Chapitre (NUM) 1 Bibliothèque `matplotlib`

L'informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes.

— **M. FELLOWS et I. PARBERRY**

Résumé & Plan

Tracer un graphe de fonction ou suite peut donner beaucoup d'informations sur cette dernière : la monotonie, les limites éventuelles, *etc.*. L'objectif de ce court chapitre est de se familiariser avec le module dédié à cela en Python : `matplotlib`. Nous développons aussi quelques algorithmes relatifs aux fonctions.

- Les chapitres d'Informatique sont composés de cours et d'exercices intégrés. Le cours sera projeté au tableau.
- Il n'est pas attendu que toute la classe aborde tous les exercices. Traitez donc en priorité les exercices présents dans la liste donnée à chaque début de séance.
- Exercices 💣 / **Pour aller plus loin** : exercices plus difficiles, ou plus techniques. À ne regarder que si les autres sont bien compris.

Fichier externe ?

NON *pas de fichier externe dans ce TP*

Dans tout ce TP, on supposera importé le module `matplotlib` de la manière suivante.

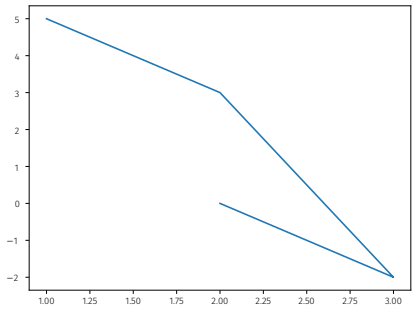
```
import matplotlib.pyplot as plt
```

1 COMMANDES PRINCIPALES

1.1 Généralités

Exemple 1 (Introductif) Commençons par un exemple illustrant le fonctionnement de `matplotlib`. Plaçons par exemple les points de coordonnées $(1,5)$, $(2,3)$, $(3,-2)$ et $(2,0)$, dans cet ordre, sur un graphique.

```
X = [1, 2, 3, 2]
Y = [5, 3, -2, 0]
plt.plot(X, Y)
plt.show()
```



On notera que par défaut, les points sont reliés par des segments de droite. Python a donc relié ici successivement les 4 points de $\mathbb{R}^2$ mentionnés.

Commande	Effet
`plt.plot(X, Y)`	Trace le nuage de points d'abscisses `X` et ordonnées `Y`
`plt.text(x, y, "texte")`	affiche `texte` au point de coordonnées `(x, y)`
`plt.savefig("chemin.eps")`	enregistre le graphique obtenu

Attention

La commande `savefig` doit être placée avant `show` et après `plot`.

COMMANDES DE PERSONNALISATION DU GRAPHIQUE. Voici quelques autres options pour les paramètres optionnels de la commande `plt.plot`, ils se placent après `color = 'r', linewidth = 4` dans la commande ci-dessus.

PARAMÈTRES OPTIONNELS POUR `plot`

Propriétés	Rôle	Valeurs possibles
`color`	Trace le nuage de points	`'red'`, `'blue'` *etc.* d'abscisses `X` et ordonnées `Y`
`label`	Légende de la courbe	une chaîne
`linestyle`	type de ligne (pointillés, etc.)	`'-'`, `'-.'`, `':'`
`linewidth`	épaisseur du trait	un entier
`marker`	forme des points	`'+'`, `','`, `'o'`, `'1'`, `'2'` *etc.* (`'o'` adapté aux suites)

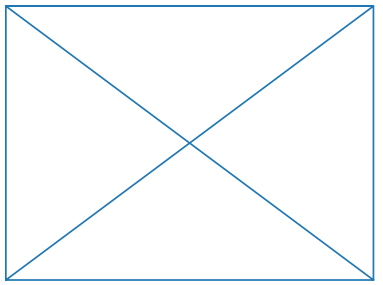
OPTIONS POUR `linestyle`

Option	Rôle
`""`	les points ne sont pas reliés
`"-"`	tracé en trait plein
`"--"`	tracé avec des tirets
`":"`	tracé en pointillés
`"-."`	alternance de tirets et de points

COMMANDES DE FORMATAGE DES TITRES, LÉGENDES, AXES

Nom	Rôle
`plt.legend(loc = 'upper left')`	permet de placer les légendes (ici haut gauche)
`plt.title('titre')`	permet de placer un titre au tracé
`plt.axis('off')`	permet d'enlever les axes
`plt.axis('scaled')`	impose la même échelle aux deux axes
`plt.axis(xmin = .., xmax = ..,..)`	impose des valeurs min/max à chaque axe en spécifiant `xmax`, `ymax`, `xmin`, `ymin`
`plt.grid(True)`	trace le quadrillage
`plt.xlabel('texte')`	nom pour l'axe des abscisses
`plt.ylabel('texte')`	nom pour l'axe des ordonnées

Exercice 1 | [Solution] Dessiner à l'aide de commande(s) `plot` le carré de côtés $(0,0),(1,0),(1,1),(0,1)$ avec ses diagonales. *On veillera à enlever les axes dans cet exercice.* L'objectif est donc d'avoir ce dessin :



1.2 Tracé d'une fonction

En général, pour les fonctions :

- la liste `X` est construite en utilisant la fonction `linspace` du module `numpy`. Elle permet de découper l'intervalle de tracé avec un grand nombre de points régulièrement espacés.
- La liste `Y` est alors construite par compréhension à l'aide de `X`.

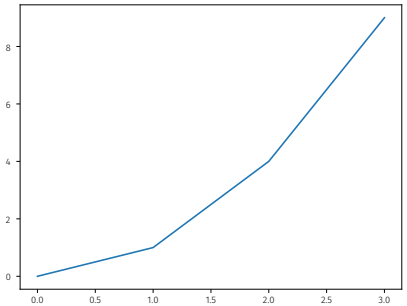
Voyons déjà un exemple d'utilisation de `linspace`.

```
>>> import numpy as np
>>> np.linspace(0, 3, 3) # découpage de l'intervalle en 3 points
array([0. , 1.5, 3. ])
>>> np.linspace(-1, 2, 10) # découpage de l'intervalle en 10 \
↪ points
array([-1.        , -0.66666667, -0.33333333,  0.        ,  0.333333
33,
        0.66666667,  1.        ,  1.33333333,  1.66666667,  2.     \
        ↪    ])
```

Remarque 1 Vous constaterez dans les résultats précédents que le type du résultat d'un `linspace` n'est pas vraiment une liste ; c'est un *tableau numpy*, que nous étudierons un peu plus tard dans l'année (Chapitre (NUM) 3).

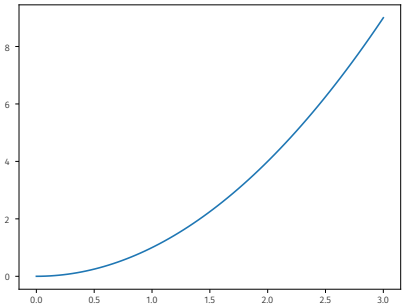
Exemple 2 Représentons la fonction carré sur $[0,3]$ grâce à des compréhensions de listes. On a besoin de découper $[0,3]$ en un très grand nombre de points, on utilise alors `linspace`.

```
X = np.linspace(0, 3, 4)
Y = [x**2 for x in X] # liste des carrés
plt.plot(X, Y)
```



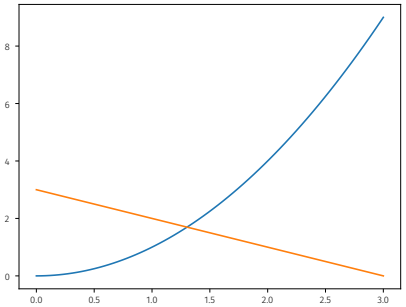
On observe que, par défaut, Python trace une ligne brisée bleue reliant les points donnés. Pour obtenir l'illusion d'une courbe, il suffit de placer suffisamment de points, dans la pratique on prendra par exemple `10**3`.

```
X = np.linspace(0, 3, 10**3) # on augmente le nombre de points \
↪ ici
Y = [x**2 for x in X] # liste des carrés
plt.plot(X, Y)
```



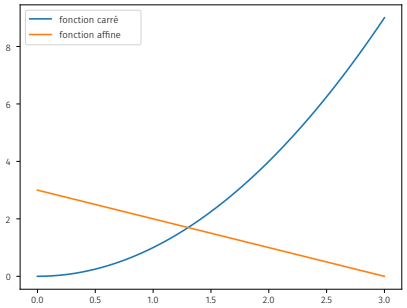
En enchaînant plusieurs `plot`, on peut tracer plusieurs courbes sur un même graphique.

```
X = np.linspace(0, 3, 10**3) # on augmente le nombre de points \
↪ ici
Y = [x**2 for x in X] # liste des carrés
plt.plot(X, Y)
Z = [3-x for x in X] #
plt.plot(X, Z)
```



On peut aussi ajouter un label, comme ci-dessous.

```
X = np.linspace(0, 3, 10**3) # on augmente le nombre de points \
↪ ici
Y = [x**2 for x in X] # liste des carrés
plt.plot(X, Y, label="fonction carré")
Z = [3-x for x in X] #
plt.plot(X, Z, label="fonction affine")
plt.legend()
```



Exercice 2 | Ensemble de fonctions puissances [Solution] Donner plusieurs instructions permettant de représenter sur un même graphique les fonctions $f : x \longmapsto x^\alpha$ pour $\alpha \in \{0.5, 1, 2\}$ sur $[0, 5]$. *On affichera en légende la puissance en question.*

Voici la méthode générale.

Méthode (NUM) 1.1 (Tracer une fonction f (et/ou sa réciproque)) Soit f une fonction définie sur $[a, b]$, avec $a, b \in \mathbb{R}, a < b$, définie sur cet intervalle et que l'on souhaite tracer.

1. On commence par définir la fonction f dans Python avec :
```
def f(x):
    return # expression de f(x)
```
2. On crée les listes X et Y, puis on trace.
```
X = np.linspace(a, b, 10**3)
Y = [f(x) for x in X]
plt.plot(X, Y)
plt.show()
```
3. Si on souhaite ajouter le graphe de la réciproque, on exploite la propriété déjà vue dans le cours de Mathématiques : les graphes sont symétriques par rapport à $y = x$, cela revient à échanger X et Y dans la commande `plot`.
```
X = np.linspace(a, b, 10**3)
Y = [f(x) for x in X]
plt.plot(Y, X)
plt.show()
```

On peut aussi enchaîner plusieurs `plot`, il est alors intéressant de spécifier des légendes pour identifier chaque courbe. La couleur quant à elle change automatiquement à chaque `plot`.

Exercice 3 | Fonctions trigonométriques [Solution]

1. Créer une fonction d'en-tête `trace_trigo(a, b)` qui prend en argument deux réels a, b tels que $a \leqslant b$, et trace les graphes des fonctions cos, sin sur $[a, b]$.
2. Écrire une fonction d'en-tête `trace_tanatan()` sans arguments, qui trace sur $\left]-\frac{\pi}{2}, \frac{\pi}{2}\right[$ la fonction tan, et superpose celle d'arctan. Ajouter en pointillés la droite d'équation $y = x$.

Dans tout cet exercice, on pourra utiliser le module numpy pour accéder aux deux fonctions trigonométriques `np.cos`, `np.sin`, et ***uniquement celles-ci****. Par ailleurs, une valeur approchée de π existe avec `np.pi`.*

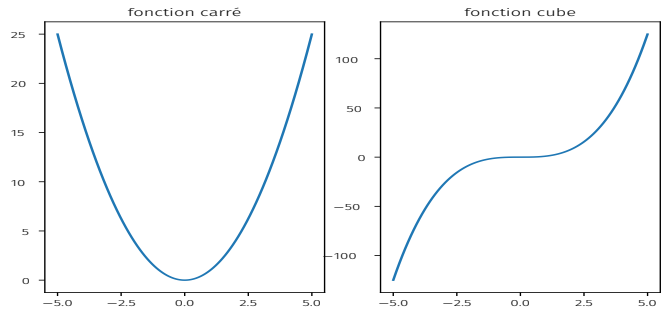
1.3 Grille de graphiques

On a déjà vu comment dessiner plusieurs courbes sur la même figure, en invoquant plusieurs fois la fonction `plot()` avant le `show()` final. On peut aussi souhaiter tracer des courbes sur plusieurs figures au sein d'une même fenêtre graphique. Pour cela, on utilisera la fonction `subplot`.

Nom	Rôle
`plt.subplot(N1, N2, i)`	crée la figure de la case i d'un tableau de N1 lignes et N2 colonnes

Chaque `subplot()` définit une figure de notre fenêtre graphique, et celle-ci possède ses propres labels, titres et légendes. Par contre, les figures sont un peu petites. Voyons un exemple.

Exemple 3 (Tracer une grille de graphes) Représentons les fonctions $x \longmapsto x^n$ pour $n = 2, 3$ sur deux graphes côte à côte, sur l'intervalle $[-5, 5]$.
```
X = np.linspace(-5, 5, 10**3)
Y = [x**2 for x in X] # liste des carrés
Z = [x**3 for x in X] # liste des puissances 3
plt.subplot(1, 2, 1)
plt.title("fonction carré")
plt.plot(X, Y)
plt.subplot(1, 2, 2)
plt.title("fonction cube")
plt.plot(X, Z)
```



Exercice 4 | **Grille de taille** 2×2 [Solution] On souhaite comparer les fonctions $f : x \longmapsto \mathrm{e}^{1/x} - 1$ et $g : x \longmapsto 1/x$ au voisinage de l'infini.

1. Définir deux fonctions Python `f` et `g`, correspondant aux fonctions mathématiques f et g.
2. Écrire une fonction `trace_grille()` sans argument, qui trace dans un carré 2×2 de figures les courbes de $f, g, f - g, \frac{f}{g}$ sur $[10, 30]$. Que conjecturer ?
3. Retrouver $\lim\limits_{x \to \infty} \frac{f(x)}{g(x)}$ par le calcul.

2 CONJECTURER DES PROPRIÉTÉS SUR LES FONCTIONS

Dans cette section, les domaines de tracés et d'autres éléments sont laissés à votre appréciation afin de répondre aux questions posées à l'aide de Python.

2.1 Conjecturer une bijectivité

Exercice 5 | **Sinus hyperbolique** [Solution] On définit sur $\mathbb{R}$ les fonctions

$$f : x \longmapsto \frac{\mathrm{e}^x - \mathrm{e}^{-x}}{2}, \quad g : x \longmapsto \ln\left(x + \sqrt{x^2 + 1}\right).$$

1. Écrire une fonction d'en-tête `trace_fg()` représentant sur une même figure les courbes des fonctions f, g sur $[-3, 3]$. Ajouter au tracé l'instruction suivante :

```
plt.plot([-10, 10], [-10, 10], ':')
```

Qu'observe-t-on ? *On pourra se servir des fonctions* `np.log` *et* `np.exp` *du module numpy.*

2. Afficher, directement dans la console, une liste de valeurs de $f \circ g - \mathrm{Id}, g \circ f - \mathrm{Id}$ arrondies à deux décimales, pour x dans $[-2, 2]$. Que peut-on conjecturer ? Pour l'arrondi, on pourra se servir de `round`. Par exemple :

```
>>> round(1.123456, 2)
1.12
```

3. Démontrer mathématiquement la conjecture.

2.2 Approximation d'extremum [H.P]

On souhaite dans cette section trouver une méthode afin d'approcher le minimum global ou local d'une fonction en cas d'existence. Commençons par une première méthode.

Comme cela est précisé par le logo [H.P], cette dernière partie doit être considérée comme des exercices supplémentaires pour s'entraîner, dont le contenu n'est pas à apprendre.

Exercice 6 | **Approximation d'un** argmin **par discrétisation** [Solution]

On considère une fonction $f : I = [a, b] \longrightarrow \mathbb{R}$, avec $a < b$ deux rels, telle que f possède un minimum et un maximum global (c'est le cas par exemple si f est continue). Une idée est de réaliser une recherche de minimum ou maximum des valeurs de f sur une subdivision de $[a, b]$.

1. Soit `L` une liste contenant des valeurs de l'intervalle $[a, b]$ (un `linspace` dans la suite). Écrire une fonction d'en-tête `maximum_sub(L, f)` qui renvoie le maximum des valeurs que prend `f` sur la liste des éléments de `L` ainsi que la valeur de l'abscisse correspondante (un élément de `L`). Par exemple, si f est la fonction carré, et `L = [-2, 0, 1]` alors la fonction renverra le tuple `(4, -2)`. *Pour tester, on commencera donc par définir la fonction carré dans Python, en tapant :*

```
def f(x):
    return x**2
```

On pourra compléter le code ci-après.

```
def maximum_sub(L, f):
    ind_maxi = 0
    maxi = _______
    for i in range(1, len(L)):
        if _______ > maxi:
            ind_maxi = i
            maxi = _______
    return maxi, _______
```

2. Même question avec le minimum.

3. On considère la fonction $f : x \longmapsto \arctan\left(x^3 - x - \frac{1}{2}\right)$. *On pourra par exemple utiliser la fonction `np.arctan` présente dans le module `numpy`.*

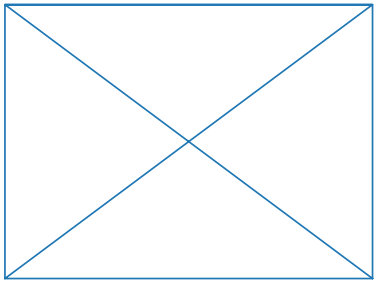
3.1) Tracer cette fonction sur l'intervalle $\left[-\frac{1}{2}, 2\right]$. Que pouvez-vous conjecturer sur l'existence d'un minimum ou maximum global ?

3.2) Déterminer numériquement ces valeurs à l'aide des questions précédentes. *Pour la liste `L`, vous ferez varier le nombre de points.*

3 SOLUTIONS DES EXERCICES

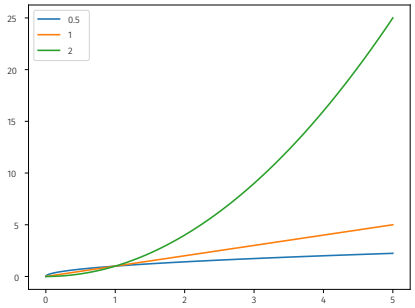
Solution (exercice 1) [Énoncé] On doit relier les points suivants $(0,0),(1,0),(1,1),(0,1),(0,0),(1,1),(0,1),(1,0)$. D'où la commande suivante :

```
plt.plot([0, 1, 1, 0, 0, 1, 0, 1], [0, 0, 1, 1, 0, 1, 1, 0])
plt.axis("off")
```



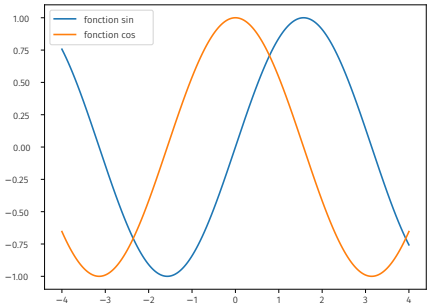
Solution (exercice 2) [Énoncé] On peut par exemple faire cela au moyen d'une boucle `for`.

```
L_pui = [0.5, 1, 2]
X = np.linspace(0, 5, 10**3)
for alpha in L_pui:
    Y = [x**alpha for x in X]
    plt.plot(X, Y, label=str(alpha))
    plt.legend()
```

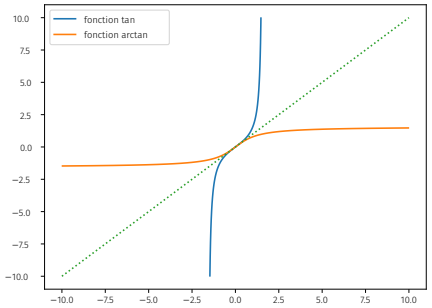


Solution (exercice 3) [Énoncé]

```
def trace_trigo(a, b):
    X = np.linspace(a, b, 10**3)
    Y = [np.sin(x) for x in X]
    Z = [np.cos(x) for x in X]
    plt.plot(X, Y, label="fonction sin")
    plt.plot(X, Z, label="fonction cos")
    plt.legend()
trace_trigo(-4, 4)
```




```
def trace_tanatan():
    X = np.linspace(-np.pi/2+0.1, np.pi/2-0.
0**3)
    Y = [np.tan(x) for x in X]
    plt.plot(X, Y, label="fonction tan")
    plt.plot(Y, X, label="fonction arctan")
    plt.plot([-10, 10], [-10, 10], ':') # re
    ↪ deux points
    plt.legend()
trace_tanatan()
```

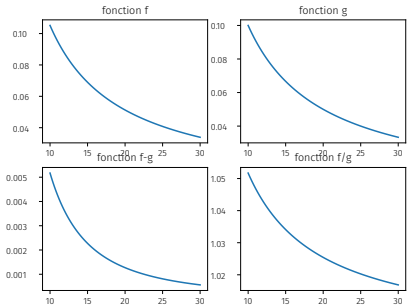


Solution (exercice 4) [Énoncé]

1.
```
def f(x):
    return ma.exp(1/x)-1
def g(x):
    return 1/x
```

2.
```
def trace_grille():
    X = np.linspace(10, 30, 10**3)
    plt.subplot(2, 2, 1)
    Y = [f(x) for x in X]
    plt.title("fonction f")
    plt.plot(X, Y)
    plt.subplot(2, 2, 2)
    Y = [g(x) for x in X]
    plt.title("fonction g")
    plt.plot(X, Y)
    plt.subplot(2, 2, 3)
    Y = [f(x)-g(x) for x in X]
    plt.title("fonction f-g")
    plt.plot(X, Y)
    plt.subplot(2, 2, 4)
    Y = [f(x)/g(x) for x in X]
```

```
    plt.title("fonction f/g")
    plt.plot(X, Y)
trace_grille()
```



On peut conjecturer les valeurs des limites $\lim\limits_{x\to\infty} f(x) = \lim\limits_{x\to\infty} g(x) = 0$, et même :

$$\lim_{x\to\infty} (f(x)-g(x)) = 0, \quad \textbf{et} \quad \lim_{x\to\infty} \frac{f(x)}{g(x)} = 1.$$

3. On souhaite calculer : $\lim\limits_{x\to\infty} \frac{e^{1/x}-1}{1/x} = \lim\limits_{x\to\infty} x\left(e^{1/x}-1\right)$ qui est une forme indéterminée.
En effectuant le changement de variable $y = 1/x$, on déduit que, en cas d'existence :

$$\lim_{x\to\infty} \frac{e^{1/x}-1}{1/x} = \lim_{y\to\infty} \frac{e^{y}-1}{y} = \exp'(0) = \boxed{1}$$

en reconnaissant un taux d'accroissement.

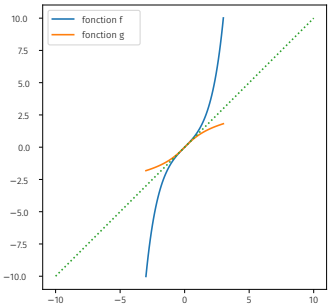
Solution (exercice 5) [Énoncé]

1.
```
def trace_fg():
    X = np.linspace(-3, 3, 10**3)

    def f(x):
        return (ma.exp(x)-ma.exp(-x))/2

    def g(x):
        return ma.log(x+ma.sqrt(x**2+1))

    Y = [f(x) for x in X]
    Z = [g(x) for x in X]
    plt.plot(X, Y, label="fonction f")
    plt.plot(X, Z, label="fonction g")
    plt.plot([-10, 10], [-10, 10], ':')
    plt.legend()
    plt.axis("scaled")
trace_fg()
```



Les graphes semblent être symétriques par rapport à $y = x$, donc les fonctions semblent être réciproques l'une de l'autre.

2.
```
>>> X = np.linspace(-2, 2, 10)
>>> L_fg = [round(f(g(x))-x, 2) for x in X]
>>> L_gf = [round(g(f(x))-x, 2) for x in X]
>>> L_fg
[np.float64(1.14), np.float64(0.77), np.float64(0.44), \
↪ np.float64(0.18), np.float64(0.02), np.float64(0.03), \
↪ np.float64(0.28), np.float64(0.93), np.float64(2.18), \
↪ np.float64(4.39)]
>>> L_gf
[np.float64(-0.54), np.float64(-0.55), np.float64(-0.57), \
↪ np.float64(-0.62), np.float64(-0.79), np.float64(-0.21), \
↪ np.float64(-0.38), np.float64(-0.43), np.float64(-0.45), \
↪ np.float64(-0.46)]
```

On retrouve la conjecture faite dans la précédente question, à savoir que $f \circ g \approx \mathrm{Id}$, $g \circ f \approx \mathrm{Id}$, et donc que f, g sont réciproques l'une de l'autre.

3. Prouvons-le mathématiquement. Les deux fonctions sont bien définies sur $\mathbb{R}$.

- Montrons que : $\forall x \in \mathbb{R}, \quad f(g(x)) = x$. Soit $x \in \mathbb{R}$. Alors :

$$f(g(x)) = \frac{x+\sqrt{x^2+1}-\frac{1}{x+\sqrt{x^2+1}}}{2} = \frac{\left(x+\sqrt{x^2+1}\right)^2-1}{2\left(x+\sqrt{x^2+1}\right)}$$

$$= \frac{x^2 + (x^2+1) + 2x\sqrt{x^2+1} - 1}{2\left(x+\sqrt{x^2+1}\right)}$$
$$= x \times \frac{2(x+\sqrt{x^2+1})}{2(x+\sqrt{x^2+1})} = x.$$

- Montrons que : $\forall x \in \mathbb{R},\quad g(f(x)) = x$. Soit $x \in \mathbb{R}$. Alors :

$$g(f(x)) = \ln\left(\frac{e^x - e^{-x}}{2} + \sqrt{\left(\frac{e^x - e^{-x}}{2}\right)^2 + 1}\right)$$
$$= \ln\left(\frac{e^x - e^{-x}}{2} + \sqrt{\frac{e^{2x} + e^{-2x} - 2 + 4}{4}}\right) \quad \text{développement du carré et réduction au même dénominateur}$$
$$= \ln\left(\frac{e^x - e^{-x}}{2} + \sqrt{\frac{(e^x + e^{-x})^2}{4}}\right)$$
$$= \ln\left(\frac{e^x - e^{-x}}{2} + \frac{e^x + e^{-x}}{2}\right) \quad e^x + e^{-x} > 0$$
$$= \ln(e^x) = x.$$

On a bien montré que f, g sont réciproques l'une de l'autre. On obtient au passage la bijectivité de f.

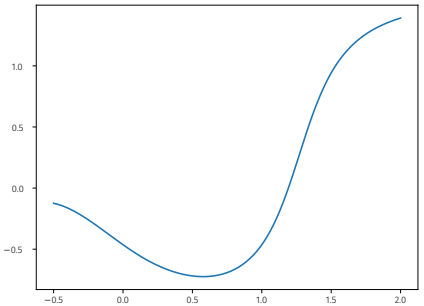
Solution (exercice 6) [Énoncé]

```
def maximum_sub(L, f):
    ind_max = 0
    max = f(L[0])
    for i in range(1, len(L)):
        if f(L[i]) > max:
            ind_max = i
            max = f(L[i])
    return max, L[ind_max]

def minimum_sub(L, f):
    ind_mini = 0
    mini = f(L[0])
    for i in range(1, len(L)):
        if f(L[i]) < mini:
            ind_mini = i
            mini = f(L[i])
    return mini, L[ind_mini]
```

```
def f(x):
    return ma.atan(x**3-x-0.5)

X = np.linspace(-0.5, 2, 10**3)
Y = [f(x) for x in X]
plt.plot(X, Y)
```



On constate l'existence d'un minimum global, et d'un maximum global. Testons à présent sur plusieurs subdivisions les fonctions précédentes.

```
>>> L = np.linspace(-0.5, 2, 4)
>>> maximum_sub(L, f)
(1.3909428270024184, np.float64(2.0))
>>> minimum_sub(L, f) # pas terrible pour le min
(-0.6724785026448458, np.float64(0.33333333333333337))
>>> L = np.linspace(-0.5, 2, 100)
>>> maximum_sub(L, f)
(1.3909428270024184, np.float64(2.0))
>>> minimum_sub(L, f) # c'est mieux !
(-0.7243390760778351, np.float64(0.5858585858585859))
```

Il est logique que le maximum fonctionne bien puisqu'il est situé au bord de l'intervalle, donc appartient au `linspace`.

On peut proposer une autre version de `minimum_sub` en exploitant la propriété suivante : x_0 est un minimum de f si, et seulement si, x_0 est un maximum de $-f$. Nous déduisons alors la version ci-après.

```
def minimum_sub(L, f):
    def g(x):
        return -f(x)
    mini, x_mini = maximum_sub(L, g)
    return -mini, x_mini
```