TD n° 2 : Fonctions — Tableaux — Entrées-Sorties

Pour cette feuille de TD, il faut avoir consulté et travaillé la documentation en ligne du module `numpy` : `https://numpy.org/doc/stable/user/quickstart.html`. Le mieux est d'ouvrir un interpréteur `python` (en tapant python3 dans le terminal de commandes ou en utilisant celui de `spyder3`), et de tester et modifier les exemples donnés, attention cela prend plusieurs heures.

Exercice 1 – Manipulation de tableaux 1D

1. Créer et afficher un tableau de type `numpy.ndarray` monodimensionnel (affecté à la variable `a`) de 10 entiers nuls et dont la cinquième valeur vaut 1.
2. Créer et afficher un tableau de type `numpy.ndarray` monodimensionnel (affecté à la variable `b`) de 10 nombres complexes nuls et dont l'avant-dernière valeur vaut $1 + i$.
3. Créer et afficher un tableau de type `numpy.ndarray` monodimensionnel (affecté à la variable `c`) contenant les entiers de 10 à 20 (10 et 20 compris).
4. Afficher le nombre d'éléments de `c` strictement supérieurs à 14 :
   (a) à l'aide d'une boucle `while`.
   (b) à l'aide d'une fonctionnalité de `numpy`.
5. Renverser `c` : le premier élément devient le dernier, etc.
6. Créer et afficher un tableau de type `numpy.ndarray` monodimensionnel (affecté à la variable `e`) contenant 10 valeurs aléatoires comprises entre 0 et 1 (voir la documentation de `numpy.random.uniform`)
7. Créer et afficher un tableau de type `numpy.ndarray` monodimensionnel (affecté à la variable `f`) contenant 20 valeurs aléatoires comprises entre 5 et 12.
8. Afficher les valeurs minimales et maximales du tableau `f` précédent.
9. Afficher les indices du tableau `f` de ses valeurs minimales et maximales.
10. À l'aide d'une boucle `for`, calculer la valeur moyenne des éléments du tableau `f`.
11. Même question avec une fonction `numpy`.

Exercice 2 – Discrétisation d'un segment

On considère le segment $[a, b]$ que l'on souhaite discrétiser, c'est-à-dire créer un ensemble *fini* de N points $\{x_0, x_1, \ldots, x_{N-1}\}$ appartenant à ce segment. Bien souvent en calcul numérique, on choisit ces points équi-répartis sur $[a, b]$, c'est-à-dire équidistants les uns des autres, et tels que $x_0 = a$ et $x_{N-1} = b$.
Soit $a = 4$, $b = 12.5$, $N = 11$:

- à l'aide d'une boucle `for`, afficher les x_i désirés.
- même question avec une fonction `numpy`.

Exercice 3 – Manipulation de tableaux 2D (matrices)

1. Créer et afficher un tableau de type `numpy.ndarray` bidimensionnel (affecté à la variable `A`) représentant la matrice suivante :

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

2. Afficher la première ligne de `A`.
3. Afficher la dernière colonne de `A`.
4. Créer et afficher un tableau de type `numpy.ndarray` bidimensionnel (affecté à la variable `I`) représentant la matrice 3×3 identité I_3.
5. Effectuer et afficher le résultat des produits matriciels AI_3 et I_3A.
6. Transformer `A` en un tableau monodimensionnel via des boucles puis via une instruction `numpy`.

Exercice 4 – Matrices et vecteurs

Dans un programme `python`,

1. Créer et afficher les tableaux `a` et `b` de type `numpy.ndarray` permettant de représenter deux vecteurs $\vec{a}$ et $\vec{b}$ définis de la manière suivante,

$$\vec{a} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ N \end{pmatrix} \quad \text{et} \quad \vec{b} = \begin{pmatrix} N \\ N-1 \\ N-2 \\ \vdots \\ 1 \end{pmatrix}$$

   N sera préalablement fixé à 10.
2. Effectuer les opérations vectorielles : $\vec{c} = \vec{a} + \vec{b}$; $\vec{d} = \vec{a} - \vec{b}$ pour définir les tableaux `c` et `d`.
3. Afficher les résultats pour `c` et `d` à l'aide d'une fonction appelée `AfficheVecteur`. Respecter le prototype suivant : `def AfficheVecteur(nom, tab):`. Autrement dit, cette fonction possède 2 arguments, une chaine de caractère et un tableau. Cette fonction n'aura pas de valeur retour, mais affichera la chaine de caractère et le tableau `numpy` 1D donnés en argument. Par exemple, l'appel suivant :
   `AfficheVecteur('vecteur t', np.array([2.1,1.1,-3.9]))`
   produira :

```
vecteur t
2.1
1.1
-3.9
```

4. À l'aide de la fonction `numpy.zeros` et de boucles imbriquées, initialiser une matrice tridiagonale $N \times N$ de la façon suivante :

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

5. Même question à l'aide de la fonction `numpy.diag`.

6. Effectuer le produit matrice vecteur suivant : $\vec{e} = \mathbf{M}\vec{b}$ et afficher le résultat à l'aide de la fonction `AfficheVecteur`.

Exercice 5 – Dérivation numérique

Dans un fichier `g.data`, on a stocké, à raison d'une valeur par ligne du ficher, les valeurs d'une fonction g évaluée aux points $x_i = i\delta x$, avec $i = 0, 1, 2 \cdots, n$ où n est donné au début du fichier. Le pas d'espace vaut $\delta x = 0.01$ et est défini en début de programme.

1. Écrire un programme en `python` permettant de lire dans le fichier `g.data` les valeurs de $g(x_i)$ et de les stocker dans un tableau de type `numpy.ndarray`.
2. Écrire la fonction `deriveeP` qui calcule la dérivée première d'une fonction à l'aide de la relation :
$$g'(x_i) \approx \frac{g(x_{i+1}) - g(x_i)}{\delta x}$$
Le prototype de la fonction `deriveeP` est le suivant :
```
def deriveeP(tab):
```
la valeur retournée est un tableau de taille n contenant les valeurs de la dérivée première en x_i, avec $i = 0, 2 \cdots, n-1$ (noter bien que la dérivée ne peut être évaluée en x_n).
3. De la même manière, écrire la fonction `deriveeS` qui calcule la dérivée seconde d'une fonction à l'aide de la relation :
$$g''(x_i) \approx \frac{g(x_{i+1}) - 2g(x_i) + g(x_{i-1})}{\delta x^2}$$
Le prototype de la fonction `deriveeS` est le suivant :
```
def deriveeS(tab):
```
la valeur retournée est un tableau de taille $n-1$ contenant les valeurs de la dérivée seconde en x_i, avec $i = 1, 2 \cdots, n-1$ (en effet, noter bien que la dérivée seconde ne peut être évaluée en x_0 et x_n).
4. Écrire dans un fichier `derivees.data`, les valeurs de x_i, $g(x_i)$, $g'(x_i)$ et $g''(x_i)$ sous le format quatre colonnes pour $i = 1, 2, \cdots, n-1$.
5. Tracer les fonctions g, g' et g''.