Université de Pau et des Pays de l'Adour
Collège STEE
L1-Mathématiques - Algorithmique Mathématiques I et Python

TP-03: Suites

Exercice 1. Suite géométrique

On considère une suite géométrique $(u_n)_n$ définie par $u_0 = 1$ et

$$\forall n, \qquad u_{n+1} = q u_n$$

où q la raison, donnée par $q = 0.987$.

1. Écrire une fonction `geometrique(q,u0,N)` retournant le $N^{\text{ième}}$ terme de la suite géométrique. L'entier naturel N sera lu au clavier.
2. Tester votre fonction en l'utilisant en particulier pour les valeurs de N suivantes: $N = 0$, $N = 1$, $N = 2$ puis $N = 10$.
3. Compléter le programme pour vérifier la formule générale pour les suites géométriques: $u_n = u_0 q^n$

Exercice 2. Suite arithmétique

On considère une suite arithmétique $(u_n)_n$ définie par $u_0 = 1$ et

$$\forall n, \qquad u_{n+1} = u_n + q$$

où q est la raison, donnée ici par $q = 5.0$.

1. Écrire une fonction `arithmetique(q,u0,N)` retournant le $N^{\text{ième}}$ terme de la suite arithmétique. L'entier naturel N sera lu au clavier.
2. Tester votre fonction en l'utilisant en particulier pour les valeurs de N suivantes: $N = 0$, $N = 1$, $N = 2$ puis $N = 10$.
3. Compléter le programme pour vérifier la formule générale pour les suites arithmétiques: $u_n = u_0 + n_t \times q$

Exercice 3. Suite de Collatz

On considère la suite de Syracuse $(u_n)_n$ d'un nombre entier $N > 0$ définie par récurrence, par $u_0 = N$ et :

$$\forall n, \qquad u_{n+1} := \begin{cases} u_n/2, & \text{si } u_n \text{ est pair} \\ 3u_n + 1, & \text{sinon.} \end{cases}$$

La conjecture de Syracuse, encore appelée conjecture de Collatz, conjecture d'Ulam, conjecture tchèque ou problème $3x + 1$, est l'hypothèse mathématique selon laquelle la suite de Syracuse de n'importe quel entier N strictement positif atteint 1. Par exemple, en commençant à 13, la séquence suivante est générée :

$$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

Cette séquence (commençant à 13 et finissant à 1) contient 10 termes.

1. Écrire une fonction `collatz(N)` retournant les termes successifs de la suite de Collatz. L'entier naturel N sera lu au clavier.

2. Tester la fonction pour $N = 10$, $N = 20$ et $N = 40$, puis calculer au bout de combien d'itérations nous arrivons à 1.

Exercice 4. Suite de Fibonacci et nombre d'or

La suite de Fibonacci est une suite récurrente définie par :

$$F_0 = 0, \quad F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n.$$

1. Créer une fonction `fibonacci(n)` qui renvoie le $(n+1)$ème terme de la suite de Fibonacci pour un entier n donné. Tester la fonction pour $n = 10$. Puis écrire un algorithme qui affiche les 10 premiers termes de la suite de Fibonacci.

2. Étudier le comportement du rapport entre deux termes successifs de la suite de Fibonacci.

   Indication:
   $$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \frac{1+\sqrt{5}}{2},$$
   où $\dfrac{1+\sqrt{5}}{2}$ est le nombre d'or.

Exercice 5. Suite logistique

On considère une suite logistique $(u_n)_n$ définie par récurrence par

$$\forall n, \qquad u_{n+1} = \mu u_n \left(1 - u_n\right)$$

où $u_0 \in [0; 1]$ et μ est un paramètre (dans $[0; 4]$ pour assurer que les valeurs de u_n restent dans $[0; 1]$). μ engendre soit une suite convergente, soit une suite soumise à oscillations, soit une suite chaotique.

Souvent citée comme exemple de la complexité de comportement pouvant surgir d'une relation non linéaire simple, cette suite fut popularisée par le biologiste Robert May en 1976. Une application de la suite logistique est la modélisation de la taille d'une population biologique au fil des générations.

1. Écrire une fonction `logistique(mu,u0,N)` retournant le $N^{\text{ième}}$ terme de la suite logistique. L'entier naturel N sera lu au clavier.

2. En prenant $u_0 = 0.75$, tester votre fonction en l'utilisant en particulier pour les valeurs de N suivantes: $N = 0$, $N = 1$, $N = 2$ puis $N = 10$.

3. Afficher les 20 premières valeurs de la suite associée à $u_0 = 0.75$ et $\mu = 2.5$. Comparer vos résultats à la limite théorique.

4. Afficher les 20 premières valeurs de la suite associée à $u_0 = 0.75$ et $\mu = 3.0$. Comparer vos résultats à la limite théorique.

5. Afficher les 100 premières valeurs de la suite associée à $u_0 = 0.75$ et $\mu = 3.25$.

6. Afficher les 100 premières valeurs de la suite associée à $u_0 = 0.75$ et $\mu = 3.5$.

7. Afficher les 100 premières valeurs de la suite associée à $u_0 = 0.75$ et $\mu = 3.55$.

8. Afficher les 100 premières valeurs de la suite associée à $u_0 = 0.75$ et $\mu = 3.8$.

Exercice 6. Deux suites imbriquées

On considère les deux suites (a_n) et (b_n) définies par:

$$\begin{cases} a_0 = b_0 = 0 \\ a_{n+1} = a_n^2 - b_n^2 + c \\ b_{n+1} = 2a_nb_n + d \end{cases}$$

Écrire un programme qui demande à l'utilisateur la valeur des paramètres c et d puis qui affiche la valeur de $\sqrt{a_{1000}^2 + b_{1000}^2}$

Tester le programme pour les valeurs suivantes :

- $c = 0.091$ et $d = 0.606$
- $c = 0.091$ et $d = 0.608$. Dans ce cas, la suite diverge, donc le programme ne peut pas calculer.

Exercice 7. Suite de Héron

Soit la fonction f définie sur $\mathbb{R}^*$ par:

$$f(x) = \frac{1}{2}\left(x + \frac{2}{x}\right).$$

On considère la suite $(u_n)_n$ définie par:

$$\begin{cases} u_0 = \dfrac{3}{2}, \\ u_{n+1} = f(u_n), \qquad \forall n. \end{cases}$$

1. Écrire un programme qui permet de calculer et afficher les 10 premiers termes de cette suite.

2. Conjecturer le sens de variation et la limite de cette suite.

Exercice 8. Fonction récursive

1. Définir une fonction récursive (i.e. qui s'appelle elle-même) qui calcule les termes de la suite (u_n) définie par l'expression $u_{n+1} = \dfrac{2u_n^2 - 1}{u_n^2 + 2}$, avec $u_0 = 2$

2. Afficher en particulier les premiers termes jusqu'à u_{10}, puis jusqu'à u_{20}, et enfin jusqu'à u_{30}.

3. Cette méthode de calcul est simple, fonctionne, mais n'est clairement pas efficace. Expliquer pourquoi, quelle est la difficulté ?

4. Une autre manière de programmer les calculs des termes d'une telle suite est d'utiliser directement la relation de récurrence reliant deux termes consécutifs:

```python
def u(n):
    u=2
    for i in range(n):
        u=(2*u**2-1)/(u**2+2)
    return u

if __name__=="__main__":
    n=10
    for i in range(n+1):
        print(u(i))
```

Ce programme calcule puis affiche tous les premiers termes jusqu'à u_{10}. Utiliser ce programme pour afficher en particulier les termes u_{10}, u_{30}, et même u_{100} et pourquoi pas u_{1000} (utiliser la fonction `input()` pour lire l'indice n au clavier). Qu'observe-t-on pour des valeurs de plus en plus grandes de n ?

Exercice 9.
On considère la suite (u_n) définie par $u_0 = 1$ et par la relation de récurrence:

$$\forall n, \qquad u_{n+1} = f(u_n),$$

où la fonction f est définie sur l'intervalle $[0; +\infty[$ par

$$f(x) = 5 - \frac{4}{x+2}.$$

On définit de plus, pour tout entier naturel n, la somme (S_n) par

$$S_n = \sum_{k=0}^{n} u_k = u_0 + u_1 + \cdots + u_n.$$

1. Écrire une fonction qui permet de calculer les trois premiers termes de la suite (u_n). L'entier naturel n sera lu au clavier.
2. Écrire une fonction qui permet de calculer S_0, S_1 et S_2. Donner une valeur approchée des résultats à 10^{-2} près.

Exercice 10. Approximation de la fonction $\sin(x)$
Utiliser le développement en série entière :

$$\sin(x) \simeq \sum_{n=0}^{N} (-1)^n \frac{x^{2n+1}}{(2n+1)!}.$$

1. Écrire un programme qui calcule la valeur approchée de $\sin(x)$ pour un réel x donné par l'utilisateur. Vous choisirez une valeur pour N assez grande. Comparer avec la valeur exacte.

   On pourra remarquer que si on pose $u_n = \frac{x^{2n+1}}{(2n+1)!}$, alors on a que $u_n = \frac{x^2}{(2n+1)(2n)} u_{n-1}$.
2. Tester le programme avec les valeurs $x = 0$, $x = \pi$, $x = \pi/6$ et $x = 10\pi$
3. Pour chacune des valeurs précédentes calculer l'erreur en fonction de N lorsque l'on prend $N = 1$, $N = 2$, $N = 4$ et $N = 8$. Puis tracer l'erreur en fonction de N.

Exercice 11. Prêt bancaire

On considère un emprunt d'une somme de $C = 200000$ € à un taux annuel hors assurances de $t_a = 1.32\%$ sur $D = 20$ ans. On cherche à dresser le tableau d'amortissement de ce prêt permettant de visualiser mois par mois la mensualité, les intérêts remboursés, le capital remboursé et le capital restant dû.

Le calcul d'une mensualité fixe M est donné par la formule[1] :

$$M = \frac{Ct(1+t)^{12D}}{(1+t)^{12D} - 1}$$

où t est le taux mensuel, calculé à partir du taux annuel t_a (intérêts composés): $t = (1+t_a)^{1/12} - 1$

Chaque mois la mensualité permet de payer les intérêts sur le capital restant dû ainsi que de rembourser le capital. La mensualité étant fixe, on rembourse de moins en moins d'intérêts à mesure que le capital est remboursé.

Ainsi, au $i^{\text{ième}}$ mois, on paye $I_i = t(C - R_{i-1})$ d'intérêts et on rembourse $M - I_i$ au capital. $R_i = \sum_{k=1}^{i}(M - I_i)$ est le capital remboursé à l'issue du $i^{\text{ième}}$ mois.

1. Vérifiez que la dernière ligne du tableau correspond à un capital remboursé égal à C.

Mois	Mensualité	Intérêts	CapitalRemboursé	CapitalDu
1	947.90	218.68	729.22	199270.78
2	947.90	217.88	1459.24	198540.76
3	947.90	217.08	2190.06	197809.94
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
240	947.90	1.04	200000.00	0.00

Table 3.1: Tableau d'amortissement pour $C = 200000$, $t_a = 1.32\%$ et $D = 20$

[1] http://images.math.cnrs.fr/Emprunts-mensualites-interet-taux.html