Informatique et Programmation

Cours 3

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/prog-py-22

Plan

- exercices du cours 2
- impressions formattées
- tableaux et listes (suite)
- itération sur les listes (suite)
- tableaux multi-dimensionnels, matrices
- exercices

dès maintenant: télécharger Python 3 en `http://www.python.org`

Exercices du cours 2

- indice du maximum dans une liste

```
def is_palindrome (s) :
    n = len(s)
    for i in range(n) :
        if s[i] != s[n-1-i] :
            return False
    return True
```

```
import sys
MAX_INT = sys.maxsize
MIN_INT = -sys.maxsize

def index_max_of (x) :
    n = len (x)
    m = MIN_INT; imax = -1
    for i in range (n):
        if x[i] > m :
            m = x[i]; imax = i
    return imax


print ("le maximum est en position %d\n" %(index_max_of (a)))
```

les sources sont en http://jeanjacqueslevy.net/prog-py-22/progs/c3a.py

Impressions formattées

- impression

```
def concat_print0 (s1, s2):
    print (s1 + " " + s2)

concat_print0 ("Hello", "World !")
Hello World !


def concat_print1 (s, n, x):
    print ("%s vaut %d ou %.2f" %(s, n, x))

concat_print1 ("Le resultat", 32, 28.5)
Le resultat vaut 32 ou 28.50


def concat_print2 (s, n, x):
    print ("{} vaut {} ou {}".format (s, n, x))

concat_print2 ("Le resultat", 32, 28.5)
Le resultat vaut 32 ou 28.5
```

- voir en `www.programiz.com/python-programming/input-output-import`

Intervalles

- intervalles (*range*)

```
>>> for i in range (0, 10):
...   print (i)
...
0
1
2
3
4
5
6
7
8
9
```

intervalle semi-ouvert

- abréviation et pas dans les intervalles

```
>>> range (10)
range(0, 10)
>>> for i in range (0, 10, 2) :
...   print (i)
...
0
2
4
6
8
```

2 est le pas

Exercice: quel est le sens de `range (10, 0, -1)` ?

Intervalles, n-uplets, strings

- tranche (*slice*)

```
>>> x
[1, 3, 4, 2, 3, 5, 9]
>>> x[3:6]
[2, 3, 5]
>>> x[3:6:2]
[2, 5]
>>> x[3:]
[2, 3, 5, 9]
>>> x[:6]
[1, 3, 4, 2, 3, 5]
>>> x[::2]
[1, 4, 3, 9]
```

intervalle semi-ouvert

- un n-uplet (*tuple)* est une liste **non modifiable**

```
>>> b = (9, "novembre", 1989)
>>> b[0]
9
>>> b[1]
'novembre'
>>> b[2]
1989
```

- une chaîne de caractère est une liste de caractères **non modifiable**

```
>>> s = "abcdefghijklmnopq"
>>> s[3:10]
‘defghij'
```

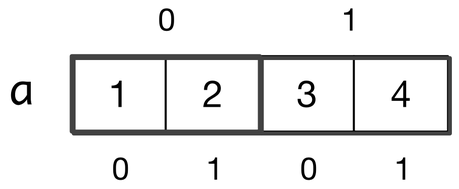
Tableaux multi-dimensionnels

- une matrice est une liste de listes

```
>>> a = [[1,2], [3,4]]

>>> a[0][0]
1
>>> a[0][1]
2
>>> a[1][0]
3
>>> a[1][1]
4
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$



- une itération sur les matrices

```
def print_matrix (a) :
    for line in a :
        for elt in line :
            print (elt, end = ' ')
        print ()
```

impression sur une ligne

```
def print_matrix (a) :
    for line in a :
        for elt in line :
            print ("%2d " %elt, end = ' ')
        print ()
```

Tableaux multi-dimensionnels

• addition de matrices

```
def add (a, b) :
    m = len (a); n = len (a[0])
    if m != len(b) or n != len(b[0]) :
        print("Erreur !"); return
    r = new_matrix (m, n)
    for i in range(m) :
        for j in range(n):
            r[i][j] = a[i][j] + b[i][j]
    return r
```

nouvelle matrice

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} = \begin{pmatrix} 11 & 22 \\ 33 & 44 \end{pmatrix}$$

Exercice Écrire la fonction qui multiplie tous les éléments d'une matrice par un scalaire

$$B = n \times A \quad \text{où} \quad b_{i,j} = n \times a_{i,j}$$

Exercice Programmer la multiplication de matrices

$$C = A \times B \quad \text{où} \quad c_{i,j} = \sum_{k=1}^{k=n} a_{i,k} \times b_{k,j}$$

Tableaux multi-dimensionnels

- création d'une matrice pleine de zéros

```
def new_matrix (m, n) :
    a = []; z = [0]*n
    for i in range(m): a.append (z.copy())
    return a
```

à comprendre plus tard !

- pour faire des calculs numériques, il vaut mieux utiliser numPy

http://numpy.org/

```
import numpy as np

a = np.zeros ( (3, 2) )
```

Tableaux multi-dimensionnels

- carré magique

```
def magique (n) :
    a = new_matrix (n, n)
    i = n - 1
    j = n // 2
    for k in range (n*n) :
        a[i][j] = k+1
        if (k+1) % n == 0 :
            i = (i - 1) % n
        else :
            i = (i + 1) % n
            j = (j + 1) % n
    return a
```

n impair

```
>>> print_matrix(magique(3))

4 9 2
3 5 7
8 1 6
```

somme 15 sur lignes et colonnes

somme 15 sur les 2 diagonales

```
>>> print_matrix2(magique(7))

 22  31  40  49   2  11  20
 21  23  32  41  43   3  12
 13  15  24  33  42  44   4
  5  14  16  25  34  36  45
 46   6   8  17  26  35  37
 38  47   7   9  18  27  29
 30  39  48   1  10  19  28
```

Recap

- mots clés en Python (déjà vus en rouge)

```
>>> help()
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

Conclusion

VU:

- impressions formattées
- intervalles, n-uplets, chaînes de caractères
- tableaux multi-dimensionnels

TODO list

- récursivité