les structures de données en Python

Les listes

Les listes

D’abord c’est quoi une séquence ??

- Une séquence est un ensemble fini d'éléments indicés de 0 à n-1 (si cette séquence comporte n éléments).

Exemples de séquence:

  - les listes
  - les tuples
  - Les chaines de caractères

- Les séquences partagent tous un comportement commun.

  En particulier,

- **Longueur: len(ch)**
- **Sélection de l'élément : ch[1]**

Définition

- Une liste est une structure de données qui contient une série de valeurs.
- Python autorise la construction de liste contenant des valeurs de type différent (par exemple entier et chaîne de caractères), ce qui leur confère une grande flexibilité.
- Une liste est déclarée par une série de valeurs séparées par des virgules, et le tout encadré par des crochets
- s'il s'agit d'une valeur de type chaînes de caractères, il faut l'encadrer par des guillemets, simples ou doubles

Définition

- Il n’existe pas de **tableaux** en python tel qu’il en existe en C, Java, Pascal…
- les objets placés dans une liste sont accessibles par l'intermédiaire d'un ***index*** (un nombre qui indique l'emplacement de l'objet dans la séquence)
- La numérotation des index commence par **0**

Exemple

```
>>> Modules=["Analyse","Algèbre","Tec","Mécanique","Elecricité","Informatique"]
>>> Modules
['Analyse', 'Algèbre', 'Tec', 'Mécanique', 'Elecricité', 'Informatique']
>>>
```

```
>>> Numero_apogee= [13002577, 13002576, 13672596, 14692562]
>>> Numero_apogee
[13002577, 13002576, 13672596, 14692562]
>>>
```

```
>>> mixte=['Ensak',2008,'kenitra',2.5]
>>> mixte
['Ensak', 2008, 'kenitra', 2.5]
>>>
```

```
>>> Filieres=['Ensak',['GI',2010,140],['INDUS',2014,50]]
>>> Filieres
['Ensak', ['GI', 2010, 140], ['INDUS', 2014, 50]]
>>>
```

Accéder aux éléments d'une liste

- Pour accéder aux éléments d'une liste, on utilise les mêmes méthodes (index, découpage en tranches) que pour accéder aux caractères d'une chaîne :

```
                    -6          -5          -4          -3          -2          -1
>>> modules=["Informatique","Analyse","Tec","Mécanique","Algèbre","Elecricité"]
>>> modules[0]      0           1           2           3           4           5
'Informatique'
```

```
>>> modules[1:3]
['Analyse', 'Tec']
>>> modules[2:]
['Tec', 'Mécanique', 'Algèbre', 'Elecricité']
>>> modules[:2]
['Informatique', 'Analyse']
>>> modules[-1]
'Elecricité'
>>> modules[-2]
'Algèbre'
>>>
```

Modifier une liste

```
>>> nombres = [17, 38, 10, 25, 72]
>>> nombres.sort()              # trier la liste
>>> nombres
[10, 17, 25, 38, 72]
>>> nombres.append(12)          # ajouter un élément à la fin
>>> nombres
[10, 17, 25, 38, 72, 12]
>>> nombres.reverse()           # inverser l'ordre des éléments
>>> nombres
[12, 72, 38, 25, 17, 10]
>>> nombres.index(17)           # retrouver l'index d'un élément
4
>>> nombres.remove(38)          # enlever (effacer) un élément
>>> nombres
[12, 72, 25, 17, 10]
```

Modifier une liste

```
>>> nombres = [17, 38, 10, 25, 72]
>>> nombres.extend([1,2,3]) #Ajout en fin de liste
>>> nombres
[17, 38, 10, 25, 72, 1, 2, 3]
>>> nombres.insert(5,55) #insertion de 555 à l'index 5
>>> nombres
[17, 38, 10, 25, 72, 55, 1, 2, 3]
>>> nombres.pop(5) #supprime l'élement à l'indice 5 et le retourne
55
```

```
>>> nombres = [17, 38, 10, 38, 72]
>>> nombres.count(38) #le nombre d'occurences de la valeur 38
2
```

Les fonctions appliquées aux listes: Supprimer un élément ou une partie d'une liste

- l'instruction **del** , permet d'effacer un ou plusieurs éléments à partir de leur(s) index :

```
>>> nombres=[12, 72, 25, 17, 10]
>>> del nombres[1]
>>> nombres
[12, 25, 17, 10]
>>> del nombres[1:3]
>>> nombres
[12, 10]
```

- la différence entre la méthode **remove()** et l'instruction **del** :
  - **del** travaille avec ***un index*** ou une tranche d'index
  - **remove()** recherche ***une valeur*** (si plusieurs éléments de la liste possèdent la même valeur, seul le premier est effacé)

Les fonctions appliquées aux listes :La taille d’une liste

```
>>> nombres=[12, 72, 25, 17, 25]
>>> len(nombres)
5
```

Techniques de « Slicing » avec l'opérateur []

Techniques de « slicing » avancé pour modifier une liste

- **Insertion d'un ou plusieurs éléments n'importe où dans une liste**

```
                    0   1   2   3   4
>>> nombres=[5,10,15,20,25]
>>> nombres[2:2]=[13] #ajoute 13 à l'index 2
>>> nombres
[5, 10, 13, 15, 20, 25]
```

```
                    0   1   2   3   4
>>> nombres=[5,10,15,20,25]
>>> nombres[3:3]=[17,18,19] # ajoute les élement 17,18,19 à l'index 3
>>> nombres
[5, 10, 15, 17, 18, 19, 20, 25]
```

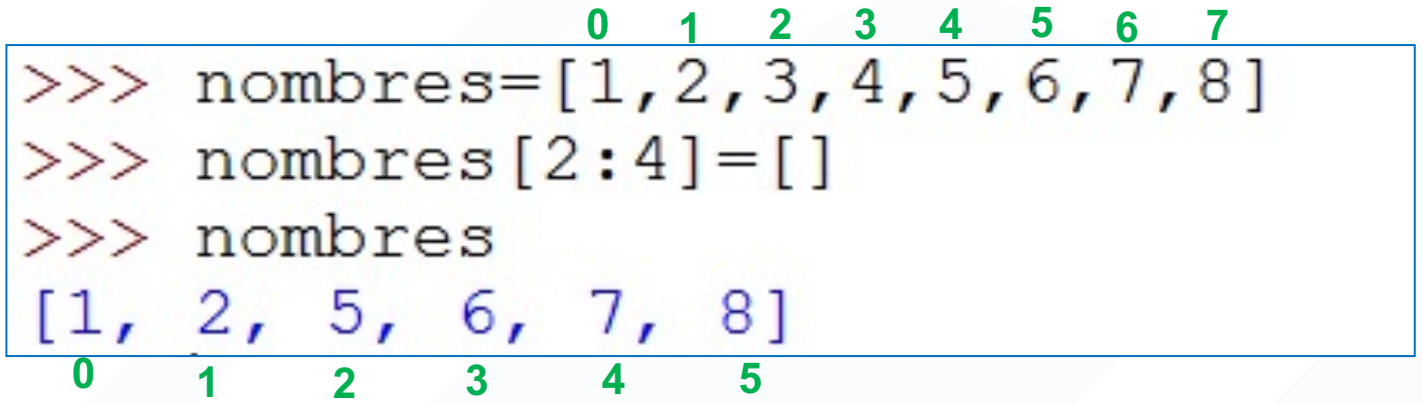
Règles à respecter

- Si vous utilisez l'opérateur **[]** à la gauche du signe égale pour effectuer une insertion ou une suppression d'élément(s) dans une liste, vous devez obligatoirement y indiquer une « tranche » dans la liste cible (c'est-à-dire deux index réunis par le symbole **:**), et non un élément isolé dans cette liste.

```
>>> nombres[2:2]=[13] #ajoute 13 à l'index 2
```

- L'élément que vous fournissez à la droite du signe égale doit lui-même être une liste. Si vous n'insérez qu'un seul élément, il vous faut donc le présenter entre crochets pour le transformer d'abord en une liste d'un seul élément.
- l'élément **nombres[2]** n'est pas une liste alors que l'élément **nombres[2:2]** en est une.

Suppression / remplacement d'éléments



nous remplaçons la tranche [2:4[par une liste vide, ce qui correspond à un effacement

```
>>> nombres[2:5] = [3]
>>> nombres
[1, 2, 3, 8]
```

nous remplaçons la tranche [2,5[par un seul élément. (Notez encore une fois que cet élément doit lui-même être « présenté » comme une liste).

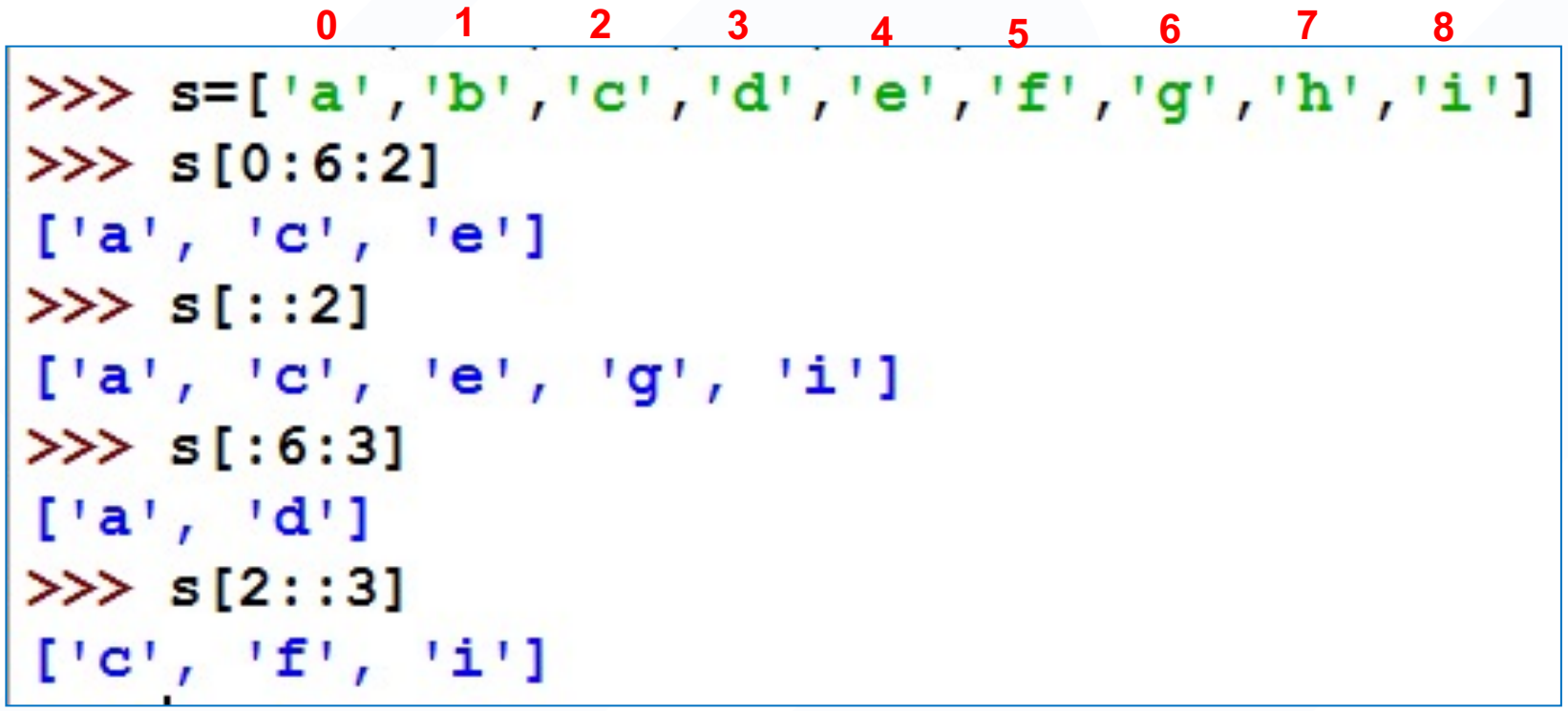
Suppression / remplacement d'éléments

```
>>> nombres=[1,5,15,20]
>>> nombres[1:]=[2,3,4,5,6,7]
>>> nombres
[1, 2, 3, 4, 5, 6, 7]
```

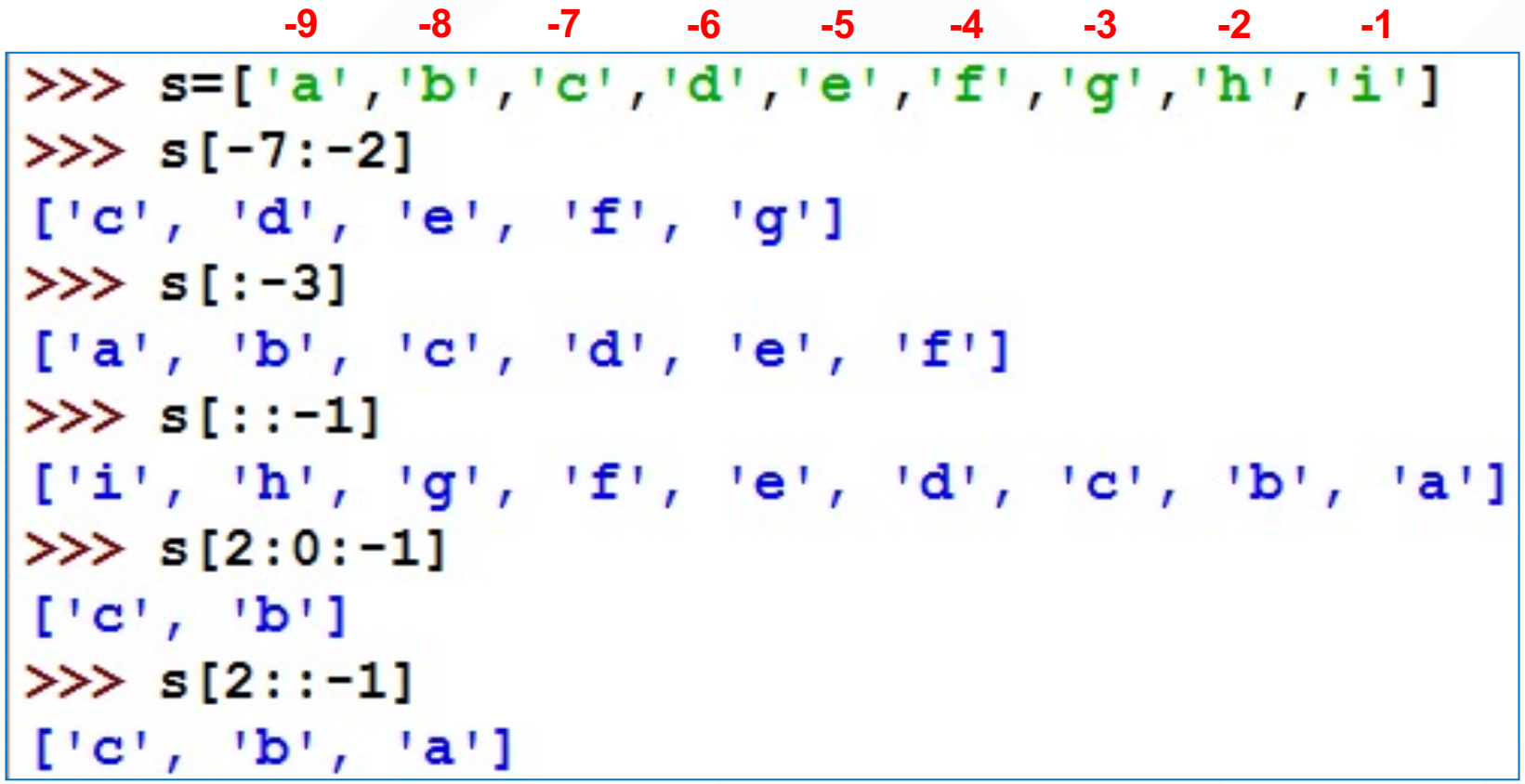
nous remplaçons une tranche de 3 éléments par une autre qui en comporte 6.

```
>>> nombres=[1,5,15,20]
>>> nombres[:2]=[-1,-2]
>>> nombres
[-1, -2, 15, 20]
```

Utilisation d’un pas dans le slicing



Utilisation d'un pas **négatif** dans le slicing



Exercice

liste= [1, 2, 3, 4, 5, 6,7,8]

Remarque : les instructions sont indépendantes

liste.insert(4,0) liste	
liste[::-1]	
liste.pop()	
liste[1:5]=[] liste	
liste[2:2]=22 liste	
liste.index(1)	
liste.remove(1) liste	
del liste[1] liste	

Correction

liste= [1, 2, 3, 4, 5, 6,7,8]

liste.insert(4,0) **liste**	**[1, 2, 3, 4, 0, 5, 6, 7, 8]**
liste[::-1]	[8, 7, 6, 5, 4, 3, 2, 1]
liste.pop()	8
liste[1:5]=[] **liste**	[1, 6, 7, 8]
liste[2:2]=22 **liste**	error
liste.index(1)	0
liste.remove(1) **liste**	[2, 3, 4, 5, 6, 7, 8]
del liste[1] **liste**	[1, 3, 4, 5, 6, 7, 8]

Création de liste avec range

- La fonction **range()** génère une liste de nombres entiers de valeurs croissantes avec un pas égal à 1 (par défaut)
- **range()** attend toujours trois arguments:

```
range(from=1, to, step=1)
```

  - ***FROM*** est la première valeur à générer,
  - ***TO*** est la dernière (ou plutôt la dernière + un),
  - ***STEP*** le « pas » à sauter pour passer d'une valeur à la suivante.

  S'ils ne sont pas fournis, les paramètres *FROM* et *STEP* prennent leurs valeurs par défaut, qui sont respectivement 0 et 1.

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(3,20,3))
[3, 6, 9, 12, 15, 18]
```

Parcours d'une liste à l'aide de for, range() et len()

On peut parcourir une liste de deux manière:
soit en parcourant directement la liste :

```
>>> etab = ['Ecole','Nationale','des','Sciences','Appliquées','kenitra']
>>> for i in etab:
        print(i,end=' ')


Ecole Nationale des Sciences Appliquées kenitra
```

soit en parcourant les indices de la liste :

```
>>> etab = ['Ecole','Nationale','des','Sciences','Appliquées','kenitra']
>>> for i in range(len(etab)):
        print (i,etab[i])


0 Ecole
1 Nationale
2 des
3 Sciences
4 Appliquées
5 kenitra
```

Test d'appartenance

- Nous pouvons déterminer si un élément fait partie d'une liste à l'aide de l'instruction **in** :

```
>>> liste=[1,4,7,12]
>>> if 4 in liste:
        print('ok')


ok
```

```
>>> if 5 not in liste:
        print("valeur non trouvée")

valeur non trouvée
```

Copie superficielle d'une liste

```
>>> liste1=[1,4,7,12]
>>> liste2=liste1
>>> id(liste1)
48737576
>>> id(liste2)
48737576
>>> liste1[0]=555
>>> liste1
[555, 4, 7, 12]
>>> liste2
[555, 4, 7, 12]
```

les noms liste1 et liste2 désignent tous deux un seul et même objet en mémoire. Pour décrire cette situation, on dis que le nom liste2 est un alias du nom liste1

Copie réelle d'une liste

```
>>> liste1=[1,2,3,4]
>>> liste2=list(liste1)
>>> id(liste1)
50813568
>>> id(liste2)
50813488
```

ou

```
>>> liste1=[1,2,3,4]
>>> liste2=liste1.copy()
>>> id(liste1)
48990608
>>> id(liste2)
48585704
```

Exercice

1. Ecrire une fonction qui permet de créer une liste de n élément et la retourner
2. Ecrire une procédure qui permet d'afficher les éléments d'une liste

Solution 1

```
def remplir_list(m):
    "c'est une fonction qui permet de remplir une liste"
    tab=[]
    for i in range(m):
        print("entrer la valeur",i+1,end=' : ')
        a=int(input())
        tab.append(a)
    return(tab)
def affichage(t):
    "Procedure qui affiche les éléments d'une liste"
    for i in range(len(t)):
        print('t[',i,']= ',t[i])

#program principal

n=int(input("Entrer la taille : "))
t=remplir_list(n)
affichage(t)
input()
```

```
Entrer la taille : 4
entrer la valeur 1 : 23
entrer la valeur 2 : 4
entrer la valeur 3 : 5
entrer la valeur 4 : 6
t[ 0 ]=  23
t[ 1 ]=  4
t[ 2 ]=  5
t[ 3 ]=  6
```

Solution 2

```python
def remplir_list(m):
    "c'est une fonction qui permet de remplir une liste"
    tab=[0]*m
    for i in range(m):
        print("entrer le valeur ",i+1,end=' : ')
        tab[i]=int(input())
    return tab

def affichage(t):
    "Procedure qui affiche les éléments d'une liste"
    for i in t:
        print(i,end=' ')

#program principal

n=int(input("Entrer la taille : "))
t=remplir_list(n)
affichage(t)
input()
```

```
Entrer la taille : 4
entrer le valeur  1 : 34
entrer le valeur  2 : 2
entrer le valeur  3 : 8
entrer le valeur  4 : 9
34 2 8 9
```

Le module random

Voici quelques fonctions fournies par le module `random` :

`randrange(a,b,k)`	Choisit un entier aléatoirement dans `range(a,b,k)`
`randint(a,b)`	Choisit un entier aléatoirement dans $[[a, b]]$
`choice(List)`	Choisit un entier aléatoirement dans la liste `List`
`random()`	Choisit un float aléatoirement dans $[0, 1[$
`uniform(a,b)`	Choisit un float aléatoirement dans $[a, b[$

Exemple

```
from random import *
def list_aleat(n):
    s = [0]*n
    for i in range(n):
        s[i] = randrange(0,100)
    return s
##########################################

#Program principal
a=int(input("Entrer le nombre de chiffre : "))
t=list_aleat(a)
print(t)
```

```
========== RESTART: C:\Users\PC\Desktop\exemple cours\aleatoire.py ==========
Entrer le nombre de chiffre : 20
[74, 42, 3, 92, 21, 39, 51, 77, 33, 1, 76, 82, 9, 57, 88, 76, 85, 90, 40, 80]
>>>
```

Exercice

- Ecrire une fonction qui permet de créer une matrice d'ordre n et la retourner
- Ecrire une procédure qui permet d'afficher les éléments d'une matrice

Solution

```
def remplissage_matrice(n):
    #remplir une matrice carrée
    t = [ ([0] * n) for i in range(n) ]

    for i in range(n):
        for j in range (n):
            print("t[",i,"][",j,"]=",end="")
            t[i][j]=int(input())
    return t
def affichage_matrice(t):
    print("les éléments de la matrice")
    for i in range(n):
        for j in range (n):

            print(t[i][j],end=" ")
        print('\n')

##################################################
n=int(input("Entrer le nombre de ligne : "))
mat=remplissage_matrice(n)
affichage_matrice(mat)
```

```
Entrer le nombre de ligne : 2
t[ 0 ][ 0 ]=6
t[ 0 ][ 1 ]=7
t[ 1 ][ 0 ]=8
t[ 1 ][ 1 ]=9
les éléments de la matrice
6 7

8 9
```