

Correction du TP

Exercice 1 (★)

Écrire un programme qui demande trois réels a , b et c et qui affiche le nombre de racines réelles du polynôme du second degré $P : x \mapsto ax^2 + bx + c$. Selon les cas, votre programme devra afficher le message « P n'admet pas de racine », « P admet une racine » ou « P admet deux racines ».

Voici le programme demandé :

```

1 a = float(input('Donner une valeur non nulle pour a : '))
2 b = float(input('Donner une valeur pour b : '))
3 c = float(input('Donner une valeur pour c : '))
4 d = b^2-4*a*c
5 if d>0 :
6     print('P admet deux racines racines réelles.')
7 elif d==0 :
8     print('P admet une racine réelle.')
9 else :
10    print('P admet zero racine réelle.')
```

Exercice 2 (★)

- Définir la fonction f suivante sur Python : $f : x \mapsto \begin{cases} 1/x & \text{si } x \neq 0, \\ 0 & \text{sinon} \end{cases}$
- On donne le script suivant :

```

1 def g(x) :
2     if x < -1 :
3         y = -2
4     if x >= -1 and x<= 1 :
5         y = 2*x
6     if x > 1 :
7         y = 2
8     return y
```

- Déterminer l'expression mathématique de la fonction g ainsi définie.
- Proposer un script analogue au précédent qui utilise `elif`.

- On peut procéder comme suit :

```

1 def f(x) :
2     if x > 0 :
3         return 1/x
4     else :
5         return 0
```

- (a) Ce programme définit la fonction suivante :

$$g(x) = \begin{cases} -2 & \text{si } x < -1, \\ 2x & \text{si } -1 \leq x \leq 1, \\ 2 & \text{si } x > 1. \end{cases}$$

(b) Voici une proposition :

```

1 def g(x) :
2     if x < -1 :
3         y = -2
4     elif x <= 1 :
5         y = 2*x
6     else :
7         y = 1
8     return y

```

Exercice 3 (★)

On considère les sommes doubles suivantes :

$$S_n = \sum_{1 \leq i, j \leq n} \frac{i}{2^j} \quad \text{et} \quad T_n = \sum_{1 \leq i \leq j \leq n} \frac{i^3}{j(j+1)}.$$

- (a) Écrire un programme qui, étant donné un entier $n \geq 1$, calcule et affiche S_n .
(b) Même question pour T_n .
- Calculer à la main S_n et T_n en fonction de n .

- (a) Voici une proposition pour le calcul de S_n :

```

1 n = int(input('Donner n : '))
2 S = 0
3 for i in range(1, n+1) :
4     for j in range(1, n+1) :
5         S = i/2**j
6 print(S)

```

- (b) Voici une proposition pour le calcul de T_n :

```

1 n = int(input('Donner n : '))
2 T = 0
3 for i in range(1, n+1) :
4     for j in range(i, n+1) :
5         T = i**3/(j*(j+1))
6 print(T)

```

- Pour S_n , on a :

$$S_n = \left(\sum_{i=1}^n i \right) \times \left(\sum_{j=1}^n \left(\frac{1}{2} \right)^j \right) = \frac{n(n+1)}{2} \times \left(\frac{1 - \left(\frac{1}{2} \right)^n}{1 - \frac{1}{2}} \right) = \frac{n(n+1)}{2} \times \left(1 - \left(\frac{1}{2} \right)^n \right)$$

Pour T_n , on a :

$$\begin{aligned} T_n &= \sum_{j=1}^n \left(\sum_{i=1}^j \frac{i^3}{j(j+1)} \right) = \sum_{j=1}^n \frac{1}{j(j+1)} \left(\sum_{i=1}^j i^3 \right) = \sum_{j=1}^n \frac{1}{j(j+1)} \times \frac{j^2(j+1)^2}{4} = \sum_{j=1}^n \frac{j(j+1)}{4} \\ &= \frac{1}{4} \sum_{j=1}^n j^2 + \frac{1}{4} \sum_{j=1}^n j = \frac{n(n+1)(2n+1)}{24} + \frac{n(n+1)}{8}. \end{aligned}$$

Exercice 4 (★)

Soit $(u_n)_{n \in \mathbb{N}}$ une suite définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n^2 + 1$.

1. Définir une fonction `suite` qui, à un entier naturel n en entrée, renvoie la valeur de u_n .
2. (a) Montrer que la suite (u_n) est croissante, puis que $\lim_{n \rightarrow +\infty} u_n = +\infty$.
 (b) Définir une fonction `rang` qui, à un réel a , renvoie le rang du premier terme de la suite (u_n) qui est supérieur ou égal à a .

1. Voici la fonction demandée :

```

1 | def suite(n) :
2 |     u = 1
3 |     for k in range(1,n+1) :
4 |         u = u**2 + 1
5 |     return u
    
```

2. (a) Pour tout $n \in \mathbb{N}$, on a :

$$u_{n+1} - u_n = u_n^2 - u_n + 1 = P(u_n)$$

où $P : x \mapsto x^2 - x + 1$ est un polynôme de discriminant < 0 . Il est donc de signe constant, du signe de son coefficient dominant 1. On a donc $u_{n+1} - u_n > 0$ pour tout $n \in \mathbb{N}$ et la suite est croissante.

Par théorème de limite monotone, on a alors deux cas possibles : soit (u_n) converge vers une limite finie $\ell \in \mathbb{R}$, soit elle diverge vers $+\infty$. Supposons qu'elle converge vers $\ell \in \mathbb{R}$. On a $u_{n+1} = u_n^2 + 1$ pour tout $n \in \mathbb{N}$. D'où en passant à la limite, on obtient (puisque P est continue) :

$$\ell = \ell^2 + 1 \quad \Rightarrow \quad P(\ell) = 0.$$

Or ceci est impossible car P n'a pas de racine réelle. Ainsi on a bien que $\lim_{n \rightarrow +\infty} u_n = +\infty$.

Déjà vu ?

(u_n) est une suite récurrente d'ordre 1, du type $u_{n+1} = f(u_n)$ avec $f : x \mapsto x^2 + 1$. On sait que si une telle suite converge vers $\ell \in \mathbb{R}$, alors ℓ est nécessairement un point fixe de f , c'est-à-dire $\ell = f(\ell)$.

Pour un plan d'étude de telles suites et des exemples, je vous renvoie au

 **Complément de cours 0. Méthodes d'étude d'une suite récurrente d'ordre 1.** disponible à l'adresse mathieu-mansuy.fr/ecs2.

(b) Voici une possibilité de définition de la fonction `rang` :

```

1 | def rang(a) :
2 |     u = 1
3 |     n = 0
4 |     while u < a :
5 |         u = u**2 + 1
6 |         n = n+1
7 |     return n
    
```

Exercice 5 (★)

Soit $(u_n)_{n \in \mathbb{N}}$ la suite définie par $u_0 = \frac{1}{2}$ et $\forall n \in \mathbb{N}$, $u_{n+1} = \frac{2u_n}{u_n + 1}$. On admet que $\lim_{n \rightarrow +\infty} u_n = 1$.

1. Définir une fonction `valabs`, qui prend en entrée un flottant et qui renvoie sa valeur absolue.

2. À l'aide de la fonction `valabs`, écrire un programme permettant de déterminer le plus petit entier naturel n pour lequel on a :

$$|u_n - 1| \leq 10^{-3}.$$

1. Voici une définition de la fonction `valabs` :

```

1 | def valabs(x) :
2 |     if x < 0 :
3 |         return -x
4 |     else :
5 |         return x

```

Nous (re)verrons bientôt que la fonction valeur absolue est bien-sûr déjà définie dans Python, mais pas directement disponible. Elle nécessite d'importer le module `math`.

2. Cela nécessite l'utilisation d'une boucle `while`. On peut procéder comme suit :

```

1 | u = 1/2
2 | n = 0
3 | while valabs(u-1) > 10**(-3) :
4 |     u = (2*u) / (u+1)
5 |     n = n+1
6 | return n

```

Exercice 6 (★★)

- On considère la suite $(u_n)_{n \in \mathbb{N}}$ définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = 2nu_n + 3$.
Écrire une fonction `suiteu` prenant en entrée un entier naturel n et renvoyant la valeur de u_n .
- Même question avec $(v_n)_{n \in \mathbb{N}}$ définie par $v_0 = 1$, $v_1 = -2$ et pour tout $n \in \mathbb{N}$, $v_{n+2} = 2v_{n+1} - v_n$.
- On considère les suites $(a_n)_{n \in \mathbb{N}}$ et $(b_n)_{n \in \mathbb{N}}$ définies par $a_0 = 1$, $b_0 = 2$ et les relations :

$$\forall n \in \mathbb{N}, a_{n+1} = \frac{a_n^2}{a_n + b_n} \quad \text{et} \quad b_{n+1} = \frac{b_n^2}{a_n + b_n}.$$

Écrire une fonction `suites` ayant en entrée un entier naturel n et qui renvoie les réels a_n , b_n .

1. Voici la procédure pour calculer u_n :

```

1 | def suiteu(n) :
2 |     u = 1
3 |     for k in range(n) :
4 |         u = 2*k*u + 3
5 |     return u

```

2. Voici la procédure pour calculer v_n :

```

1 | def suitev(n) :
2 |     vavant = 1
3 |     vapres = -2
4 |     for k in range(n) :
5 |         aux = vapres
6 |         vapres = 2*vapres - vavant
7 |         vavant = aux
8 |     return vavant

```

3. Voici la procédure pour calculer a_n et b_n :

```

1 | def suites(n) :
2 |     a = 1
3 |     b = 2
4 |     for k in range(n) :
5 |         aaux = a
6 |         baux = b
7 |         a = aaux**2 / (aaux + baux)
8 |         b = baux**2 / (aaux + baux)
9 |     return a,b

```

Exercice 7 (★★)

Soient $(u_n)_{n \in \mathbb{N}^*}$ et $(v_n)_{n \in \mathbb{N}^*}$ les suites définies par : $\forall n \in \mathbb{N}^*$, $u_n = \sum_{k=1}^n \frac{1}{k^2}$ et $v_n = u_n + \frac{1}{n}$.

1. Montrer que les suites $(u_n)_{n \in \mathbb{N}^*}$ et $(v_n)_{n \in \mathbb{N}^*}$ sont adjacentes.

2. Écrire une fonction **approx** qui, à un réel strictement positif ε , associe une approximation de $\sum_{k=1}^{+\infty} \frac{1}{k^2}$ à ε près.

1. Montrons que $(u_n)_{n \in \mathbb{N}^*}$ et $(v_n)_{n \in \mathbb{N}^*}$ sont adjacentes :

- Croissance de $(u_n)_{n \in \mathbb{N}^*}$. Pour tout $n \in \mathbb{N}$, on a :

$$u_{n+1} - u_n = \sum_{k=1}^{n+1} \frac{1}{k^2} - \sum_{k=1}^n \frac{1}{k^2} = \sum_{k=1}^n \frac{1}{k^2} + \frac{1}{(n+1)^2} - \sum_{k=1}^n \frac{1}{k^2} = \frac{1}{(n+1)^2} \geq 0.$$

- Décroissance de $(v_n)_{n \in \mathbb{N}^*}$. Pour tout $n \in \mathbb{N}$, on a :

$$\begin{aligned} v_{n+1} - v_n &= \left(u_{n+1} + \frac{1}{n+1} \right) - \left(u_n + \frac{1}{n} \right) = \frac{1}{(n+1)^2} + \frac{1}{n+1} - \frac{1}{n} \\ &= \frac{n + n(n+1) - (n+1)^2}{n(n+1)^2} = \frac{-1}{n(n+1)^2} \leq 0. \end{aligned}$$

- Pour tout $n \in \mathbb{N}$, on a : $v_n - u_n = \left(u_n + \frac{1}{n} \right) - u_n = \frac{1}{n} \xrightarrow{n \rightarrow +\infty} 0$.

Les suites $(u_n)_{n \in \mathbb{N}^*}$ et $(v_n)_{n \in \mathbb{N}^*}$ sont donc adjacentes. D'après le théorème des suites adjacentes, elles convergent vers une même limite ℓ . De plus, on a pour tout $n \in \mathbb{N}^*$:

$$u_n \leq \ell \leq v_n.$$

2. On cherche le premier $n \in \mathbb{N}$ tel que $v_n - u_n = \frac{1}{n} \leq \varepsilon$. On utilise donc une boucle **while** :

```

1 | def approx(eps) :
2 |     n = 1
3 |     u = 1
4 |     while 1/n > eps :
5 |         n = n+1
6 |         u = u+1/(n**2)
7 |     return(u)

```

Exercice 8 (★★ - Raisonnement par dichotomie)

On considère une fonction f continue et strictement monotone sur un intervalle $[a, b]$ (avec $a < b$) telle que $f(a)f(b) < 0$. Le théorème de la bijection assure que l'équation $f(x) = 0$ possède une unique solution α sur $]a, b[$.

On construit deux suites (a_n) et (b_n) définies de la manière suivante : $a_0 = a, b_0 = b$ et pour tout entier naturel n :

- Si $f(a_n)f(\frac{a_n+b_n}{2}) \leq 0$, alors $\alpha \in]a_n, \frac{a_n+b_n}{2}[$. On pose $a_{n+1} = a_n$ et $b_{n+1} = \frac{a_n+b_n}{2}$.
- Si $f(a_n)f(\frac{a_n+b_n}{2}) > 0$, alors $\alpha \in]\frac{a_n+b_n}{2}, b_n[$. On pose $a_{n+1} = \frac{a_n+b_n}{2}$ et $b_{n+1} = b_n$.

1. Étude des suites (a_n) et (b_n) .

(a) Montrer que les suites (a_n) et (b_n) sont adjacentes et convergent vers α .

(b) Montrer que pour tout $n \in \mathbb{N}$, on a $b_{n+1} - a_{n+1} = \frac{b_n - a_n}{2}$, puis que $b_n - a_n = \frac{b-a}{2^n}$.

2. On considère l'équation $(E) : 1 - x - x^3 = 0$, où x désigne un réel. On note f la fonction définie sur \mathbb{R} par $f(x) = 1 - x - x^3$.

(a) Justifier que (E) possède une unique solution α , et vérifier que α appartient à $]0, 1[$.

(b) Écrire un script, incluant la définition de la fonction f par l'en-tête `def f(x)`, permettant de trouver et d'afficher une valeur approchée de α à 0,001 près.

1. (a) Commençons pour cela par montrer par récurrence que pour tout $n \in \mathbb{N}$, on a :

$$a_n \leq a_{n+1} \leq b_{n+1} \leq b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)f(b_n) \leq 0.$$

Init. Par construction, on a soit $a_1 = a_0$ et $b_1 = \frac{a_0 + b_0}{2}$, soit $a_1 = \frac{a_0 + b_0}{2}$ et $b_1 = b_0$. Comme $a_0 = a < b = b_0$, on a ainsi l'inégalité $a_0 \leq a_1 \leq b_1 \leq b_0$. D'autre part, $b_0 - a_0 = b - a = \frac{b-a}{2^0}$ et $f(a_0)f(b_0) = f(a)f(b) \leq 0$ par hypothèse. Donc la propriété est vraie au rang $n = 0$.

Hér. Soit n un entier naturel. Supposons la propriété au rang n vraie. Montrons la propriété au rang $n + 1$.

Par construction, on a soit $a_{n+2} = a_{n+1}$ et $b_{n+2} = \frac{a_{n+1} + b_{n+1}}{2}$, soit $a_{n+2} = \frac{a_{n+1} + b_{n+1}}{2}$ et $b_{n+2} = b_{n+1}$. Par hypothèse de récurrence, $a_{n+1} \leq b_{n+1}$. On a donc dans les deux cas l'inégalité $a_{n+1} \leq a_{n+2} \leq b_{n+2} \leq b_{n+1}$.

De plus, par construction, on a $b_{n+1} - a_{n+1} = \frac{b_n - a_n}{2}$. Par hypothèse de récurrence, $b_n - a_n = \frac{b-a}{2^n}$. Donc $b_{n+1} - a_{n+1} = \frac{b-a}{2^{n+1}}$.

Enfin, comme par hypothèse de récurrence $f(a_n)f(b_n) \leq 0$, $f(a_n)$ et $f(b_n)$ sont de signe opposé. Donc $f(\frac{a_n + b_n}{2})$ n'a pas le même signe que $f(a_n)$ ou $f(b_n)$. Par définition de a_{n+1} et de b_{n+1} , on a bien $f(a_{n+1})f(b_{n+1}) \leq 0$.

Donc la propriété au rang $n + 1$ est vraie.

Par le principe de récurrence, on a bien pour tout $n \in \mathbb{N}$:

$$a_n \leq a_{n+1} \leq b_{n+1} \leq b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)f(b_n) \leq 0.$$

On en déduit donc que (a_n) est croissante, (b_n) est décroissante, et on a :

$$\forall n \in \mathbb{N}, \quad b_n - a_n = \frac{b-a}{2^n} \xrightarrow{n \rightarrow +\infty} 0.$$

Les suites (a_n) et (b_n) sont donc adjacentes et, d'après le théorème des suites adjacentes, convergent vers la même limite $\ell \in]a, b[$.

Reste enfin à montrer que $\ell = \alpha$. En passant à la limite dans l'inégalité $f(a_n)f(b_n) \leq 0$, on obtient (en utilisant que f est continue sur $[a, b]$) :

$$\lim_{n \rightarrow +\infty} (f(a_n) \cdot f(b_n)) = \left(\lim_{n \rightarrow +\infty} f(a_n) \right) \left(\lim_{n \rightarrow +\infty} f(b_n) \right) = (f(\ell))^2 \leq 0.$$

D'où $(f(\ell))^2 \leq 0$, ce qui implique $f(\ell) = 0$. Et comme par hypothèse f s'annule en une seule valeur $\alpha \in]a, b[$, on a donc $\ell = \alpha$.

(b) On l'a démontré à la question précédente, on a pour tout $n \in \mathbb{N}$:

$$b_{n+1} - a_{n+1} = \frac{b_n - a_n}{2}.$$

La suite $(b_n - a_n)_n$ est géométrique de raison $\frac{1}{2}$. On a donc :

$$\forall n \in \mathbb{N}, \quad b_n - a_n = \left(\frac{1}{2}\right)^n (b_0 - a_0).$$

2. (a) On applique le théorème de la bijection.

- La fonction f est polynomiale, donc **continu** sur \mathbb{R} .
- f étant polynomiale, elle est aussi de classe \mathcal{C}^1 sur \mathbb{R} , et on a :

$$\forall x \in \mathbb{R}, \quad f'(x) = -1 - 3x^2 < 0.$$

Donc f est **strictement décroissante** sur \mathbb{R} .

- On a $\lim_{x \rightarrow -\infty} f(x) = +\infty$ et $\lim_{x \rightarrow +\infty} f(x) = -\infty$.

Par le théorème de la bijection, l'équation $f(x) = 0$ admet une unique solution α sur \mathbb{R} . De plus, on a :

$$-1 = f(1) < 0 = f(\alpha) < f(0) = 1.$$

Par stricte décroissance de f , on en déduit que $\alpha \in]0, 1[$.

(b) On peut utiliser le script suivant :

```

1 | def f(x) :
2 |     return 1-x-x**3
3 |
4 | a = 0
5 | b = 1
6 | while b-a>10**(-3):
7 |     m = (b-a)/2
8 |     if f(a)*f(m) <= 0 :
9 |         b = m
10 |    else a = m
11 | print(a)

```