Listes (1) — TP8

Exercice 1 — Le cours, en inverse

On rappelle qu'on a vu en cours 2 méthodes pour créer des listes : l'une utilisant une boucle `for`, l'autre dite "en compréhension".

Q1 Écrire une fonction `f` prenant en argument un entier $n \in \mathbb{N}^*$ et renvoyant la liste dont les éléments sont $1,\ \frac{1}{2},\ \frac{1}{3},\ \ldots,\frac{1}{n}$. Testez votre fonction.

Q2 Écrire une fonction `g` faisant la même chose que la fonction `f` mais selon l'autre méthode vue en cours. Testez votre fonction.

Q3 Écrire une fonction `inverse` prenant en argument une liste de nombres `L` et renvoyant la liste dont les éléments sont les inverses des éléments de `L`. Testez votre fonction.

Q4 Écrire une fonction `inverse_bis` faisant la même chose que la fonction `inverse` mais selon l'autre méthode vue en cours. Testez votre fonction.

Exercice 2 — Un peu de statistiques

On rappelle que si $(x_1, x_2, \ldots, x_n)$ est une série de nombres réels, alors on définit sa moyenne (arithmétique) $\overline{x}$ et son écart type σ par les formules :

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{et} \qquad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Q1 Écrire une fonction `moy` prenant en argument une liste de nombres réels et renvoyant sa moyenne. On pourra utiliser deux fonctions classiques vues en cours.

Q2 Écrire une fonction `ecart` prenant en argument une liste de nombres réels et renvoyant l'écart type de cette série de nombres.

Q3 Créer la liste `L = [1, √3, √5, √7, ..., √2023]` et vérifier que `moy(L)` vaut environ 29,992 et `ecart(L)` vaut environ 10,603.

Exercice 3 — Encore des graphes

On considère la suite (u_n) définie par $u_1 = 5$ et : $\forall n \in \mathbb{N}^*,\ u_{n+1} = \dfrac{1+u_n^3}{1+u_n^2}$. On souhaite tracer le graphe de la suite (u_n) c'est-à-dire placer les points de coordonnées (k, u_k) sur un schéma.

Q1 Écrire une fonction `suite` prenant en argument $n \in \mathbb{N}^*$ et renvoyant la liste `[`u_1`,`u_2`,...,`u_n`]`. On commencera par initialiser une liste `L` à la liste vide puis on calculera successivement les termes de la suite (u_n) et on les ajoutera dans `L`. On vérifiera que les premiers termes de la suite sont : 5 ; 4.846 ; 4.689 ; 4.528.

Q2 Que doit-on choisir comme liste d'abscisses `absi` pour pouvoir tracer le graphe de (u_n) grâce à la commande `plt.plot(absi, [`u_1`,`u_2`,...,`u_n`])` ? Tracez alors le graphe de (u_n) jusqu'à $n = 50$. On ne reliera pas les points du graphe entre eux : pour cela, utiliser la syntaxe `plt.plot(absi,ordo,'o')`

On considère maintenant la suite (v_n) définie par : $\forall n \in \mathbb{N}^*,\ v_n = \displaystyle\sum_{k=1}^{n} \frac{1}{k^2}$.

Q3 Tracer le graphe de la suite (v_n) jusqu'à la valeur de votre choix. On vérifiera que $v_1 = 1,\ v_2 = 1,25,\ v_3 \simeq 1,36,\ v_4 \simeq 1,42$. Quelle conjecture pouvez-vous faire quant au comportement de v_n lorsque $n \to +\infty$?

Exercice 4 Selection artificielle

Q1 Écrire une fonction `tronque` prenant en argument une liste de nombres `L` et un nombre réel `m` et qui renvoie la liste des éléments de `L` plus petits que `m`. Par exemple, `tronque([4,1,7,10,5,6,7],6)` doit renvoyer `[4,1,5,6]`.

Q2 Écrire une fonction `select` prenant en argument une liste de nombres entiers `L` et renvoyant la liste formée des éléments de `L` qui sont divisibles par 3. Par exemple `select([1,6,5,5,3,12])` doit renvoyer `[6,3,12]`.

Exercice 5 Clé de sécurité

On souhaite transmettre un message codé sous forme d'une liste de nombres entiers. Mais le canal de transmission utilisé n'est pas parfait. Il est donc possible que le message reçu ne soit pas exactement identique au message envoyé. Pour permettre au destinataire du message de vérifier s'il y a eu une erreur de transmisison, on propose d'ajouter en fin de message un dernier nombre appelé clé de sécurité et noté s. Cette clé s'obtient en sommant tous les nombres du message, puis en prenant les deux derniers chiffres du nombre obtenu.

Par exemple, si le message à envoyer est `L = [41,2,100,17,64]` alors on calcule $41+2+100+17+64 = 224$, la clé de sécurité sera donc $s = 24$, et le message finalement envoyé sera `M = [41,2,100,17,64,24]`.
Supposons alors qu'une erreur de transmission se produise et que le destinataire reçoive finalement le message `M' = [45,2,100,17,64,24]`. En calculant $45+2+100+17+64 = 228$, le destinataire saura qu'il y a eu une erreur de transmission puisque la clé de sécurité dans le message reçu n'est pas $s = 28$, il pourra alors par exemple demander de renvoyer le message. [1]

Q1 Écrire une fonction `cle` prenant en argument une liste de nombres `L` correspondant au message à envoyer et renvoyant la clé de sécurité s correspondante.

Q2 En déduire une fonction `securise` prenant en argument le message `L` et renvoyant le message à envoyer `M` avec la clé de sécurité. *On prendra soin de ne pas modifier la liste `L`, on vérifiera en particulier la valeur de `L` après avoir appelé `securise(L)`.*

On se place désormais du côté du destinataire qui a reçu le message `M` et souhaite savoir s'il est correct ou non. Pour cela, il lui faut extraire le dernier élément de la liste `M` grâce à la commande `pop` : la syntaxe suivante

```
1 x = M.pop()
```

permet de retirer le dernier élément de `M` pour le stocker dans la variable `x`. Cela modifie donc la liste `M`.

Q3 Dans la console, définissez la liste `A = [1,2,3]`. Enlevez ensuite le dernier élément de `A` et placez-le dans une variable `a`. Vérfiez les valeurs de `A` et de `a`. Rajouter ensuite l'élément `a` à `A` et vérifiez à nouveau les valeurs de `A` et de `a`.

Q4 Écrire une fonction `verifie` prenant en argument le message `M` reçu, et qui indique si la clé de sécurité du message `M` est bien celle attendue à la lecture du reste du message. Votre fonction renverra `True` si c'est le cas et `False` sinon.

1. Bien sûr, il peut arriver que la clé de sécurité soit correcte même s'il y a eu erreur de transmission, ou à l'inverse que l'erreur de transmission n'ait porté que sur la clé de sécurité elle-même la rendant incorrecte alors que le reste du message est correct... Mais il n'y a pas de risque zéro, et cet exemple n'est qu'un cas très simple de ce qu'on appelle les *codes correcteurs*.