Sommes, produits, suites numériques

Exercice 1 - Quelques séries classiques

Soit $(u_n) \in \mathbb{R}^{\mathbb{N}}$. On appelle **série de terme général** u_n la suite des sommes partielles (S_n) définie par :

$$\forall\, n \in \mathbb{N},\ S_n = \sum_{k=0}^{n} u_k$$

1. On pose $u_n = \dfrac{1}{n(n-1)}, n \geq 2$. Dans le langage Python, construire la fonction S_1 qui pour tout entier n donné, affiche les valeurs de S_k pour k allant de 2 à $n-1$ puis renvoie la valeur de S_n.
2. On pose $u_n = \dfrac{(-1)^n}{n}, n \geq 1$. Dans le langage Python, construire la fonction S_2 qui pour tout entier n donné, affiche les valeurs de S_k pour k allant de 1 à $n-1$ puis renvoie la valeur de S_n.
3. On pose $u_n = \dfrac{1}{\sum_{\ell=1}^{n} \ell^2}, n \geq 1$. Dans le langage Python, construire la fonction S_3 qui pour tout entier n donné, affiche les valeurs de S_k pour k allant de 1 à $n-1$ puis renvoie la valeur de S_n.

Exercice 2 - Recherche d'un seuil pour une précision donnée

On définit la suite (S_n) par :

$$\forall\, n \in \mathbb{N}^*,\ S_n = \sum_{k=1}^{n} \frac{(-1)^{k-1}}{k^2}$$

1. Dans le langage Python, construire la fonction S qui pour tout entier n non nul, construit la liste des valeurs S_k pour k allant de 1 à n, puis renvoie la liste obtenue **et** la valeur de S_n.
2. Dans l'année, on montrera que la suite (S_n) est convergente de limite $\ell = \dfrac{\pi^2}{12}$.
   Construire alors le programme *indice* qui pour tout $\epsilon > 0$, calcule les valeurs de S_n tant que $|S_n - \ell| > \epsilon$, puis renvoie l'indice n à partir duquel :
   $$|S_n - \ell| \leq \epsilon$$

Exercice 3 - Formule de Stirling

On définit la suite (u_n) par :

$$\forall\, n \in \mathbb{N},\ u_n = 2n\left(\frac{n}{e}\right)^{2n}$$

avec $e = \exp(1)$.

1. Dans le langage Python, construire la suite u qui pour tout entier n donné, renvoie la valeur de u_n.

On souhaite alors comparer le comportement asymptotique de (u_n) avec la suite $((n!)^2)$. Pour cela, on pose :

$$\forall\, n \in \mathbb{N}^*,\ v_n = \frac{(n!)^2}{u_n}$$

2. Dans le langage Python, construire la fonction $facto$ qui pour tout entier n donné, renvoie la valeur de $n!$. On pensera à proposer un test conditionnel sur n afin de renvoyer un message d'erreur si $n \notin \mathbb{N}$.
3. Construire la fonction *representation* qui pour tout entier n donné, construit, dans un plan muni d'un repère, la suite de points (k, v_k) pour k allant de 1 à n.
4. On appelle la fonction précédente pour $n = 50$. Que remarquez-vous ?
5. En déduire une suite (w_n) équivalente à $n!$, c'est à dire une suite (w_n) telle que $\dfrac{n!}{w_n} \underset{n\to+\infty}{\longrightarrow} 1$.

Ce dernier équivalent nous donne un résultat bien connu : la ***formule de Stirling****.*

Exercice 4 - Etude d'un système dynamique discret en fonction des conditions initiales

On considère la suite (u_n) définie par :

$$\begin{cases} u_0 = a,\ a \in \mathbb{R}_+ \\ \forall\, n \in \mathbb{N},\ u_{n+1} = \dfrac{1}{6}(u_n^2 + 8) \end{cases}$$

1. Dans le langage Python, construire le programme $pointfixe$ qui ne prend pas d'argument, mais qui représente dans un même graphe, les courbes d'équation $y = x$ et $y = \frac{1}{6}(x^2 + 8)$.
2. Construire alors la suite u qui, pour tout couple (n, a) donné, renvoie la liste des u_k pour k allant de 1 à n.
3. Tester votre programme avec $a = 1$, $a = 2$, $a = 3$, $a = 4$ et $a = 5$. Que pouvez-vous constater ?

Exercice 5 - Vers le nombre d'or

On définit la **suite de Fibonacci** par :

$$\begin{cases} u_0 = u_1 = 1 \\ \forall\, n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n \end{cases}$$

1. Dans le langage Python, construire la fonction *fibonacci* qui pour tout entier n donné, renvoie la valeur de u_n.
2. En déduire le programme *limite* qui pour tout entier n donné, affiche simplement la valeur des fractions $\dfrac{u_{k+1}}{u_k}$ pour k allant de 0 à n.

Exercice 6 - Suite de Syracuse

On définit la **suite de Syracuse** associée à l'entier k_0 par :

$$u_0 = k_0 \text{ et } \forall\, n \in \mathbb{N},\ u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ \frac{3u_n+1}{2} & \text{sinon} \end{cases}$$

1. Ecrire une procédure itérative *syracuse* qui pour un couple (k_0, n) donné, renvoie les valeurs $u_0, \dots, u_n$.
2. Calculer `syracuse(15,50)` et `syracuse(127,50)`. Que remarquez-vous ?
3. On appelle alors temps de vol le plus petit indice p tel que $u_p = 1$.
   (a) Ecrire une procédure itérative *indice* qui pour un entier k_0 donné, calcule les valeurs u_n tant que $u_n \neq 1$, puis renvoie le temps de vol obtenu.
   (b) Déterminer le temps de vol pour $k_0 = 15$ et $k_0 = 222$.

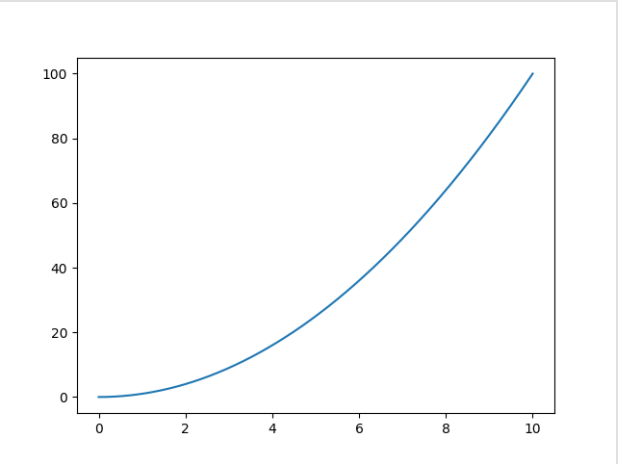
Exercice 7 - Calcul intégral

Soit f une fonction continue sur $[a, b]$, alors on rappelle que :

$$\int_a^b f(t)\, dt = F(b) - F(a)$$

avec F une primitive de f sur $[a, b]$.

1. Soient $a, b \in \mathbb{R}_+^*$. Donner une primitive de $t \longmapsto t^n$ sur $[a, b]$.
2. Dans le langage Python, construire alors le programme *integral* qui pour tout triplet (a, b, n) donné, renvoie la valeur de $\int_a^b t^n\, dt$.
3. On importe les fonctions intégrées au sous-module **scipy.integrate**.
   Que renvoie l'instruction `quad(lambda t:t**2,1,2)` ?
4. En déduire la fonction *approx* qui pour tout triplet (a, b, n) donné, renvoie une valeur approchée de $\int_a^b t^n\, dt$.
5. On fixe $n = 2$. En utilisant vos souvenirs de terminale, construire de la même façon un programme vous permettant de calculer une autre approximation de $\int_a^b t^2\, dt$?



C'est d'ailleurs sur ce genre de méthode numérique que s'appuie la fonction `quad`*...*