Gammes de la programmation Python

Les exercices suivants sont des fonctions de bases que chacun doit savoir refaire sans difficulté en arrivant en deuxième année.

Pour chacune de ces fonctions, il importe de connaître sa complexité.

Les programmes signalés par un point d'exclamation sont ceux qui sont explicitement mentionnés dans le programme officiel, ce qui signifie qu'ils seraient considérés comme question de cours dans un sujet de concours.

Exercice 1. Tableaux Python

Nous appelons tableau Python les objets de type `list`. Nous appellerons matrice un tableau de tableaux.

1. Calculer la somme des éléments d'un tableau, puis d'une matrice.
2. Calculer le maximum et l'argmax (l'indice où le couple d'indices où le max est atteint) d'un tableau, puis d'une matrice.
3. Renvoyer un tableau contenant tous les éléments positifs d'un tableau de nombres donné.
4. (!) Tester la présence d'un élément d'un tableau, d'abord pour un tableau quelconque puis par dichotomie pour un tableau trié.

Exercice 2. Modification de tableaux

Ici, on demande des procédures, c'est-à-dire des programmes qui ne renvoient rien. Ces procédures modifieront le tableau passé en argument.

1. Écrire une procédure prenant en entrée un tableau de nombres `t` et un nombre `x` et multipliant chaque case de `t` par `x`.
2. Écrire une procédure `transpose` prenant en entrée un tableau `t` et deux indices i et j qui échange les cases i et j de `t`.
3. Écrire une procédure `renverse` qui prend un tableau `t` et le renverse (son premier élément est échangé avec le dernier, le deuxième avec l'avant-dernier, etc.)
4. Écrire une procédure `applique` prenant en entrée une fonction f et un tableau `t` et appliquant la fonction f à chaque case de `t`. Par exemple, si `carré` est la fonction $x \mapsto x^2$, et si `t` est le tableau `[1,2,3,4]`, alors après avoir exécuté `applique(carré, t)`, `t` doit contenir `[1,4,9,16]`.

Exercice 3. Tableaux numpy

On rappelle que la commande `np.zeros( (n,p), dtype=` *type*`)` permet de créer une matrice numpy de format $n \times p$ dont les coefficients ont le type *type*.

1. Prendre une matrice numpy `M` de flottants et renvoyer la matrice numpy de même format contenant les parties entières des coefficients de `M`.
2. (!) Programmer les fonctions élémentaires sur les lignes d'une matrice utiles pour le pivot de Gauss : échange, dilatation, et transvection.

Exercice 4. Chaînes de caractères

(!) Reconnaître si un mot (en pratique une petite chaîne) est inclus dans un texte (en pratique, une plus grosse chaîne).

Exercice 5. Écriture binaire

1. Écrire une fonction qui prend un tableau de 0 et de 1 représentant un entier positif écrit en base 2, et renvoyant cet entier.
2. Écrire la fonction réciproque prenant un entier positif et renvoyant son écriture en base 2.

Exercice 6. ! Calcul approché

On rappelle ici les différents algorithmes au programme du second semestre de première année. Pour chacun il faut connaître la complexité et la précision.

1. méthode de Newton : savoir écrire un programme prenant en entrée une fonction f, un flottant `eps`, un flottant `u0` et renvoyant si possible une valeur approchée d'une solution de l'équation $f(x) = 0$, d'inconnue x, avec une précision de l'ordre de `eps` en utilisant la méthode de Newton.
2. Même chose avec la méthode de dichotomie.
3. Méthode des rectangles et des trapèzes : savoir écrire une fonction prenant en entrée deux flottants a et b tels que $a < b$, une fonction f continue sur $[a, b]$, un pas h et renvoyant une valeur approchée de $\int_a^b$ par chacune des deux méthodes.
4. Méthode d'Euler : écrire une fonction prenant en entrée une fonction F, des flottants h, t_0, t_f et y_0, et calculant une valeur approchée de la fonction f telle que $\forall t \in [t_0, t_f]$, $f'(t) = F(f(t), t)$ et $f(t_0) = y_0$.
5. Pivot de Gauss : savoir écrire une procédure qui réduit une matrice par la méthode du pivot de Gauss.