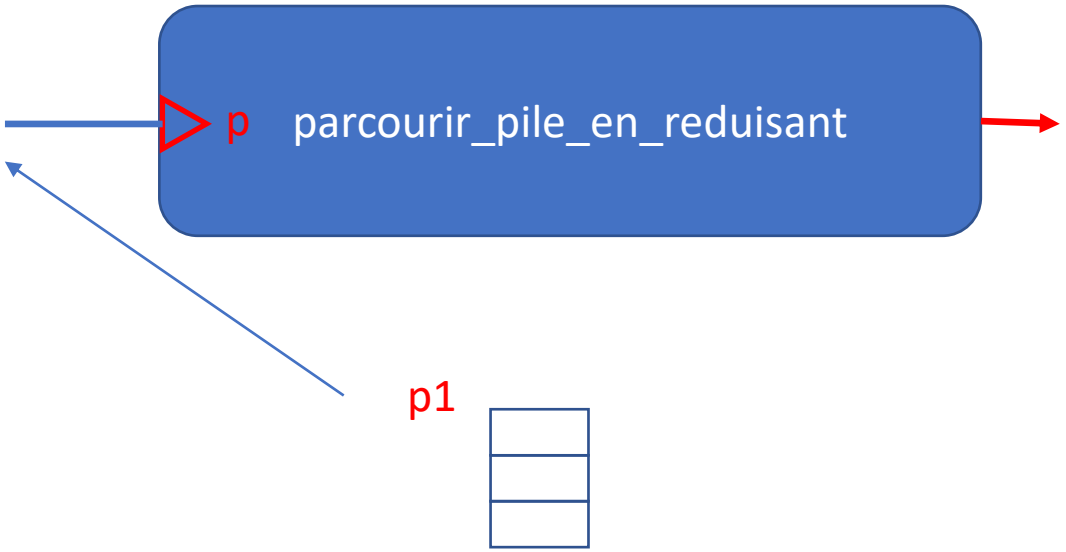

```
1  def parcourir_pile_en_reduisant(p):
2      q = creer_pile_vide()
3      while taille(p) >= ....:
4          reduire_triplet_au_sommet(p)
5          e = depiler(p)
6          empiler(q, e)
7      while not est_vide(q):
8          ..............
9          ..............
10     return p
```

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

luc.vincent@ac-bordeaux.fr

Muable ou immuable

- Bonne méthode
- Dans un projet ne JAMAIS mélanger les modifications en place et les fonctions pures (sans effet de bord)
- Dans le cas d'objet muable faire une copie de l'objet pour éviter les effets de bord !



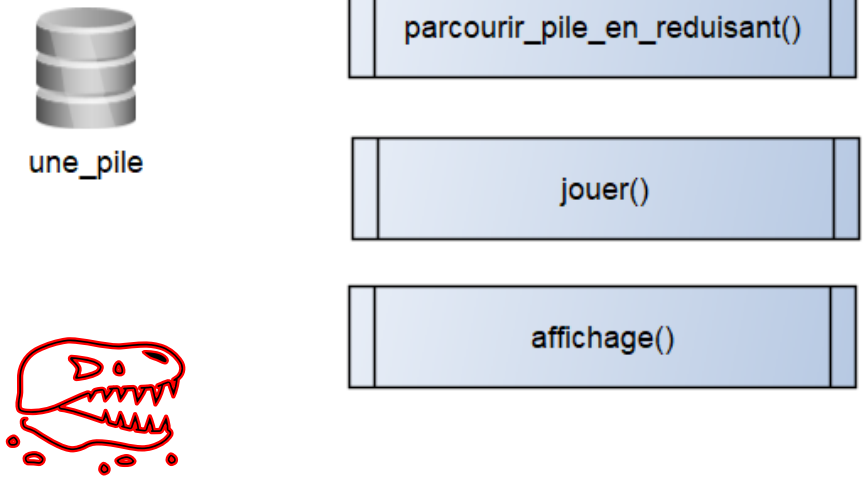

```
def reduire_triplet_au_sommet(p):
    """Reduction de la pile p
    permet de supprimer l'élément central des trois premiers éléments
    en partant du haut de la pile,
    si l'élément du  bas et du haut sont de même parité.
    Les éléments dépilés et non supprimés sont
    replacés dans le bon ordre dans la pile.
    """
    # la pile p étant une liste c'est un objet muable
    # On commence par en faire une copie (superficielle)
    # p = [p[i] for i in range(len(p))]
    # p = list(p)
    p = p.copy()
    a = depiler(p)
    b = depiler(p)
    c = sommet(p)
    if a % 2 != c % 2:
        empiler(p, b)
    empiler(p, a)
    return p
```

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Muable ou immuable

- Procédure uniquement
  - Variable globale connue de chaque procédure




```
# une_pile est une variable globale connue de toutes les fonctions
une_pile = creer_pile_vide()
poussette = (6, 9, 8, 3, 4, 5, 7)
for value in poussette:
    empiler(une_pile, value)
jouer()
affichage()
```

```
86 def jouer():
87     """Simplifier entièrement la pile"""
88     # la pile p étant une liste c'est un objet muable
89     # Cette fonction simplifie en place la pile une_pile
90     # la variable est globale "une_pile"
91     taille_p_ar = taille(une_pile) # taille_p_ar avant reduction
92     parcourir_pile_en_reduisant()
93     if taille(une_pile) != taille_p_ar:
94         jouer()
95
```

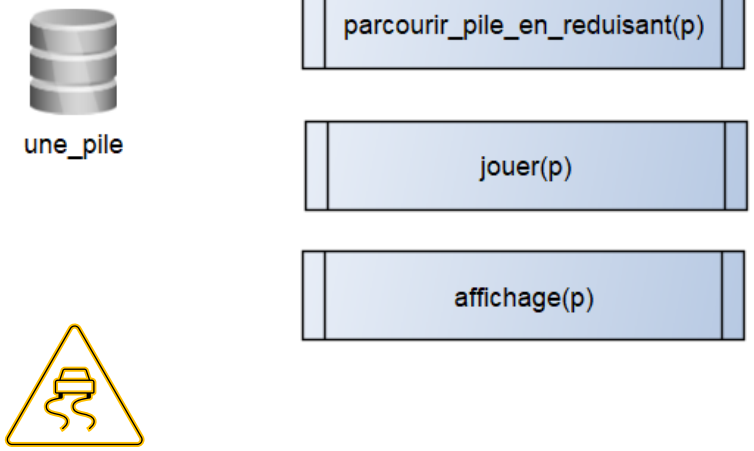
Exercice_2_E_pile_avec_effetDeBord_varGlobale

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Muable ou immuable

- Procédure avec effet de bord
  - Référence passée comme argument
  - Renvoi par effet de bord et sans « return»




```
# une_pile est passée par référence
# à chaque fonction
une_pile = creer_pile_vide()
poussette = (6, 9, 8, 3, 4, 5, 7)
for value in poussette:
    empiler(une_pile, value)
jouer(une_pile)
affichage(une_pile)
```

```
def jouer(p):
    """Simplifier entièrement la pile"""
    # la pile p étant une liste c'est un objet muable
    # Cette fonction simplifie la pile en place
    taille_p_ar = taille(p) # taille_p_ar avant reduction
    parcourir_pile_en_reduisant(p)
    if taille(p) != taille_p_ar:
        jouer(p)
```

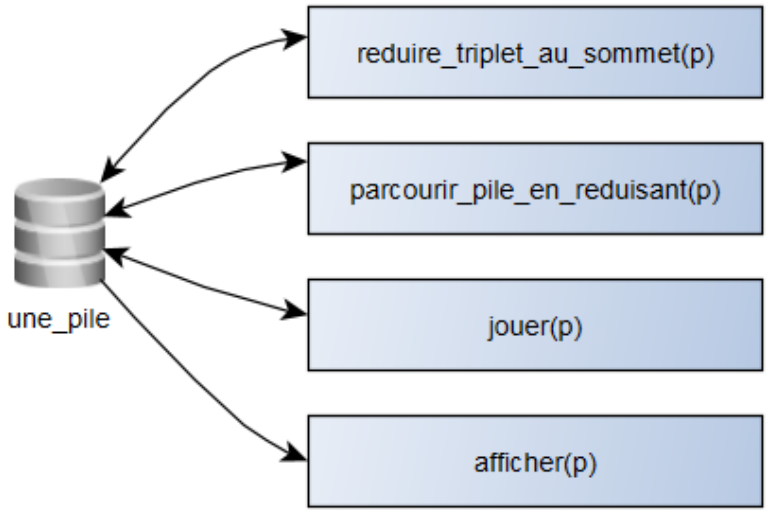
Exercice_2_C_avec_EffetDeBord_AvecArgument

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Muable ou immuable

- Fonction avec effet de bord
  - Référence passée comme argument
  - Renvoi par effet de bord et sans « return»



```
poussette = (1, 6, 7, 8, 4, 3)
une_pile = creer_pile_vide()
for value in poussette:
    empiler(une_pile, value)
une_pile_j = jouer(une_pile)  # fonction
affichage(une_pile_j)
```

```
def jouer(p):
    """Renvoyer une pile entièrement simplifiée"""
    q = parcourir_pile_en_reduisant(p) # fonction
    if q == p :
        return q
    else:
        return jouer(q)
```

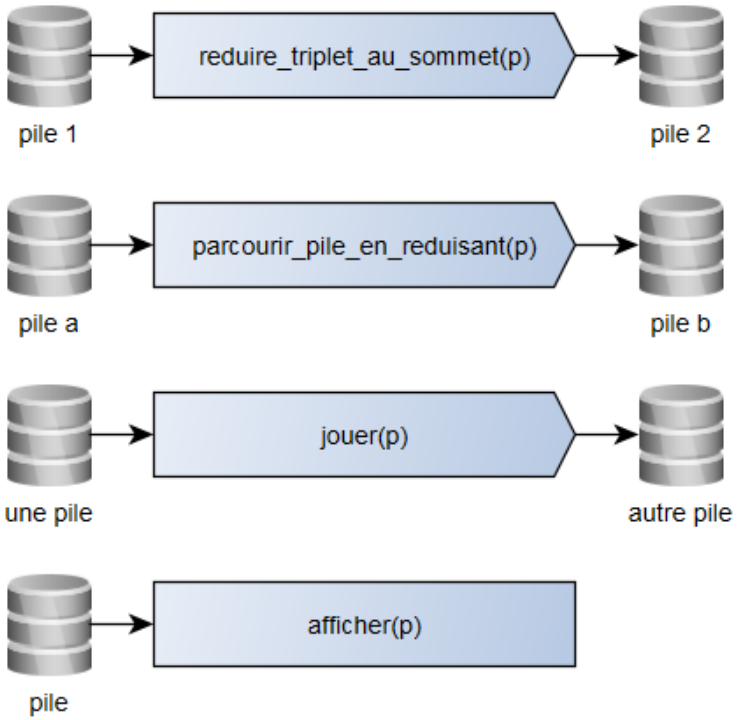
Exercice_2_F_pile_avec_effetDeBord_AvecArgument

luc.vincent@ac-bordeaux.fr

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Muable ou immuable

- Fonction pure
  - Référence passée comme argument
  - Renvoi d'un nouvel objet sans modifier l'objet initial




```
poussette = (6, 9, 8, 3, 4, 5, 7)
une_pile = creer_pile_vide()
for value in poussette:
    empiler(une_pile, value)
une_pile_j = jouer(une_pile)  # fonction
affichage(une_pile_j)
```

```
def jouer(p):
    """Renvoyer une pile entièrement simplifiée"""
    q = parcourir_pile_en_reduisant(p) # fonction
    if q == p :
        return q
    else:
        return jouer(q)
```

Exercice_2_D_pile_avec_effetDeBord_AvecArgument

luc.vincent@ac-bordeaux.fr