
MPSI -ITC - TD 11 - 24 NOVEMBRE 2025

DOCUMENTATIONS ET TRACÉ DE COURBES

Vous disposez d'un memento numpy et matplotlib en fin d'énoncé.

Exercice 1 (Documentation de modules et de fonctions)

Si vous avez importé un module (par exemple `import math`) vous pouvez taper dans une console python `help(nom du module)` pour accéder à sa documentation. De même, dans une console `help(nom de la fonction)` permet d'accéder à la documentation d'une fonction. Le module `math` contient essentiellement les fonctions mathématiques usuelles pour le type `float`. En regardant les documentations pertinentes, répondre aux questions suivantes :

1. Parmi les fonctions `floor` et `ceil`, laquelle correspond à un arrondi au supérieur et laquelle à un arrondi à l'inférieur ?
2. Quelle fonction du module `math` convertit en radians un angle exprimé en degrés ? Quelle est la fonction réciproque ? Vérifiez que vous savez les coder vous-mêmes.
3. Dans la section **DATA** de la documentation du module `math`, quelles sont les deux données dont il est bon de connaître l'existence ? Sinon, comment les calcule-t-on ?

Remarque 1 : il n'existe pas qu'une seule manière d'associer un entier à un réel ! Je vous conseille de **ne pas** utiliser la fonction `int` de la librairie standard pour convertir un flottant en entier, car son nom n'est pas explicite sur quelle méthode de conversion est utilisée. Utilisez plutôt, selon l'opération que vous voulez faire, les fonctions `floor`, `ceil` ou `trunc` (du module `math`, à importer, donc) ou `round` (de la librairie standard, ne nécessite pas d'import).

Remarque 2 : le module `cmath` est l'équivalent du module `math` pour les nombres complexes.

Exercice 2 (Module numpy)

numpy est l'abréviation de Numerical Python. C'est un module très utilisé pour le calcul scientifique.

Alors qu'une liste python est de taille variable et peut contenir des données de nature différentes, numpy propose le type `ndarray` (pour *numerical data array*) pour les tableaux homogènes de taille fixe, plus efficaces à l'usage. Ces tableaux sont créés à l'aide de fonctions du module numpy : `array`, `zeros`, `empty`, `linspace`, `arange`, `full`, etc. Le type des éléments peut être déclaré à la création par l'argument optionnel `dtype` (*data type*). Par défaut les tableaux contiennent des flottants (`numpy.float64` ou `float` ou `numpy.double`).

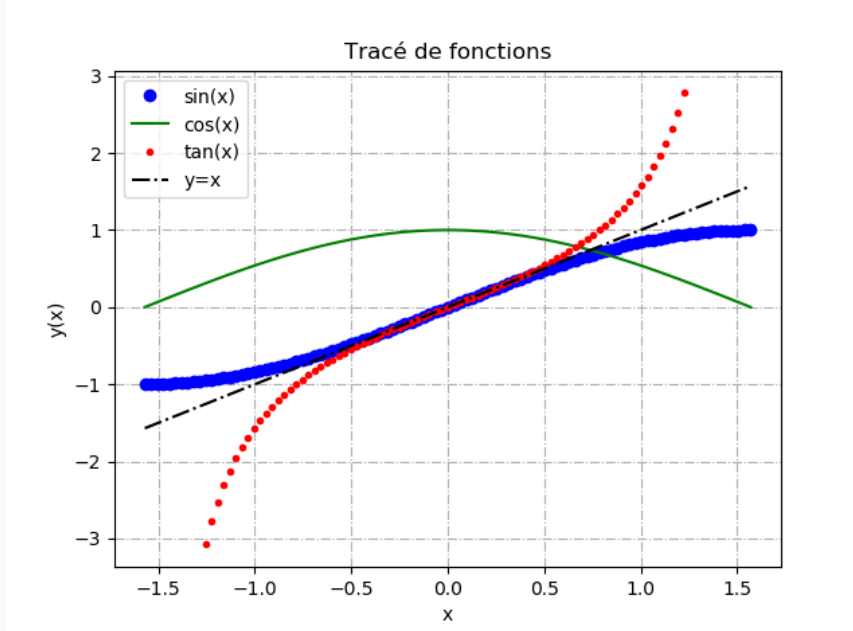
```
import numpy as np
import matplotlib.pyplot as plt
pi = np.pi
x = np.linspace(-pi/2, pi/2, 30)
y = np.sin(x)
plt.title('Tracé de la fonction sinus avec Numpy')
plt.plot(x,y,'bo',label='np.sin(x)')
plt.legend()
plt.show()
```

1. Exécuter ce code. Afficher `x`. À quoi sert la fonction `np.linspace` ?
2. Quel est le type de `x` ? Comment ferait-on si on voulait le convertir en une liste python ?
3. Que prend comme argument la fonction `np.sin` ? Quelle différence avec la fonction `math.sin` ? Comment traduire les termes *element-wise* et *ufunc* dans ce contexte ?

Exercice 3 (Module matplotlib.pyplot : tracé de courbes)

On importe traditionnellement ce module par `import matplotlib.pyplot as plt`. Tous les éléments de ce module ont donc pour préfixe `plt`.

On souhaite tracer les courbes représentatives des fonctions $\sin$, $\cos$, $\tan$ sur $[-\pi/2, \pi/2]$ en utilisant des listes python et 100 points d'échantillonnage régulièrement répartis.



1. À l'aide du mémento fourni, reproduire la figure précédente.
2. Conservez un fichier `template_matplotlib.py` contenant la partie du code qui vous a permis de tracer la figure.

Exercice 4 (Représentation de fonctions)

1. Tracer sur le même graphe et sur l'intervalle $[-1, 1]$, avec des couleurs différentes, les courbes d'équations $y = x^n$ pour $n \in \{2; 3; 4; 5; 10\}$.
Expliquer pourquoi les courbes sont plus ou moins plates autour de zéro, vérifier la parité des fonctions.
2. Tracer simultanément sur l'intervalle $[-1, 3]$ les courbes d'équations $y = 1 + 2x$ et $y = 1 + 2x + (x - 1)^n$ pour $n \in \{2; 3; 4; 5\}$.
Donner la valeur en $x = 0$ de ces fonctions et vérifiez-la sur les courbes.
Observer que toutes les courbes passent par un même point pour $n > 1$, donner ses coordonnées.
Observer que toutes les courbes ont la même tangente en $x = 1$.

Mémento Matplotlib

<code>import matplotlib.pyplot as plt</code>	Importe le module, préfixe <code>plt</code> .
<code>plt.plot(X, Y, 'ro', label='')</code>	X et Y sont des tableaux de même taille contenant la liste des coordonnées x et y . 'ro' → red, symbole o 'b+' → blue, symbole +
<code>plt.legend()</code>	affiche les différents label définis via <code>plt.plot</code>
<code>plt.arrow(X,Y,Vx,Vy,head_width=0.2, head_length=0.5,color="red")</code>	Trace le vecteur de coordonnées (v_x, v_y) à partir du point de coordonnées (x, y) avec les mêmes conventions que <code>plt.plot</code> .
<code>plt.axis('equal')</code> ou si bug, <code>plt.gca().set_aspect('equal')</code>	Permet d'avoir la même représentation de l'unité en x et en y
<code>plt.xlim(xmin, xmax)</code> <code>plt.ylim(ymin, ymax)</code>	Fixe les limites souhaitées en x et en y
<code>plt.text(x,y,"texte", fontsize=12)</code>	Ecrit <code>texte</code> au point de coordonnées (x, y)
<code>plt.xlabel("x (unité)")</code> <code>plt.ylabel("y (unité)")</code>	Pour écrire les grandeurs sur les axes
<code>plt.grid(linestyle='-.')</code>	Trace la grille en arrière plan
<code>plt.savefig('image.png')</code>	Sauvegarde le graphique dans un fichier png
<code>plt.show()</code>	Pour afficher la figure. A mettre à la fin du programme.
<code>plt.figure()</code>	Pour avoir plusieurs fenêtres de graphique, à mettre au début de chaque bloc correspondant à une fenêtre

Mémento Numpy

Les fonctions fournies par le module sont **vectérielles** !

<code>import numpy as np</code>	Importe le module, préfixe <code>np</code> .
<code>np.linspace(start, stop, n)</code>	renvoie un tableau de n valeurs régulièrement réparties sur l'intervalle <code>[start,stop]</code>
<code>np.arange(start, stop, step)</code>	renvoie un tableau de valeurs comprises en start inclu et stop exclu par pas de step
<code>np.zeros(N)</code>	renvoie un tableau de N zéros
<code>np.empty(N)</code>	renvoie un tableau de N cases non initialisées

Soit `x` un tableau.

Tracé de la fonction <code>f</code> Méthode 1 – fonction <code>f</code> vectorielle	<code>plt.plot(x, f(x))</code>
Tracé de la fonction <code>f</code> Méthode 2 – fonction <code>f</code> non vectorielle	<code>y = [f(u) for u in x]</code> <code>plt.plot(x, y)</code>
Tracé de la fonction <code>f</code> Méthode 3 – fonction <code>f</code> non vectorielle	<code>y = list(map(f, x))</code> <code>plt.plot(x, y)</code>
Tracé de la fonction <code>f</code> Méthode 4 – fonction <code>f</code> non vectorielle	<code>g = np.vectorize(f)</code> <code>plt.plot(x, g(x))</code>