Info

Python : révisions

Construction de vecteurs

Exercice 1 (Suites explicites)
Proposer une fonction Python qui prend en entrée un entier n et :

1. Renvoie les n premiers termes de la suite définie par : $\forall k \in \mathbb{N},\ u_k = \dfrac{k}{1+k^2}$
2. Renvoie les n premiers termes de la suite définie par : $\forall k \in \mathbb{N}^*,\ v_k = \dfrac{\ln(k)}{k}$

Exercice 2 (Suite à récurrence double)
Proposer une fonction qui prend en entrée un entier n et renvoie les n premiers termes de la suite définie par : $v_0 = 0$, $v_1 = 1$ et $\forall k \in \mathbb{N},\ v_{k+2} = \dfrac{k+1}{v_k + v_{k+1}}$

Exercice 3 (Paradoxe des anniversaires)

1. Dans une pièce, n personnes sont réunies. Montrer que la probabilité qu'au moins deux personnes aient la même date d'anniversaire est donnée par : $p_n = 1 - \dfrac{365!}{(365-n)! \times 365^n}$.
2. Vérifier que pour tout $n \in [\![1, 364]\!]$, $p_{n+1} = 1 - (1-p_n)\left(1 - \dfrac{n}{365}\right)$.
3. Proposer un script Python qui construit et affiche le vecteur `P = [` $p_1, p_2, \dots, p_{365}$ `]` .
   Remarque : On a par exemple $p_{50} = 97,04\%$

Sommes et produits

Exercice 4 (Somme basique)

1. Proposer au moins deux façons différentes de définir une fonction `somme(n)` qui calcule $S_n = \displaystyle\sum_{k=1}^{n} \frac{(-1)^k}{k}$
2. Proposer une fonction `vecsomme(n)` qui construit et renvoie le vecteur $V =$ `[` $S_1, S_2, \dots, S_n$ `]`
   sans utiliser la fonction `somme` précédente.

Exercice 5 (Coefficients binomiaux)

1. (a) Lorsque $p \leqslant n$, rappeler la définition de $\binom{n}{p}$ avec des factorielles.
   (b) En déduire une fonction Python `binome(p,n)` qui calcule $\binom{n}{p}$.
2. (a) Lorsque $p \leqslant n$, écrire $\binom{n}{p}$ comme un produit de seulement p termes.
   (b) En déduire une fonction Python `binome(p,n)` qui calcule $\binom{n}{p}$.

Exercice 6 (Sommes exploitant une relation de récurrence)

1. Proposer une fonction `somme(x,n)` qui, à l'aide d'une boucle `for`, calcule la somme $\displaystyle\sum_{k=0}^{n} \frac{x^k}{k!}$
   sans utiliser l'instruction `**` ni les factorielles.
2. Même question avec la somme $\displaystyle\sum_{k=0}^{n} \frac{(-1)^k}{(2k)!} x^{2k}$.

Construction de matrices

Exercice 7 (Matrice en forme de N)

1. Définir en Python la matrice $N = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$.
2. Plus généralement, proposer une fonction Python qui prend en entrée un entier n et renvoie la matrice carrée de taille $n \times n$ avec des 1 sur la diagonale, sur la première et dernière colonne, et des 0 partout ailleurs.

Exercice 8 (Matrice en forme de Z)

Même question avec une matrice "en forme de Z".

Résolution approchée d'équations

Exercice 9 (Approximation de π)

A l'aide de la méthode de dichotomie, proposer un script Python qui calcule et affiche une approximation de π à 10^{-5} près. On exploitera le fait que π est l'unique solution de l'équation $\cos(\frac{x}{2}) = 0$ sur le segment $[0, 4]$.

Exercice 10 (Approximation de $\sqrt{2}$)

Dans tout cet exercice, on suppose qu'on ne dispose pas de l'instruction `sqrt`...

Première façon : méthode de Dichotomie

0. Proposer un script Python qui calcule et affiche une approximation à 10^{-5} près de la valeur de $\sqrt{2}$.

(On rappelle qu'il s'agit bien-sûr de l'unique solution positive de l'équation $x^2 = 2$)

Deuxième façon : méthode du point fixe

On définit la fonction $g : x \mapsto \frac{2}{3}\left(x + \frac{1}{x}\right)$ ainsi qu'une suite u : $\quad u_0 > \sqrt{2}, \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = g(u_n)$.

1. (a) Déterminer l'unique point fixe de g sur $\mathbb{R}_+^*$

   (b) Montrer que $\forall n \in \mathbb{N},\ u_n > \sqrt{2}$.

   (c) Montrer que la suite u est décroissante et en déduire qu'elle converge vers $\sqrt{2}$.

2. (a) En exploitant une identité remarquable, justifier que pour tout $n \in \mathbb{N}, \quad 0 < u_n - \sqrt{2} \leqslant \frac{1}{2}(u_n^2 - 2)$.

   (b) Proposer un script Python qui, en exploitant la suite u, calcule et affiche une approximation de $\sqrt{2}$ à 10^{-5} près. On n'utilisera évidemment pas l'instruction `sqrt` et on mettra en jeu une boucle `while`.

3. (a) Déterminer un réel $M \in]0, 1[$ tel que $\forall x > 1,\ |g'(x)| \leqslant M$.

   (b) En déduire que pour tout $n \in \mathbb{N}, \quad u_{n+1} - \sqrt{2} \leqslant M(u_n - \sqrt{2}) \quad$ puis que $\quad u_n - \sqrt{2} \leqslant M^n(u_0 - \sqrt{2})$.

   (c) On propose de poser $u_0 = 2$, de sorte que $u_n - \sqrt{2} \leqslant M^n$.

   On donne la valeur : $\dfrac{5}{\ln(3) - \ln(2)} \simeq 12,3$.

   Pour quelle valeur de n au minimum est-on certain que u_n est une approximation de $\sqrt{2}$ à 10^{-5} près ?
   Proposer ainsi un nouveau script Python qui calcule et affiche une approximation de $\sqrt{2}$ à 10^{-5} près, en utilisant cette fois une boucle `for`.