Travaux pratiques n°4

Exercices d'algorithmique

F. Cuvellier, J. Tanoh

M.P.S.I. 1, 2016–2017

1 Algorithmes de calcul numérique

1.1 Calculs sur un tableau

Les tableaux de nombres jouent un rôle important en Python, comme dans de nombreux autres langages de programmation. Formellement, ils consistent en une suite finie d'objets, séparés par des virgules, délimités dans leur ensemble par des crochets. Les objets sont indexés par des nombres entiers naturels consécutifs, le premier étant 0. On accède l'objet d'indice `k` d'un tableau `L` par la commande `L[k]`. Pour d'autres fonctions (ou méthodes) applicables à un tableau, on pourra consulter la page d'aide (`help(list)`).
Le paquet NumPy fournit une autre structure de tableau, qui diffère sensiblement de la structure de tableau native de Python.

Exercice 1. On considère un tableau de nombres `L`. Donner un algorithme pour le calcul de la somme des éléments de `L`, puis écrire une fonction `sum` de paramètre `L`, qui retourne cette somme. (Dans le cas où le tableau est vide, la somme est nulle.)

Exercice 2. Donner une fonction qui retourne la moyenne arithmétique, la moyenne géométrique et la moyenne harmonique d'un tableau de nombres.

Exercice 3.

1. On considère un tableau d'au moins deux nombres `L` et une fonction numérique `f` de deux variables numériques. Donner un algorithme qui calcule le nombre obtenu en remplaçant les deux premiers nombres de `L` par la valeur qu'ils donnent à `f`, et en répétant l'opération sur le tableau qui en résulte jusqu'à ce qu'il n'ait plus qu'un élément. C'est cet élément que retourne l'algorithme.

2. Écrire la fonction `power(L, f)` correspondante.

3. On veut compléter la définition de cette fonction en adjoignant au tableau de ses paramètres un paramètre optionnel `reverse` booléen, de valeur par défaut `False`, de sorte qu'elle retourne le nombre obtenu comme précédemment si `reverse` est `False`, et, dans le cas contraire, le nombre obtenu avec le tableau lu de droite à gauche. Dans les deux cas, le premier argument de `f` est celui de gauche et le second celui de droite. Écrire la fonction `power(L, f, reverse)`.

4. Calculer $2^{3^{4^5}}$ et $((2^3)^4)^5$. (Au moins l'un de ces deux nombres entiers dépasse les capacités de Python.)

1.2 Valeurs approchées d'une intégrale

Exercice 4. Étant donnée une fonction numérique f continue dans un intervalle fermé $[a, b]$ de $\mathbf{R}$, pour estimer son intégrale $\int_a^b f(x)\,dx$, on peut se servir de l'expression

$$R_n(f) = \frac{b-a}{n} \sum_{k=0}^{n-1} f\big(a + \frac{k}{n}(b-a)\big),$$

où n est un nombre entier naturel non nul. On démontre que $R_n(f)$ tend vers $\int_a^b f(x)\,dx$ quand n croît indéfiniment.

1. Écrire une fonction `rect(f, n, a=0, b=1)` qui retourne $R_n(f)$. Les valeurs par défaut de a et de b sont 0 et 1.
2. Estimer la vitesse de convergence de $(R_n(f))$ vers $\int_a^b f(x)\,dx$ dans le cas où f est une fonction polynomiale de degré < 5. Dépend-elle du degré ? (La vitesse de convergence est ici donnée par le paramètre d'une famille convenable de suites convergentes vers 0.)

2 Nombres premiers

2.1 Recherche du plus petit diviseur premier

Exercice 5. Pour déterminer le plus petit diviseur premier d'un nombre entier naturel non nul n, on se fonde sur quelques remarques élémentaires :

- Le nombre 1 n'a pas de diviseur premier.
- Ou n est premier, ou son plus petit diviseur premier est plus petit que $\sqrt{n}$.
- Tout nombre premier autre que 2 et 3 précède ou suit un multiple de 6.

On peut donc tester si le nombre entier naturel n est divisible par 2, par 3 ou par le prédécesseur ou par le successeur d'un multiple de 6.

1. Proposer à partir de ces remarques un algorithme qui donne le plus petit diviseur premier de n.
2. Écrire la fonction `leastprime(n)` correspondante.

Exercice 6. Donner une fonction qui retourne le tableau des diviseurs premiers d'un nombre entier naturel non nul. (Elle retournera le tableau vide pour le nombre 1.)

2.2 Crible d'Ératosthène

Le *crible d'Ératosthène* est une méthode de détermination pratique des nombres premiers plus petits qu'un nombre entier naturel n donné.

- On part du tableau des nombres entiers de 2 à n, rangés dans l'ordre croissant.
- Soit p le plus petit de ces nombres entiers (au début, $p = 2$). On le met de côté et on retire du tableau tous les multiples de p.

- On obtient un nouveau tableau de nombres entiers naturels, rangés dans l'ordre croissant, et on retourne à l'étape précédente aussi longtemps que le tableau qui en résulte est non vide.
- Les nombres premiers $\leqslant n$ sont les nombres p mis de côté.

Exercice 7. Utiliser le crible d'Ératosthène pour écrire une procédure à un paramètre, le nombre entier naturel n, qui retourne le tableau des nombres premiers $\leqslant n$ rangés dans l'ordre croissant. En fonction de n, estimer les nombres des opérations nécessaires pour dresser ce tableau (additions, multiplications, divisions, tests).

3 Manipulation de chaînes de caractères

3.1 Mise en page

Exercice 8. À partir d'un nombre strictement positif N et d'une chaîne de caractères T, qui représente un texte constitué de mots et de signes de ponctuation, on voudrait former la chaîne de caractères qui représente le même texte, mais où chaque ligne possède au plus N caractères. Si deux mots du le texte d'origine sont séparés par une ou plusieurs espaces, ou ils se retrouvent dans une même ligne du texte final et une seule espac devra les séparer, ou ils se trouvent dans deux lignes consécutives et les espaces qui les séparaient disparaissent dans le caractère de fin de ligne.

1. Donner un algorithme qui effectue cette transformation.
2. L'implémenter en Python. (Par exemple, donner une fonction `num(N, T)` qui, à partir d'un nombre entier naturel non nul `N` et d'une chaîne de caractères `T`, renvoie la chaîne de caractères dont les lignes ont au plus `N` caractères. Bien sûr, on choisira `N` de sorte qu'aucun mot n'ait moins de `N` caractères.)

3.2 Exercice de numérotation

Exercice 9. On dispose d'une chaîne de caractères dont les mots sont des nombres entiers naturels non nuls, distincts, qui chacun apparaissent au début d'une ligne. Par exemple, on part de la chaîne de caractères `"""\n1\n3\n5\n2\n7\n4\n"""` (le caractère `\n` désigne celui de nouvelle ligne), ou

```
"""
1
3
5
2
7
4
"""
```

On voudrait former une nouvelle chaîne de caractères, dont les mots sont les mêmes nombres, mais que l'on place de sorte que chacun d'eux se trouve au début de la ligne dont il serait le numéro. Avec les nombres de l'exemple précédent, on obtiendrait

```
"""
1
2
3
4
5

7
"""
```

1. Donner un algorithme qui réalise cela dans le cas où les nombres apparaissent dans l'ordre croissant.
2. L'implémenter en Python. Le paramètre est une chaîne de caractères délimitée par des triples apostrophes ou des triples guillemets.
3. Donner un algorithme qui traite le cas général.
4. L'implémenter en Python.