TP 1

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 :

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

Exercice 2 (★)

1. Définir la fonction f suivante sur `Python` : $f : x \mapsto \begin{cases} 1/x & \text{si } x \neq 0, \\ 0 & \text{sinon} \end{cases}$

2. On donne le script suivant :

```
def g(x) :
    if x < -1 :
        y = -2
    if x >= -1 and x<= 1 :
        y = 2*x
    if x > 1 :
        y = 2
    return y
```

   (a) Déterminer l'expression mathématique de la fonction g ainsi définie.

   (b) Proposer un script analogue au précédent qui utilise `elif`.

1. On peut procéder comme suit :

```
def f(x) :
    if x > 0 :
        return 1/x
    else :
        return 0
```

2. (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 :

```
def g(x) :
    if x < -1 :
        y = -2
    elif x<= 1 :
        y = 2*x
    else :
        y = 1
    return y
```

Exercice 3 (★)

On considère les sommes doubles suivantes :

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

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

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

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

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

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

2. Pour S_n, on a :

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

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 $\lim\limits_{n\to+\infty} u_n = +\infty$.
   (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 :

```
def suite(n) :
    u = 1
    for k in range(1,n+1) :
        u = u**2 + 1
    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 $\lim\limits_{n\to+\infty} u_n = +\infty$.

> **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` :

```
def rang(a) :
    u = 1
    n = 0
    while u < a :
        u = u**2 + 1
        n = n+1
    return n
```

Exercice 5 (★)
Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = \dfrac{1}{2}$ et $\forall n \in \mathbb{N}$, $u_{n+1} = \dfrac{2u_n}{u_n + 1}$. On admet que $\lim\limits_{n\to+\infty} u_n = 1$.

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` :

```
def valabs(x) :
    if x < 0 :
        return -x
    else :
        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 :

```
u = 1/2
n = 0
while valabs(u-1) > 10**(-3) :
    u = (2*u) / (u+1)
    n = n+1
return n
```

Exercice 6 (★★)

1. 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.

2. Même question avec $(v_n)_{n\in\mathbb{N}}$ définie par $v_0 = 1$, $v_1 = -2$ et pour tout $n \in \mathbb{N}$, $v_{n+2} = 2v_{n+1} - v_n$.

3. 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 :

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

2. Voici la procédure pour calculer v_n :

```
def suitev(n) :
    vavant = 1
    vapres = -2
    for k in range(n) :
        aux = vapres
        vapres = 2*vapres - vavant
        vavant = aux
    return vavant
```

3. Voici la procédure pour calculer a_n et b_n :

```
def suites(n) :
    a = 1
    b = 2
    for k in range(n) :
        aaux = a
        baux = b
        a = aaux**2 / (aaux + baux)
        b = baux**2 / (aaux + baux)
    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 : $v_n - u_n = \left(u_n + \frac{1}{n}\right) - u_n = \frac{1}{n} \underset{n\to+\infty}{\longrightarrow} 0$.

   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` :

```
def approx(eps) :
    n = 1
    u = 1
    while 1/n > eps :
        n = n+1
        u = u+1/(n**2)
    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 $f(a_n)f\left(\frac{a_n+b_n}{2}\right) \leq 0$, alors $\alpha \in \left]a_n, \frac{a_n+b_n}{2}\right[$. On pose $a_{n+1} = a_n$ et $b_{n+1} = \frac{a_n+b_n}{2}$.
- Si $f(a_n)f\left(\frac{a_n+b_n}{2}\right) > 0$, alors $\alpha \in \left]\frac{a_n+b_n}{2}, b_n\right[$. 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 $b_{n+1} - a_{n+1} = \dfrac{b_n - a_n}{2}$, 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 $b_1 = \dfrac{a_0 + b_0}{2}$, soit $a_1 = \dfrac{a_0 + b_0}{2}$ 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, $b_0 - a_0 = b - a = \dfrac{b-a}{2^0}$ 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 $b_{n+2} = \dfrac{a_{n+1} + b_{n+1}}{2}$, soit $a_{n+2} = \dfrac{a_{n+1} + b_{n+1}}{2}$. 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 $b_{n+1} - a_{n+1} = \dfrac{b_n - a_n}{2}$. Par hypothèse de récurrence, $b_n - a_n = \dfrac{b-a}{2^n}$. Donc $b_{n+1} - a_{n+1} = \dfrac{b-a}{2^{n+1}}$.

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 $f(m_{n+1}) = f(\dfrac{a_n + b_n}{2})$ 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 :

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

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 $[a, b]$) :

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

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 **continue** sur $\mathbb{R}$.
- f étant polynomiale, elle est aussi de classe $\mathscr{C}^1$ 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 $\lim\limits_{x\to-\infty} f(x) = +\infty$ et $\lim\limits_{x\to+\infty} f(x) = -\infty$.

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 :

```
def f(x) :
    return 1-x-x**3

a = 0
b = 1
while b-a>10**(-3):
    m = (b-a)/2
    if f(a)*f(m) <= 0 :
        b = m
    else a = m
print(a)
```