Exercice 1. Soit $f : \mathbb{R} \to \mathbb{R}$ l'application définie par $f(x) = \ln(1 - \ln(x))$.

1. Déterminer l'ensemble de définition D_f de f.

2. Tracer avec Python la courbe $x \mapsto f(x)$ sur l'intervalle $]0, \mathrm{e}[$. On pourra utiliser la commande `linspace`.

3. Donner le tableau de variation de f .

4. Montrer que f admet une fonction réciproque g . Déterminer g .

5. Montrer que $f(x) = \ln(2)$ admet une unique solution $\alpha \in D_f$. Calculer α.

6. Montrer que f'' s'annule une seule fois en un réel β que l'on calculera.

7. Donner la tangente en β à $\mathscr{C}_f$ et la position relative par rapport à $\mathscr{C}_f$.

Exercice 2. 1. Rappeler le développement limité en 0 de $\ln(1 + x)$. En déduire, lorsque n tend vers $+\infty$, que

$$\ln(1 + n) = \ln(n) + \frac{1}{n} - \frac{1}{2n^2} + o\left(\frac{1}{n^2}\right).$$

2. On considère la suite $(u_n)_{n\in\mathbb{N}^\star}$ définie par $u_n = \ln(n) - 2\ln(n + 1) + \ln(n + 2)$.

(a) Écrire en langage Python une fonction `suite(n)` prenant en paramètre un entier naturel non nul `t` et retournant la valeur de u_n.

(b) Calculer u_{10}, u_{100} et u_{1000}. Que peut-on observer ?

(c) En utilisant la première question, montrer que la suite $(u_n)_{n\in\mathbb{N}^\star}$ converge.

3. On considère la suite $(S_n)_{n\in\mathbb{N}^\star}$ définie par $S_n = \sum_{k=1}^{n} u_k$.

(a) Écrire en langage Python une fonction `somme(n)` prenant en paramètre un entier naturel non nul `n` et retournant la valeur de S_n.

(b) Calculer S_{10}, S_{100} et S_{1000}. Que peut-on observer ?

(c) Montrer que $S_n = \ln(n + 2) - \ln(n + 1) - \ln(2)$ pour tout $n \in \mathbb{N}^\star$.

(d) En déduire que $\sum u_n$ converge et donner sa valeur.

Exercice 3. On considère pour $n \in \mathbb{N}^\star$ l'intégrale

$$I_n = \int_1^{+\infty} \frac{t^{-n}}{1 + t}\,\mathrm{d}t.$$

1. Montrer que I_n existe pour tout $n \in \mathbb{N}^\star$. Calculer I_1.

2. Montrer que la suite (I_n) est décroissante.

3. En majorant l'intégrale, montrer que (I_n) converge vers 0.

4. Calculer $I_n + I_{n+1}$. En déduire un équivalent de (I_n).

5. En utilisant la question 4, écrire en langage Python une fonction `suite(n)` prenant en paramètre un entier naturel `n` et retournant la valeur de I_n.

Exercice 4. On définit pour $n \in \mathbb{N}$, le nombre

$$I_n = \int_0^{\pi/2} \sin(t)^n\,\mathrm{d}t.$$

1. Calculer I_0, I_1 et I_2.

2. Montrer que la suite (I_n) est décroissante.

3. Montrer que pour tout $n \in \mathbb{N}$, on a (n +2)In+2 = (n +1)In.

4. (a) Écrire en langage Python une fonction `suite(n)` prenant en paramètre un entier naturel `n` et retournant la valeur de I_n

(b) Calculer I_{10}, I_{100} et I_{1000}. Que peut-on observer ?

5. En utilisant la question 3, déterminer une expression de I_{2p} et I_{2p+1} pour $p \in \mathbb{N}$.

Exercice 5. Soit $f : \mathbb{R} \to \mathbb{R}$ la fonction définie par $f(x) = \frac{1}{2}(x + \cos(x))$. On définit la suite (x_n) par $x_0 = 1$ et $x_{n+1} = f(x_n)$ pour tout $n \in \mathbb{N}$.

1. (a) Écrire en langage Python une fonction `suite(n)` prenant en paramètre un entier naturel `n` et retournant la valeur de x_n.

(b) Calculer x_{10}, x_{100} et x_{1000}. Que peut-on observer ?

2. Montrer que la fonction f est croissante.

3. Montrer que la suite (x_n) est positive et décroissante.

4. En déduire que la suite converge vers un réel $\ell \in \mathbb{R}$ vérifiant $\cos(\ell) = \ell$.

5. Combien de solution l'équation $\cos(x) = x$ admet-elle pour $x \in \mathbb{R}$.

Numpy : import numpy as np

np.array(L) ---- Transforme la liste L en matrice numpy
np.zeros([n,m]) ---------- matrice nulle de taille $n \times m$
np.eye(n) ---------- matrice unité de taille n
np.diag(L) ---------- matrice diagonale
np.transpose(M) ---------- transposée de M
np.dot(M,P) ---------- produit matriciel MP
np.sum(M) ---------- somme de tous les éléments de M
np.max(M) ---------- le plus grand élément de M
np.min(M) ---------- le plus petit élément de M
np.shape(M) ---------- format de la matrice M
np.arange(a,b,p) ---- nombres ente a et b avec un pas p.

Liste

[] ---------- une liste vide
$[a] * n$ - une liste avec n fois l'élément a
L.append(a) ------ Ajoute a à la fin de L
L1 + L2 --- Concatène les listes $L1$ et $L2$
len(L) -- nombre d'éléments de la liste L
L.pop(k) --- enlève l'élément d'indice k
L.remove(a) ------- enlève une fois a à L
max(L) ----- le plus grand élément de L
min(L) ------- le plus petit élément de L
sum(L) ------ somme des éléments de L

random : import random as rd

rd.random() ---------- un nombre au hasard entre 0 et 1, 1 exclus
rd.randint(a,b) ---------- un nombre entier au hasard entre a et b inclus
rd.choice(L) ---------- Choisit aléatoirement un élément de la liste.

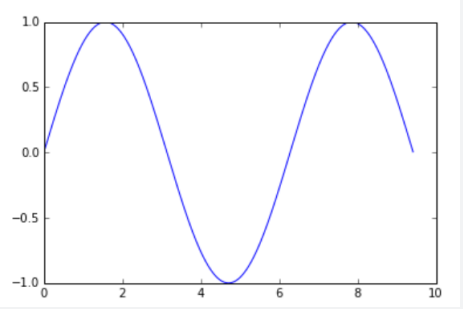
math : import math as m

m.atan(x) ---------- $\arctan(x)$
m.factorial(n) ---------- $n!$ si $n \in \mathbb{N}$
m.sqrt(x) ---------- $\sqrt{x}$ si $x \geq 0$
m.log(x) ---------- $\ln(x)$ si $x > 0$
m.exp(x) ---------- e^x
m.sin(x) ---------- $\sin x$
m.cos(x) ---------- $\cos x$

Tracé de courbe

```
import matplotlib.pyplot as plt

def f(x):
    return math.sin(x)
X = np.arange(0,3*np.pi,0.01)
Y = [ f(x) for x in X ]
plt.plot(X, Y)
plt.show()
```



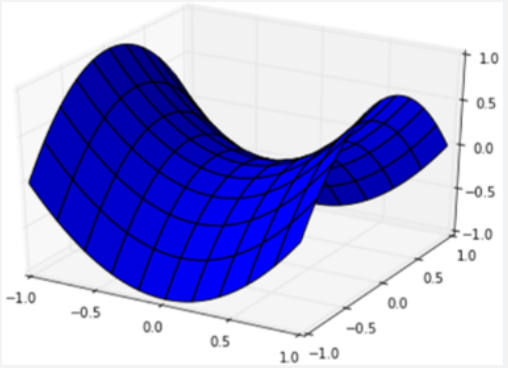
numpy.linalg import numpy.linalg as la

la.inv(M) ----- l'inverse de la matrice M
la.eig(M) liste des valeurs propres de M et matrice de passage associée
la.matrix_rank(M) ---------- rang de M
la.det(M) ---------- déterminant de M

Tracé de surfaces

```
from mpl_toolkits.mplot3d
    import Axes3D
import matplotlib.pyplot as plt

ax = Axes3D(plt.figure())
def f(x,y):
    return x**2- y**2
X = np.arange(-1,1,0.02)
Y = np.arange(-1,1,0.02)
X, Y = np.meshgrid(X, Y)
Z = f(X, Y)
ax.plot_surface(X, Y, Z)
plt.show()
```



Tracé de lignes de niveau

```
import matplotlib.pyplot as plt
def f(x,y):
  return x**2+ y**2+ x*y
X = np.arange(-1,1,0.01)
Y = np.arange(-1,1,0.01)
X, Y = np.meshgrid(X, Y)
Z = f(X, Y)
plt.axis('equal')
plt.contour(X, Y, Z,
    [0.1,0.4,0.5])
plt.show()
```