Python

Algorithmes sur les tableaux ou listes (suite de valeurs successives)

Valérie Bellynck (2009),
d'après un diaporama d'Alain Bozzi (2008),
lui-même venant de celui de Bob Cordeau (2008)

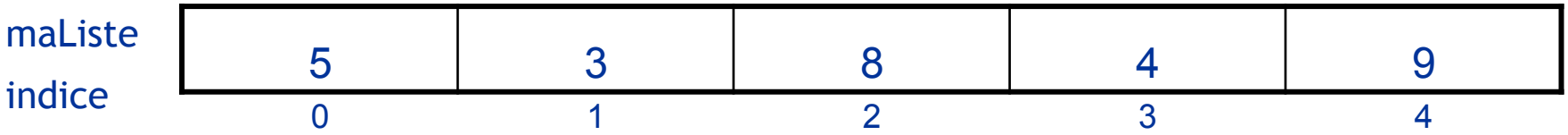
Définition d'une liste

Liste

dans le langage python on parle plutôt de listes que de tableaux.

suite de valeurs dont le type peut être identique ou différent.

chaque valeur et est repérée par un indice.



Exemples :

maListe = [5,3,8,4,9] # 5 éléments

fruits = ['pomme', 'orange', 'figue', 'raisin']

listeDiverses = ["toto", 5, "fleurs", 32.50]

Accès aux éléments d'une liste

- **Accès**
  - l'accès en lecture se fait de la manière suivante :

variable = **nomListe[i]**

  - l'accès en écriture se fait de la manière suivante :

nomListe[i] = valeur

- **Exemple 1 :**

```
maListe[5,8,12,4]

valeur = maListe[0]

print valeur # affiche 5
```

Accès aux éléments d'une liste

- **Exemple 2 :**

```
maListe[5,8,12,4]

maListe[1] = 3 # modification de la liste 8 -> 3

print maListe[1] # affiche 3
```

Parcours d'une liste avec `for`

```
maListe = [5,3,8,4,9]
for i in maListe :
   print i

couleurs = [“cyan”, “magenta”, “jaune”, “noir”]
for i in couleurs :
   print i

fruits = [‘pomme’, ‘orange’, ‘figue’, ‘raisin’]
for i in fruits :
   print i
```

Actions sur les listes

Dans le langage Python, il existe des fonctions qui permettent certaines actions sur les listes.

- **sort()** pour trier une liste

```
maListe = [5,3,8,4,9]
maListe.sort() # [3,4,5,8,9]
```

- **append()** pour ajouter un élément à la fin d'une liste

```
maListe = [3,4,5,8,9]
maListe.append(12) # [3,4,5,8,9,12]
```

Actions sur les listes

- **remove()** pour supprimer un élément à une liste

```
maListe = [3,4,5,8,9,12]
maListe.remove(5) # [3,4,8,9,12]
```

- **reverse()** pour inverser l'ordre d'une liste

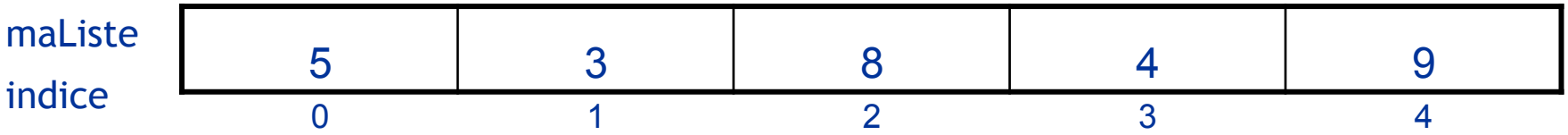
```
maListe = [3,4,8,9,12]
maListe.reverse() # [12,9,8,4,3]
```

Remplissage d'une liste avec un for

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
"*****************************"
listeEntier = []            # liste vide
for i in range(5) :
   valeur = int(raw_input("Saisir un entier "))
 # on ajoute la valeur en fin de liste
   listeEntier.append(valeur)
# affichage de la liste
lg = len(listeEntier)     # longueur de la liste
for i in range(lg) :
   print listeEntier[i]
```

Recherche séquentielle dans une liste

- La recherche dans une liste fait apparaître deux étapes :
  - un parcours
  - une action conditionnelle



- le parcours

```
while (indice < lg_maListe)
```

- l'action conditionnelle

```
maListe[i] == elem
```

Recherche séquentielle dans une liste non triée

```
liste = [5,8,3,7]
lg = len(liste) # 4
i = 0 # indice de la liste initilisé à 0

# parcours action conditionnelle
while (i < lg and liste[i] != elem) :
i = i + 1  # incrémentation

trouve = i < lg
```

- **En sortie de boucle, nous avons :**

  soit i = lg et elem n'est pas dans la liste

  soit i < lg et elem est dans la liste

Recherche séquentielle dans une liste non triée

- **Utilisation d'une sentinelle : on place la valeur recherchée (elem) en fin de liste, ce qui permet de supprimer le teste de `i < lg`**

```
liste = [5,8,3,7,elem]
i = 0 # initilisation

#  action conditionnellle
while (liste[i] != elem) :
    i = i + 1  # incrémentation

trouve = i < lg
```

- **En sortie de boucle, nous avons :**

  soit liste[i] = elem avec i = lg => elem n'est pas dans la liste

  soit liste[i] = elem avec i < lg => elem est dans la liste

Recherche séquentielle dans une liste triée

- Dans une liste triée, il y a une relation d'ordre : liste[i-1] <= liste[i]
- Nous devons en tenir compte dans notre algorithme de recherche.

```
liste = [3,5,7,8]
lg = len(liste) # 4
i = 0 # initilisation

# parcours et action conditionnellle
while (i < lg and liste[i] < elem) :
    i = i + 1  # incrémentation

trouve = i < lg and liste[i] == elem
```

- En sortie de boucle, nous avons :

  soit i = lg => elem n'est pas dans la liste

  soit i < lg et liste[i] = elem => elem est dans la liste