Exercice 1 Approximation de π à l'aide d'une intégrale

Soit f une fonction dérivable sur l'intervalle $[a,b]$; on suppose qu'il existe un réel $M_1 \geqslant 0$ tel que pour tout $x \in [a,b]$ on a $\mid f'(x) \mid \leqslant M_1$.

Pour tout $n \in \mathbb{N}^\star$, on pose

$$R_n = \sum_{k=1}^{n} \frac{b-a}{n} f\left(a + k\frac{b-a}{n}\right).$$

On démontre que

$$\mid R_n - \int_a^b f(x)dx \mid \leqslant \frac{M_1 \mid b-a \mid^2}{2n}.$$

Soit f la fonction qui à tout réel x associe $f(x) = \dfrac{1}{x^2+1}$.

Ecrire un programme qui demande un réel $\varepsilon > 0$, puis calcule et affiche une valeur approchée à ε près de π.

Exercice 2 Polynôme de Lagrange

Soit $n > 0$ une suite strictement croissante $(x_0, x_2, ..., x_n)$ de $n+1$ nombres réels d'un intervalle I. Pour $k \in [\![0,n]\!]$, on définit la fonction polynôme L_k (polynôme de Lagrange) par

$$L_k(x) = \prod_{\substack{0\leqslant j\leqslant n \\ j\neq k}} \frac{x - x_j}{x_k - x_j}$$

a) Ecrire une fonction python qui prend en entrée $X = (x_0, x_1, ..., x_n)$, un entier $k \in [\![0,n]\!]$ et un réel x, puis calcule et affiche le nombre $L_k(x)$.

b) Supposons que la bibliothèque

```
import matplotlib.pyplot as plt
```

est importée, Ecrire un programme qui tracer sur l'intervalle $[-3,3]$ chacune des fonctions L_k correspond à la suite $X = (-2,-1,0,1,2)$.

Exercice 3Algorithme de Newton

On veut résoudre l'équation $f(x) = 0$ par la méthode dite « l'algorithme de Newton ».

— Soit I un intervalle contenant une unique solution x_0 de l'équation $f(x) = 0$ pour f dérivable et de dérivée non nulle sur I et $a \in I$ (donc $f'(a) \neq 0$).

— On définit par récurrence la suite $(a_n)_{n>0}$ par : $a_0 = a$ et, pour tout entier naturel n, $a_{n+1} = a_n - \dfrac{f(a_n)}{f'(a_n)} = g(a_n)$ pourvu que $g(I) \subset I$.

On Montre que, pour tout entier n, a_{n+1} est l'abscisse du point d'intersection de l'axe des abscisses avec la tangente à la courbe de f au point d'abscisse a_n.

On admet que, dans des conditions adéquates, la suite $(a_n)_n > 0$ converge et que sa limite est solution de l'équation $f(x) = 0$. De plus, on montre que a_n est une valeur approchée de cette solution à ε près lorsque $\mid a_{n+1} - a_n \mid \leqslant \varepsilon$.

Ecrire un programme python, qui calcule et affiche une valeur approchée à ε près de la solution de l'équation $x + \ln(x) = 0$ dans l'intervalle $]0;+\infty[$, et le nombre d'itérations réalisées. $a; \varepsilon$ deux réels données par l'utilisateur.

Exercice 4 *Algorithme de HORNER*

Soit $P(x) = \sum_{i=0}^{n} a_i x^i$ est une fonction polynôme et x_0 un nombre réel donné.
Pour déterminer la valeur de $P(x_0)$ on utilise l'algorithme de Horner :
— on multiplie le premier coefficient par x_0 puis on ajoute au produit le second coefficient.
— On multiplie alors le nombre obtenu par x_0 puis on ajoute le troisième coefficient,
— etc ...on obtient alors

$$P(x_0) = ((...((a_n x_0 + a_{n-1})x_0 + a_{n-2})x_0 + ...)x_0 + a_1)x_0 + a_0$$

Ecrire une fonction python **eval_poly** prenant comme arguments, une fonction polynôme P, un réel x_0 et renvoie la valeur de $P(x_0)$ en utilisant la méthode de HORNER.

Exercice 5 2022

Soient b, c et d des constantes réelles et considérons l'équation $f(x) = 0$ avec $f(x) = x^3 + bx^2 + cx + d$ pour $x \in \mathbb{R}$.

1. Justifier brièvement pourquoi il existe un entier naturel m tel que $f(m)f(-m) \leqslant 0$.
2. Compléter en Python la fonction trouver_chg_signe ci-dessous afin qu'elle renvoie le plus petit entier naturel m vérifiant $f(m)f(-m) < 0$.

```
def trouver_chg_signe(b, c, d):
    def pol3(x):
       return x**3 + b*x**2 + c*x + d
    m = ________
    while ________________________:
        m = __________
    return m
```

On utilise une méthode de dichotomie pour trouver une solution de l'équation dans $[-m, m]$: on considère les suites(a_n) et (b_n) définies comme suit : $a_0 = -m, b = m$ et pour tout $n \geqslant 0$

$$(a_{n+1}, b_{n+1}) = \begin{cases} (a_n, c_n) & \text{si} f(c_n)f(a_n) \leqslant 0, \\ (c_n, b_n) & \text{sinon,} \end{cases}$$

où on a posé $cn = (a_n + b_n)/2$ pour $n \geqslant 0$. Si on se donne un réel $\varepsilon > 0$ définissant la précision souhaitée, on considèrera que la solution est atteinte quand$|b_n - a_n| < \varepsilon$. Dans ce cas, on peut choisir $c_n = (a_n + b_n)/2$ comme solution.

Compléter en Python la fonction `madichotomie` ci-dessous afin qu'elle renvoie une solution de l'équation f(x) = 0 obtenue par cette méthode de dichotomie avec une précision epsilon. Cette fonction peut faire appel à la fonction `trouver_chg_signe`.

```
def madichotomie(b, c, d, epsilon):
   def pol3(x):
       return x**3 + b*x**2 + c*x + d
   m = _________________________
   an = _______
   bn = _______
while (________________________________________):
    cn = (an + bn)/2
    if (_________________________________):
        _______ = cn
    else:
       _______ = cn
    res = ___________
return res
```

Exercice 6 (2022)

1. Déterminer trois réels a, b, et c tels que :

$$\forall x \in \mathbb{R} \setminus \{-1; 0\}, \frac{1}{x(x+1)^2} = \frac{a}{x} + \frac{b}{x+1} + \frac{c}{(x+1)^2}$$

2. En déduire que, pour tout entier naturel k non nul, on a :

$$\sum_{n=k}^{+\infty} \frac{1}{n(n+1)^2} \leqslant \frac{1}{k(k+1)}.$$

3. Soit$(a_n)_{n\in\mathbb{N}^*}$ une suite de réels strictement positifs.
(a) On suppose qu'on dispose d'une fonction Python `a` telle que pour tout entier naturel n non nul, `a(n)` soit égal àa_n.
Écrire en Python une fonction prenant en argument un entier n et renvoyant la somme $\displaystyle\sum_{k=1}^{n} \frac{k}{a_1 + a_2 + ...a_k}$.

On cherchera à limiter au maximum le nombre d'appels à la fonction `a`.

(b) Démontrer que :

$$\forall n \in \mathbb{N}^*, \frac{n^2(n+1)^2}{4} \leqslant (a_1 + a_2 + ...a_n) \sum_{k=1}^{n} \frac{k^2}{a_k}.$$

(c) On suppose que la série $\displaystyle\sum_{n\geqslant 1} \frac{1}{a_n}$ est convergente. Montrer que la série $\displaystyle\sum_{n\geqslant 1} \frac{n}{a_1 + a_2 + ...a_n}$ est converge et qu'il existe un réel $K > 0$ ne dépendant pas de la suite $(a_n)_{n\in\mathbb{N}^*}$ telle que :

$$\sum_{n=1}^{+\infty} \frac{n}{a_1 + a_2 + ...a_n} \leqslant K \sum_{k=1}^{+\infty} \frac{1}{a_k}.$$

Exercice 7 Escp 2022

Soit a un réel strictement positif.

1. Étudier la nature de la série $\displaystyle\sum_{n\in\mathbb{N}} (arctan(n+a) - arctan(n))$.
2. Proposer un programme Python permettant d'en donner une valeur approchée à 0.001 près quand $a = \dfrac{1}{2}$

On rappelle que $atan(x)$ renvoie la valeur de $arctan(x)$

Exercice 8 EML 2015

On considère l'application $f : \mathbb{R} \longrightarrow \mathbb{R}, x \longmapsto f(x) = x^3 e^x$, et la série convergente $\displaystyle\sum_{n\geqslant 1} \frac{1}{f(n)}$.

On note $S = \displaystyle\sum_{n=1}^{+\infty} \frac{1}{f(n)}$ on a

$$\forall n \in \mathbb{N}^* \mid S - \sum_{k=1}^{n} \frac{1}{f(k)} \mid \leqslant \frac{1}{(e-1)\, e^n}$$

Ecrire fonction en Python qui calcule une valeur approchée de S à 10^{-4} près.

Exercice 9 Ecricome 2015 ECS

1. On pose pour tout entier naturel n, $a_n = sin\left(\dfrac{\pi}{3{\times}2^n}\right)$ et $b_n = cos\left(\dfrac{\pi}{3{\times}2^n}\right)$.

L'étude mathématique a permis de montrer que : $\forall n \in \mathbb{N}$, $a_{n+1} = \sqrt{\dfrac{1-b_n}{2}}$ $(*)$ et$b_{n+1} = \sqrt{\dfrac{1+b_n}{2}}$ $(**)$

Puis que pour tout $n \in \mathbb{N}$: $\quad 9 \times 2^n \dfrac{a_n}{2+b_n} < \pi < 2^n \left(2a_n + \dfrac{a_n}{b_n}\right)$,

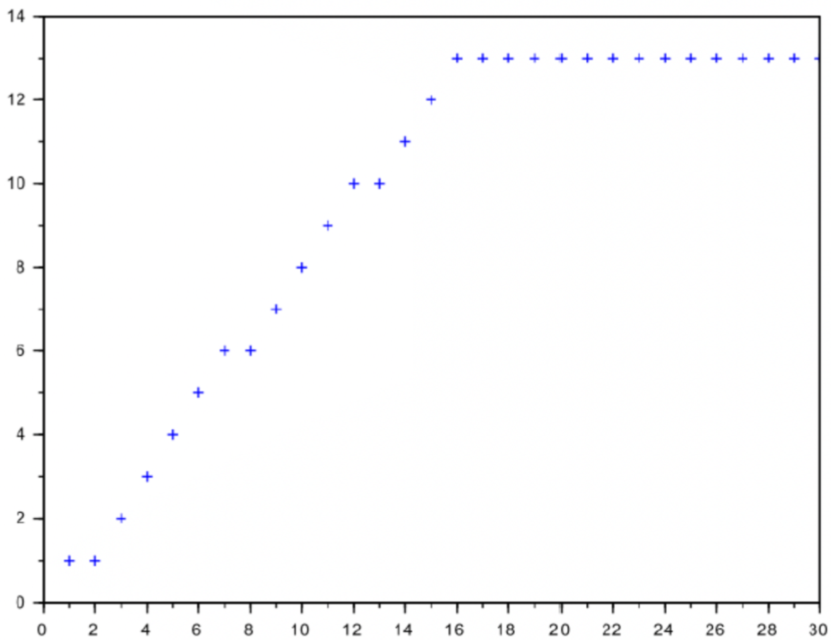
les deux termes de l'encadrement tendant vers π quand n tend vers l'infini.

2. Compléter la fonction python suivante afin qu'elle retourne, à l'aide des relations (*) et (**), une approximation$x$ deπ à e près, ainsi que le nombre k d'itérations qui ont été nécessaires.

```
def h(e):
    k = 0
    a =sqrt(3)/2
    b = 1/2
    while ......................:
        a = ....................
        b = ....................
        k = ....................
   x =..........................
   return x,k
```

3. On souhaite étudier l'évolution du nombre d'itérations nécessaires en fonction de la précision souhaitée. Ecrire une fonction python qui prend comme paramètre d'entrée un entier p et qui retourne une liste de taille p qui contient les nombres d'itérations nécessaires pour les précisions 10^{-k}, pour $k \in \{1, 2, ..., p\}$.

4. On utilise la fonction précédente avec $p = 30$ et on représente graphiquement les valeurs obtenues. On obtient le graphe suivant :



Commenter ce graphe.