exos numpy

May 7, 2025

1 Exercices

L'objectif de ces exercices est de manipuler des tableaux. Il est interdit d'utiliser des boucles for.

1.1 Création d'un tableau

Créer un tableau NumPy contenant les éléments suivants : [1, 2, 3, 4, 5]

- Afficher la taille du tableau.
- Afficher le type de données.
- Créer le même tableau mais avec des nombres complexes

[6]:
```
tableau = np.array([1, 2, 3, 4, 5])
tableau = np.arange(1, 6)
print(tableau)

len(tableau)
tableau.dtype

tableau = np.array([1, 2, 3, 4, 5], dtype=complex)
tableau
```

```
[1 2 3 4 5]
```

[6]:
```
array([1.+0.j, 2.+0.j, 3.+0.j, 4.+0.j, 5.+0.j])
```

1.2 Indexation

On considère le tableau suivant :

```
arr = np.array([[10, 20, 30], [40, 50, 60]])
```

Afficher :

- La première ligne
- La deuxième colonne
- L'élément 50

[4]:
```
arr = np.array([[10, 20, 30], [40, 50, 60]])
```

```
print(arr)

arr[0, :]
```

```
[[10 20 30]
 [40 50 60]]
```

[4]:
```
array([10, 20, 30])
```

[7]:
```
arr[: 1]
```

[7]:
```
array([[10, 20, 30]])
```

[9]:
```
arr[1, 2]
```

[9]:
```
60
```

1.3 Filtres et masques

On considère le tableau suivant :

```
arr = np.array([10, 25, 30, 5, 90])
```

- Créer un nouveau tableau ne contenant que les éléments supérieurs à 20.

[11]:
```
arr = np.array([10, 25, 30, 5, 90])
arr[arr>20]
```

[11]:
```
array([25, 30, 90])
```

1.4 Moyenne, somme, min, max

Avec le tableau suivant :

```
arr = np.array([5, 10, 15, 20])
```

Calculer la moyenne, la somme des éléments, le minimum et la maximum

[14]:
```
arr = np.array([5, 10, 15, 20])

print(arr.mean())
print(arr.sum())
print(np.sum(arr))
print(arr.min())
print(arr.max())
```

```
12.5
50
50
```

```
5
20
```

- Calculer la somme des entiers de 1 à n, pour n=351 en une ligne.
- Calculer la somme de la racive carré des entiers de 1 à n, pour n=351 en une ligne.

[17]:
```
n = 351
print(np.arange(1, n+1).sum())
print( np.sum(np.arange(1, n+1)) )
print(n*(n+1)/2)
```

```
61776
61776
61776.0
```

[21]:
```
n = 351
np.sum(np.sqrt(np.arange(1, n+1)))
```

[21]:
```
4393.150429650206
```

Calculer l'intégrale de e^{-x^2} entre 0 et 1 par la méthode des rectangles. On prendra $N = 1000$ rectangles.

- Créer un tableau $x_i = il/N$ (où $l = 1$ dans notre example)
- Calculer le tableau $y_i = e^{-x_i^2}$. Tracer la fonction sur un graph.
- Calculer $\sum e^{-x_i^2} \frac{l}{N}$

Créer une fonction qui renvoie l'intégrale entre a et b de f

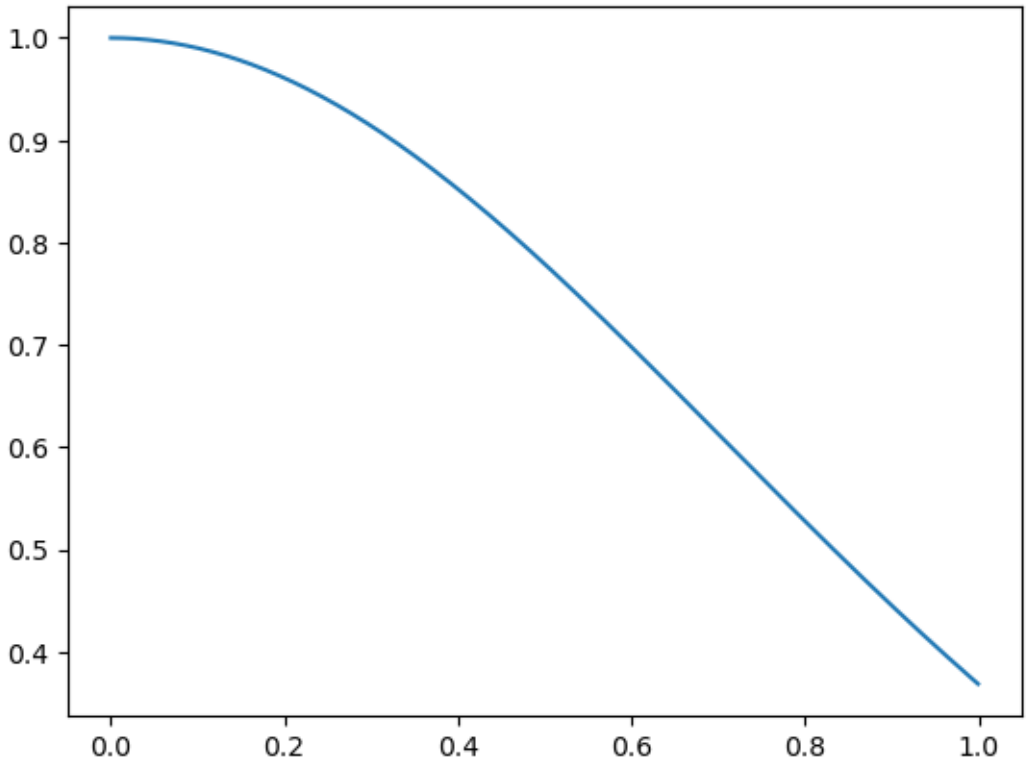
[26]:
```
#[0, 0.001, 0.002, 0.003, ....] = [0, 1/1000, 2/1000, 3/1000, ... ]
#                               = [0, 1, 2, 3, ...]/1000

N = 1000
x = np.arange(N)/N

y = np.exp(-x**2)

import matplotlib.pyplot as plt
plt.plot(x, y)
```

[26]:
```
[<matplotlib.lines.Line2D at 0x7efadbe64a90>]
```



[27]:
```
x = np.arange(N)/N
y = np.exp(-x**2)

np.sum(y)/N
```

[27]: 0.7471401317785991

[29]:
```
def f(x):
    return np.exp(-x**2)

a = 0
b = 1
N = 1000

x = a + (b-a)/N * np.arange(N)
y = f(x)
intregrale = np.sum(y)*(b-a)/N
intregrale
```

[29]: 0.7471401317785991

[33]:
```
def calcule_integrale(f, a, b, N):
    x = a + (b-a)/N * np.arange(N)
    y = f(x)
    integrale = np.sum(y)*(b-a)/N
    return integrale

calcule_integrale(np.sin, 0, 1, 1000)
```

[33]: 0.4592769203313145

[35]:
```
1-np.cos(1)
```

[35]: 0.45969769413186023

2 Jour le plus chaud

Le fichier temperature.dat (https://python.sdv.u-paris.fr/data-files/temperatures.dat) contient un relevé de quatre températures pour chaque jour de la semaine

À l'aide du module NumPy, on souhaite déterminer quel est le jour de la semaine le plus chaud. Pour cela nous vous proposons les étapes suivantes : 1. Récupérez le nom des jours de la semaine depuis le fichier et stockez-les dans une liste days. 2. Récupérez les valeurs de températures depuis le fichier et stockez-les dans un array 2D. La fonction np.loadtxt et son argument usecols vous seront utiles. 3. Parcourez chaque ligne de la matrice, calculez la température moyenne de chaque jour puis stockez-la dans une liste mean_temps. 4. À l'aide des deux listes days et mean_temps, déterminez et affichez le jour le plus chaud.

3 Calcul de la moyenne sur un tableau 2D

Un tableau numpy contient les notes d'une classe. Chaque ligne correspond à un élève et chaque colone à un examen.

[36]:
```
import numpy as np

N_eleves = 35
N_examens = 3

notes = np.random.rand(N_eleves, N_examens)*20
```

- Calculer la moyenne de chaque élève. Calculer la moyenne pour chaque examen.
- Calculer la moyenne de chaque élève, sachant que les coeficients pour les 3 examens sont de 1, 3 et 2.

[43]:
```
notes
```

[43]:
```
array([[ 7.99015337,  2.20315651, 18.04428873],
       [ 3.2943243 , 12.59925317,  3.30108986],
       [ 9.26636459, 11.46862254, 11.38151766],
       [18.89072141, 18.77973399,  7.12169985],
       [ 9.7458631 ,  1.30277296, 10.68349112],
       [ 6.47814944,  4.24446042, 16.36383407],
       [ 3.60667636,  5.58110786, 19.99438447],
       [18.70456045, 12.48399427, 11.890586  ],
       [11.10359637, 12.35655439, 18.40763719],
       [ 9.36659175, 16.03248715, 15.68847713],
       [15.24391312, 13.37605111,  8.64255029],
       [ 6.16716241,  9.6229677 , 19.63382531],
       [15.47117612, 17.42214665, 12.08923687],
       [ 9.42473395,  4.61966542, 10.90174641],
       [12.31205845,  6.75882151,  0.34921549],
       [ 0.05255151,  2.82937221, 13.61612886],
       [10.62373868, 11.06311746, 10.45203411],
       [19.4467789 ,  3.89544222,  4.6892614 ],
       [ 8.44651276, 18.42214673,  8.86803652],
       [ 4.21522236,  4.16242401, 13.08812124],
       [12.04638057, 13.28833072,  7.39465953],
       [13.3876088 ,  7.27315261,  5.59634972],
       [15.28994454, 11.52309069, 10.05253042],
       [ 1.4756528 , 15.64303529, 19.66629499],
       [17.50826647, 12.96204121,  9.89573345],
       [ 8.08198834,  5.60485511,  3.07941442],
       [ 7.8336043 , 17.52265789,  4.85826814],
       [ 0.28830997, 16.84314605,  8.77190949],
       [18.26867082, 16.4620001 ,  1.28502227],
       [ 4.57890965,  3.47629458, 15.02382133],
       [ 7.61963525,  2.79339917,  1.35687629],
       [ 5.67023818, 19.56255359,  9.28476261],
       [ 5.72204532,  2.80687199,  1.97935884],
       [ 5.35504649,  2.49789067,  7.83212809],
       [18.7751067 ,  8.82216731,  7.74645806]])
```

[41]:
```
notes.mean(axis=0)

notes.mean(axis=1)
```

[41]:
```
array([ 9.41253287,  6.39822244, 10.7055016 , 14.93071842,  7.24404239,
        9.02881464,  9.72738956, 14.35971357, 13.95592931, 13.69585201,
       12.42083817, 11.80798514, 14.99418654,  8.31538193,  6.47336515,
        5.49935086, 10.71296342,  9.34382751, 11.912232  ,  7.15525587,
       10.90979027,  8.75237038, 12.28852189, 12.26166102, 13.45534704,
        5.58875262, 10.07151011,  8.63445517, 12.00523106,  7.69300852,
        3.92330357, 11.50585146,  3.50275872,  5.22835508, 11.78124403])
```

[45]:
```
(notes[:,0]*1 + notes[:,1]*3 + notes[:,2]*2)/6
```

[45]:
```
array([ 8.44803339,  7.94904392, 11.07254459, 14.91222051,  5.8368607 ,
        8.65653314, 10.05646148, 13.32295254, 14.16475565, 14.80683458,
       12.10952784, 12.38395269, 15.3193483 ,  7.51453717,  5.54782566,
        5.96215431, 10.78619321,  6.75193806, 13.57483767,  7.14645615,
       11.11678196,  7.73329434, 11.66071291, 14.62289144, 12.69764283,
        5.17589708, 11.68635238, 11.39359452, 11.70411928,  7.50923934,
        3.11893089, 13.82123736,  3.0168965 ,  4.75216245, 10.1224208 ])
```

[48]:
```
coeffs = np.array([1, 3, 2])

np.sum(notes[0,:]*coeffs)/np.sum(coeffs)
```

[48]:
```
8.448033393112974
```

4 Tracer une fonction mathématique

Objectif : tracer la courbe de la fonction suivante :

$$y = \sin(x) + 0.5 \cdot \cos(2x)$$

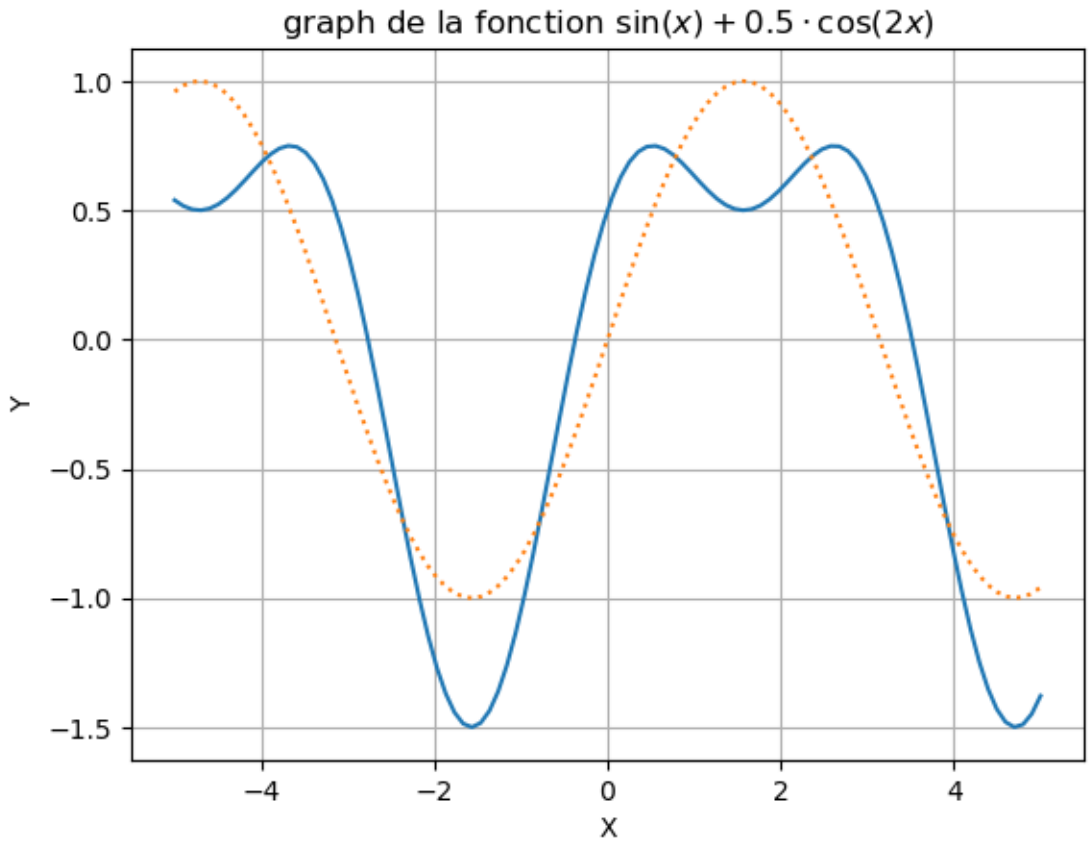
1. Créer l'axe des x
   Utilise `np.linspace` pour générer 100 points entre -5 et 5.
2. Calculer les valeurs de y avec numpy
3. Tracer la courbe
4. Ajouter un titre au graphique.
5. Ajouter les noms des axes x et y.
6. Tracer aussi $\sin(x)$ sur le même graphique, avec un style différent (par exemple en pointillés).
7. Ajouter une grille pour améliorer la lisibilité.

[56]:
```
x = np.linspace(-5, 5, 100)
y = np.sin(x) + 0.5*np.cos(2*x)

plt.plot(x, y)
plt.title(r'graph de la fonction $ \sin(x) + 0.5 \cdot \cos(2x)$')
plt.xlabel('X')
plt.ylabel('Y')

y2 = np.sin(x)
plt.plot(x, y2, ':')

plt.grid()
```



5 PIB des pays

Nous avons téléchargé des fichiers contenant le PIB en fonction de l'année pour un pays donné. Ces fichiers sont en json. Pour les lire, on utilisera le code suivant :

[ ]:
```
import json

with open('data_FR.json') as f:
    data = json.load(f)
```

1. Afficher la variable `data`. Quel est son type de donnée ? Que contient `data[1]` ?
2. A l'aide d'une boucle for, extraire deux liste, l'une contenant l'année et l'autre le PIB
3. Tracer un graphique représentant l'évolution du PIB de la France.
4. Regrouper dans une fonction le code pour extraire les donnée à partir du nom du fichier.
5. Tracer sur un même graphique l'évolution du PIB de la France, l'Allemagne et les États-Unis (on utilisera l'option `label` et la fonction `legend`)

[ ]:
```

```