Séquences : listes et tableaux NumPy

I. Séquences en Python

1) Les principaux types de séquences

Definition 1 :
– *Une **séquence** est une suite d'éléments, de types quelconques, stockés dans des cases mémoire ordonnées.*
– *Une séquence est dite **mutable ou muable** si on peut la modifier.*

En Python, il existe plusieurs types de séquences :
– Séquences muables : les listes (LIST), les tableaux numpy (ARRAY)...
– Séquences immuables : les chaînes de caractères (STRING), les n-uplets (TUPLE), les RANGE...

Exemple 1 : • [1, 2, 3, 4, 5] == range(1,6) ♯ ce test d'égalité renvoie False
• Si mot = 'PT star', alors les commandes : mot[0] = 'E' et mot.append('S') renvoient une erreur.

Proposition 1 *: l'ensemble des listes, des tuples et des chaînes est muni de la loi + de **concaténation**.*

II. Accès aux éléments d'une séquence

1) Accès à une tranche (slice)

Definition 2 : soit S une séquence de longueur n. Soit 3 entiers i, j et pas, avec $0 \leqslant i < j \leqslant n-1$.
`S[i:j: pas]` retourne la tranche `[ S[i],S[i+pas], S[i+2*pas], ...,S[j-pas]]`

2) Conversion de types de séquences

- Pour convertir une chaîne de caractères C en la liste de ses caractères, on peut utiliser : `list(C)`.
- Pour convertir une liste L en la chaîne de ses caractères, on peut utiliser la commande : `"".join(L)`.
- Pour convertir une liste L en tuple, on peut utilise la commande : `tuple(L)`.

3) Fonctions usuelles associées aux séquences : `min, max, del, sum...`

4) «Méthodes» usuelles associées aux ***listes*** : `append, count, sort, index, pop...`

III. Tableaux NumPy

La bibliothèque numpy permet de représenter des matrices de dimension arbitraire, de type *numpy.ndarray*[1].

1) Fonctions numpy pour l'étude des matrices en algèbre linéaire

Une matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ s'implémente en : `A=np.array([[1,2],[3,4]])`.
• A+B la somme matricielle MAIS A*B est la multiplication terme à terme, A**k est la puissance terme à terme.

• `np.dot(A,B)` est la multiplication matricielle habituelle, `np.linalg.matrix_power(A,7)` la puissance matricielle.

• On comprendra aisément les commandes d'algèbre linéaire :
`np.trace(A)` ; `np.transpose(A)` ; `np.linalg.det(A)` ; `np.linalg.inv(A)` ; `np.linalg.eig(A)`

1. NumPy est une bibliothèque qui contient une collection d'algorithmes mathématiques et de fonctions enrichissant encore la puissance du langage Python, en fournissant à l'utilisateur un environnement de traitement de données et système de prototypage rivalisant avec Matlab, IDL, Octave, R-Lab, et SciLab.

2) Différences notables entre les tableaux numpy et les listes

a) Les tableaux numpy doivent être constitués d'éléments du même type.

Attention : en cas de modification d'un élément, le remplaçant est automatiquement typé comme les autres !

b) La longueur des tableaux numpy est fixée à la création et non modifiable.

Attention : La méthode `append` ne fonctionne pas sur les tableaux numpy.

c) Index matriciel

L'appel d'un élément d'index (i,j) dans un tableau numpy T peut se faire par : `T[i][j]` ou `T[i,j]`

3) Avantages des tableaux numpy sur les listes

a) Nombreux tableaux pré-programmés

arange(min,max,pas) définit la sous-liste des scalaires incluse dans [min, max[, avec un pas donné [2].

linspace(min,max, n) définit une liste de n scalaires formant une discrétisation à pas constant de l'intervalle [min,max].

Matrices usuelles :

- ligne de n zéros : `np.zeros(n)`
- matrice nulle : `np.zeros((largeur,hauteur))`
- matrice I_n : `np.eye(taille)`
- matrices diagonales : `np.diag([liste des éléments diagonaux])` `np.diag([liste des éléments surdiagonaux],k=1)`

b) Application d'une fonction numpy aux éléments d'un tableau numpy

PROPOSITION 2 *: Une opération usuelle (+,*,÷) ou une fonction mathématique de numpy appliquée à des tableaux numpy s'applique à chaque élément des tableaux ("terme à terme").*

4) Application : tracé de courbe représentative de fonctions

On veut représenter graphiquement une fonction f sur l'intervalle $[a,b]$ (avec $a < b$). f a été préalablement définie en Python.

```
import numpy as np
import matplotlib.pyplot as pl          # Importation du module graphique
pl.close()                              # Ferme le graphique courant

def f(x) :
    return ...                          # Les fonctions doivent être dans numpy.

h = 0.01
xlist = np.arange(a,b+h,h)          # équivaut à : np.linspace(a,b,int((b-a)/h))
ylist = f(xlist)

""" On represente alors la courbe (xlist, ylist [, options]).
Les options : choix de la couleur ('r','b','g'... pour red, blue, green...) ,
du style de trait ('--', '.' ...), de l'etiquette...
"""

pl.plot ( xlist, ylist, "r .", label = "nom de courbe")
pl.legend()
pl.xlabel("abscisses") ; pl.ylabel("ordonnees") ; pl.title("Titre")
pl.show()
```

2. soit : `np.array([ min + k* pas for k in range(0, int((max-1-min)/pas )+1) ])`