4 Introduction à numpy

Exercice 31. La classe `numpy.array` correspond à des tableaux (multidimensionnels) d'éléments de même type.

1. Le constructeur prend une collection en argument. Si l'argument est une collection de collections de même taille, le tableau sera de dimension 2. Si l'argument est une collection de collections de même taille de collections de même taille, le tableau sera de dimension 3...

   Deux attributs importants sont `shape` qui donnent un tuple correspondant à la taille de chaque dimension (appelée axe), et `dtype` qui le donne le type des éléments du tableau.

   ```
   >>> import numpy as np
   >>> a = np.array([1, 2, 3])
   >>> a
   >>> a.shape
   >>> a.dtype
   >>> b = np.array((1.0,2,3))
   >>> b
   >>> b.dtype
   >>> c = np.array([[1,2],[3,4,5]])
   >>> c
   >>> c.shape
   >>> d = np.array([[1,2],[3,4]])
   >>> d
   >>> d.shape
   >>> d = np.array([[1,2],(3,4)])
   >>> d
   >>> d[0,1] = 7
   >>> d
   ```

2. Pour créer des tableaux, la fonction `arange` est l'équivalent de `range` mais renvoie un `numpy.array`. La fonction `linspace` permet de créer un tableau de nombres régulièrement espacés dans un intervalle.

   ```
   >>> np.arange(0,10,2)
   >>> np.arange(0,10,3)
   >>> np.arange(0,9,3)
   >>> np.linspace(1,6,10)
   ```

   D'autres exemples de fonctions pour créer des tableaux : `zeros, ones, eye, diag`.

3. Les fonctions et opérateurs mathématiques sont appliquées élément par élément.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0,10,100)
print(np.sin(x)/ x)
plt.plot(x, np.sin(x)/ x)
plt.show()
```

Exercice 32. Slicing
Pour extraire des sous-tableaux, il existe une syntaxe similaire à celle pour les listes.

```
A = np.array([1, 2, 3, 4, 5])
print(A)
print(A[::-1])
print(A[::2])
print(A[:-1])

A = np.array([[n + m * 10 for n in range(5)] for m in range(5)])
A
print(A[1:4])
print(A[1:4, :])
A[1]
A[:, 1]
A[:, ::-1]

row_indices = [1, 2, 4]
print(A[row_indices])
A[:, row_indices]

B = np.arange(5)
row_mask = np.array([True, False, True, False, False])
print(B)
print(B[row_mask])
```

Exercice 33. Copies et vues

1. Different array objects can share the same data. The view method creates a new array object that looks at the same data.

   ```
   a = np.arange(12)
   a.shape = 3,4
   c = a.view()
   c is a
   c.base is a                        # c is a view of the data owned by a
   ```

```
c.flags.owndata

c.shape = 2,6                       # a's shape doesn't change
a.shape

c[0,4] = 1234                       # a's data changes
a
```

2. Slicing an array returns a view of it :

```
s = a[ : , 1:3]
s[:] = 10   # s[:] is a view of s.
            # Note the difference between s=10 and s[:]=10
a
```

3. The copy method makes a complete copy of the array and its data.

```
d = a.copy()          # a new array object with new data is created
d is a

d.base is a           # d doesn't share anything with a

d[0,0] = 9999
a
```

Exercice 34. Broadcasting

1. Si a un est tableau de nombres de forme (n_1, n_2, n_3) et b est un tableau de nombres de forme (m_1, m_2, m_3) tels que $n_i = m_i$ ou $n_i = 1$ pour $1 \leq i \leq 3$, alors $c = a * b$ est le tableau de forme $(n_1 \vee m_1, n_2 \vee m_2, n_3 \vee m_3)$ où

$$c[i, j, k] = a[i \wedge n_1, j \wedge n_2, k \wedge n_3] * b[i, j, k]$$

Cette même règle s'applique pour tout opérateur binaire (pour toute fonction de deux arguments) et pour un nombre d'axes quelconque.
Exemple :

```
a = np.arange(24).reshape(2,3,4)
a
b = np.arange(8).reshape(2,1,4)
b
a + b
c = np.arange(4)
a + c            # tout se passe comme si c est de forme (1,1,4)

d = c[np.newaxis, np.newaxis,:]
d.shape
a + d
```

2. Exemple

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0,100)[:, np.newaxis]
y = np.arange(0,100)[np.newaxis, :]
tpg = np.gcd(x, y) == 1
print(x.shape, y.shape, tpg.shape)
plt.imshow(tpg, cmap=plt.cm.gray)
plt.show()
```

Exercice 35. Créer le tableau (sans l'écrire littéralement)

```
[[1,  6, 11],
 [2,  7, 12],
 [3,  8, 13],
 [4,  9, 14],
 [5, 10, 15]]
```

Exercice 36. Diviser chaque colonne du tableau

`a = np.arange(25).reshape(5, 5)`

avec le tableau `b = np.array([1., 5, 10, 15, 20])` (élément par élément).
Utiliser `np.newaxis`.

Exercice 37. Créer un tableau 10 x 3 de nombres aléatoires dans l'intervalle [0,1]. Pour chaque ligne, donner la valeur la plus proche de 0.5.
Utiliser `abs` et `argsort`.

Exercice 38. Etant donné un marché financier constitué d'une action de cours $(S_t)_{t\geq 0}$ et d'une épargne de taux d'intérêt mathématique ρ, une option d'achat (*call* en anglais) de prix d'exercice K et d'échéance T est un contrat conclu à l'instant 0, au terme duquel son acquéreur détient le droit, mais non l'obligation, d'acheter une action à l'instant T et au prix K préalablement choisis, et ce quel que soit le prix de l'actif S_T à l'instant T. Dans le modèle de Black-Scholes, on peut établir que le prix C de l'option d'achat est donné par

$$C = \mathbb{E}\left(S_0 \exp\left(-\sigma\sqrt{T}N - \frac{1}{2}\sigma^2 T\right) - K\exp(-\rho T)\right)^+$$

N suivant une loi $\mathcal{N}(0,1)$.

1. Par la méthode de Monte-Carlo, donner une approximation de C en fonction de σ, T, K, ρ et S_0, ainsi que l'intervalle de confiance asymptotique au niveau de confiance 0.95.
   On prendra
   $$S_0 = 15, \quad K = 20, \quad \sigma = 0.5, \quad \rho = \log(1.03)$$
   et T dans l'intervalle $[0.5, 20]$.
   Utiliser la fonction `np.maximum` pour calculer x^+.

2. Représenter graphiquement les résultats (avec `matplotlib.pyplot.errorbar`).
3. Le prix C peut se calculer de la façon suivante. Étant données les fonctions :

$$\forall x \in \mathbb{R},\ \Phi(x) = (2\pi)^{-1/2} \int_{-\infty}^{x} \exp(-t^2/2)dt,$$
$$d_1(x) = T^{-1/2}\sigma^{-1}\big[\ln(K^{-1}x) + (\rho + \sigma^2/2)T\big]$$
$$d_2(x) = d_1(x) - \sigma T^{1/2}$$

alors,

$$C = S_0\Phi\big(d_1(S_0)\big) - K\exp(-\rho T)\Phi(d_2(S_0)),$$

où Φ désigne la fonction de répartition de la loi normale centrée réduite.

Comparer la valeur donnée par cette formule et celle obtenue par Monte-Carlo.
Utiliser
`from scipy.stats import norm`
et la fonction `norm.cdf` pour Φ.