Structures de données simples en Python

juillet 2020

1 Listes par compréhension

Exercice 1

Définir les listes suivantes ***par compréhension*** *:*

1. *les carrés de tous les entiers compris entre 1 et 100 (inclus) ;*
2. *les couples (x, y) d'entiers positifs tels que $x + y = 100$;*
3. *les couples d'entiers de la forme (x, y) tels que $x \leq y$ et les deux entiers x et y sont tous deux inférieurs à 10 (indication : il est plus facile de les ranger par y croissants) ;*
4. *les mêmes qu'à la question précédente, mais rangés par x croissants ;*
5. *les entiers compris entre 1 et 1000 qui ne sont multiples ni de 3 ni de 5.*

Exercice 2

Objectif pour tout cet exercice : la réponse à chaque question tient en 1 ligne. On fera bon usage de la fonction `sum`.

1. *Calculer* $\sum_{1 \leq i \leq j \leq 100} i^j$.
2. *On représente une matrice comme la liste de ses lignes.*
   Écrire une fonction `sommes_lignes(M)` *qui renvoie* $[S_1, \dots, S_n]$ *où* S_i *est la somme de la i-ème ligne de* M.
3. *Écrire une fonction* `Id(n)` *qui renvoie la matrice identité* $I_n = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$.

2 Mémoïzation à l'aide de dictionnaires en Python

2.1 Échauffement

Exercice 3 (Échauffement sur les dictionnaires)

Programmer les fonctions suivantes (qui existent en fait nativement en Python) :

1. `get(dico, key, default)`
   renvoie la valeur associée à `key` *si celle-ci est présente dans* `dico`*, sinon* `default`.
2. `items(dico)`
   renvoie la liste des couples `(cle, valeur)` *du dictionnaire.*
3. `setdefault(dico, key, default)`
   Si `key` *est présente dans* `dico`*, on renvoie la valeur associée ; sinon, on ajoute cette clé avec pour valeur* `default`.

2.2 Mémoïzation simple

La *mémoïzation* consiste à stocker en mémoire une valeur pour ne pas avoir à la recalculer si elle s'avérait utile à nouveau par la suite. Nous allons en voir quelques applications.

Exercice 4 (Syracuse)

Pour tout entier $p \in \mathbb{N}^*$, on définit la suite de Syracuse associée par

$$u_0 = p \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = \begin{cases} u_n / 2 & \text{si } u_n \text{ pair} \\ 3u_n + 1 & \text{sinon} \end{cases}$$

On admet que pour toute valeur raisonnable de p[1]*, la suite finit par boucler sur* $1, 4, 2$. *On note vol*(p) *le plus petit n tel que* $u_n = 1$, *lorsque* $u_0 = p$.

1. *Quelle est la plus grande valeur de vol*(p) *pour* $1 \leq p \leq 1000$ *? Et jusqu'à* $p = 10^7$ *?*
   Indice dont vous vous doutiez : *une attaque par force brute de ce problème est vouée à l'échec...*
2. *Combien d'entiers* $p \leq 10^6$ *vérifient vol*$(p) \leq 30$ *?*
3. Optionnel : *combien d'entiers p (sans limite de valeur !) vérifient vol*$(p) \leq 50$ *?*

2.3 Plus longue sous suite commune

Le problème On définit une sous-suite de A comme étant un mot de la forme $a_{i_1} a_{i_2} \dots a_{i_k}$ avec $i_1 < i_2 < \dots < i_k$, autrement dit une suite de lettres apparaissant dans le même ordre dans A mais non nécessairement consécutives.

On considère deux mots (sur un alphabet quelconque)

$$A = a_1 \dots a_n \quad \text{et} \quad B = b_1 \dots b_m$$

On recherche alors les sous-suites communes à A et B, et plus particulièrement la (ou une des) plus longue. Exemple : $A = aabaababaa$, $B = ababaaabb$ alors leur plus longue sous-suite commune (PLSSC) est *ababaaa*.

Résolution On généralise le problème à tous les préfixes de A et de B, et on note $p(i, j)$ la longueur de la plus longue sous-suite commune à $a_1 \dots a_i$ et $b_1 \dots b_j$.

On démontre assez facilement que

$$p(i,j) = \begin{cases} 1 + p(i-1, j-1) & \text{si } a_i = b_j \\ \max\left(p(i, j-1),\ p(i-1, j)\right) & \text{sinon} \end{cases}$$

Il est donc possible de résoudre le problème par une approche récursive, mais avec une écriture naïve de l'algorithme, de nombreuses valeurs $p(i, j)$ risquent d'être calculées plusieurs fois.

Exercice 5 (Plus longue sous-suite commune) *Adapter la technique de mémoïzation pour calculer* $p(i, j)$ ***efficacement*** *tout en conservant une écriture récursive.*

Exercice 6 (Cerise sur le gâteau) *Comment adapter l'algorithme précédent pour calculer non seulement la longueur de la plus longue sous-suite commune mais aussi en exhiber une ?*

1. Aux dernières nouvelles, cela a été vérifié jusqu'à $p = 5 \times 2^{60} \simeq 5.8 \times 10^{18}$.