L1 — Calcul Scientifique

Frédéric Jurie (frederic.jurie@unicaen.fr)
Université de Caen Normandie
Année universitaire 2019-2020

La librairie numpy

Objectifs de ce cours

- Présentation de la librairie Python numpy
- wikipedia:

  “NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

  Plus précisément, cette bibliothèque logicielle open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

  NumPy est la base de SciPy, regroupement de bibliothèques Python autour du calcul scientifique.”

Introduction

- Il s'agit d'une librairie permettant de réaliser des opérations sur des tableaux de nombres
- Représentés sous forme de tableau à 1 dimension (vecteur), 2 dimensions (matrices) ou plus.
- Permet de faire globalement des traitements sur l'ensemble des valeurs sans avoir à faire de boucles
- Les opérations sont exécutées rapidement, la librairie étant écrite en langage C, même si elle est utilisée à partir de programmes en Python
- Librairie de base pour d'autres librairies : optimisation, calcul symbolique, etc.
- Installation :
  - `pip3 install numpy`
  - `http://www.numpy.org`

L’objet array

- L’objet array est au coeur de numpy
- C’est lui qui permet de représenter des tableaux multidimensionnels
- Exemple de création à partir de listes Python :
`data = np.array([[1, 2], [3, 4], [5, 6]])`

Attribute	Description
shape	A tuple that contains the number of elements (i.e., the length) for each dimension (axis) of the array.
size	The total number of elements in the array.
ndim	Number of dimensions (axes).
nbytes	Number of bytes used to store the data.
dtype	The data type of the elements in the array.

L’objet array (suite)

- Les données du tableau peuvent être de type différent :
`data = np.array([[1, 2], [3, 4], [5, 6]], dtype=np.int)`

Attribute	Variants	Description
int	int8, int16, int32, int64	Integers.
uint	uint8, uint16, uint32, uint64	Unsigned (non-negative) integers. Boolean (True or False).
bool	bool	Floating-point numbers. Complex-valued floating-point numbers.
float	float16, float32, float64, float128	Integers.
complex	complex64, complex128, complex256	Unsigned (non-negative) integers. Boolean (True or False).

- Permet un contrôle fin de l’occupation mémoire et de la précision de la représentation
- Une fois l’array créé, changer de type est possible :
  - `data = np.array(data, dtype=np.int)` (copie)
  - `Data = data.astype(np.int)` (changement de type)

Création des arrays

Function name	Type of array
np.array	Creates an array for which the elements are given by an array-like object, which, for example, can be a (nested) Python list, a tuple, an iterable sequence, or another ndarray instance.
np.zeros	Creates an array – with the specified dimensions and data type – that is filled with zeros.
np.ones	Creates an array – with the specified dimensions and data type – that is filled with ones.
np.diag	Creates a diagonal array with specified values along the diagonal, and zeros elsewhere.
np.arange	Creates an array with evenly spaced values between specified start, end, and increment values.
np.linspace	Creates an array with evenly spaced values between specified start and end values, using a specified number of elements.
np.logspace	Creates an array with values that are logarithmically spaced between the given start and end values.
np.meshgrid	Generate coordinate matrices (and higher-dimensional coordinate arrays) from one- dimensional coordinate vectors.

Création des arrays (suite)

Function name	Type of array
np.fromfunction	Create an array and fill it with values specified by a given function, which is evaluated for each combination of indices for the given array size.
np.fromfile	Create an array with the data from a binary (or text) file. NumPy also provides a corresponding function np.tofile with which NumPy arrays can be stored to disk, and later read back using np.fromfile.
np.genfromtxt, np.loadtxt	Creates an array from data read from a text file. For example, a comma-separated value (CSV) file. The function np.genfromtxt also supports data files with missing values.
np.random.rand	Generates an array with random numbers that are uniformly distributed between 0 and 1. Other types of distributions are also available in the np.random module.

Création des arrays (suite)

- Création d'arrays non initialisés :
  ```
  In [61]: np.empty(3, dtype=np.float)
  Out[61]: array([  1.28822975e-231,   1.28822975e-231,
  2.13677905e-314])
  ```
- Création d'arrays à partir d'autres arrays (même tailles):
  `np.ones_like`
  `np.zeros_like`
  `np.full_like`
  `np.empty_like`
- Matrices communes :
  `np.identity`
  `np.eye(3, k=1) (offset optionnel)`
  `np.diag(np.arange(0, 20, 5))`

Indexation et slicing

- Tableau unidimensionnels : syntaxe [ ], identique listes Python ; indexes négatifs : à partir de la fin (-1 dernier, -2 avant dernier, etc.). Possibilité utiliser t-uple.

Expression	Description
a[m]	Select element at index m, where m is an integer (start counting form 0).
a[-m]	Select the mth element from the end of the list, where m is an integer.
a[m:n]	The last element in the list is addressed as -1, the second-to-last element as -2, and so on.
a[:] or a[0:-1]	Select elements with index starting at m and ending at n – 1 (m and n are integers). Select all elements in the given axis.
a[:n]	Select elements starting with index 0 and going up to index n – 1 (integer).
a[m:] or a[m:-1]	Select elements starting with index m (integer) and going up to the last element in the array.
a[m:n:p]	Select elements with index m through n (exclusive), with increment p.
a[::-1]	Select all the elements, in reverse order.

Vues

- Les sous-tableaux extraits avec des opérations de slicing peuvent être affectés à de nouvelles variables
- Dans ce cas, ce ne sont pas des variables indépendantes, mais ce sont des vues des tableaux
- Si l'on modifie le tableau initial, la vue et modifiée, et réciproquement

```
import numpy as np
a=np.ones((4,4))
b=a[::2,::2]
b[0,0]=0
a[2,2]=3
a,b
(array([[ 0.,  1.,  1.,  1.],
        [ 1.,  1.,  1.,  1.],
        [ 1.,  1.,  3.,  1.],
        [ 1.,  1.,  1.,  1.]]),
array([[ 0.,  1.],
       [ 1.,  3.]]))
```

Vues et copies

- Si l'on veut que la variable soit indépendante, il faut créer une copie de la variable

```
import numpy as np
a=np.ones((4,4))
b=a[::2,::2].copy()
b[0,0]=0
a[2,2]=3
a,b
```

```
(array([[ 1.,  1.,  1.,  1.],
        [ 1.,  1.,  1.,  1.],
        [ 1.,  1.,  3.,  1.],
        [ 1.,  1.,  1.,  1.]]),
array([[ 0.,  1.],
       [ 1.,  1.]]))
```

Fancy indexing

- Possibilité d'indexer des arrays au moyen de listes
  exemple :

```
In [94]: A = np.linspace(0, 1, 11)
Out[94]: array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,
0.6,  0.7,  0.8,  0.9,  1. ])
In [95]: A[np.array([0, 2, 4])]
Out[95]: array([ 0. ,  0.2,  0.4])
In [96]: A[[0, 2, 4]]  # même chose avec une liste
Out[96]: array([ 0. ,  0.2,  0.4])
```

- Peut être utilisé avec des valeurs booléennes :
  exemple :

```
In [97]: A > 0.5
Out[97]: array([False, False, False, False, False,
False, True, True, True, True, True], dtype=bool)
In [98]: A[A > 0.5]
Out[98]: array([ 0.6,  0.7,  0.8,  0.9,  1. ])
```

- Les matrices produites sont indépendantes (pas des vues)

Redimensionnement des arrays

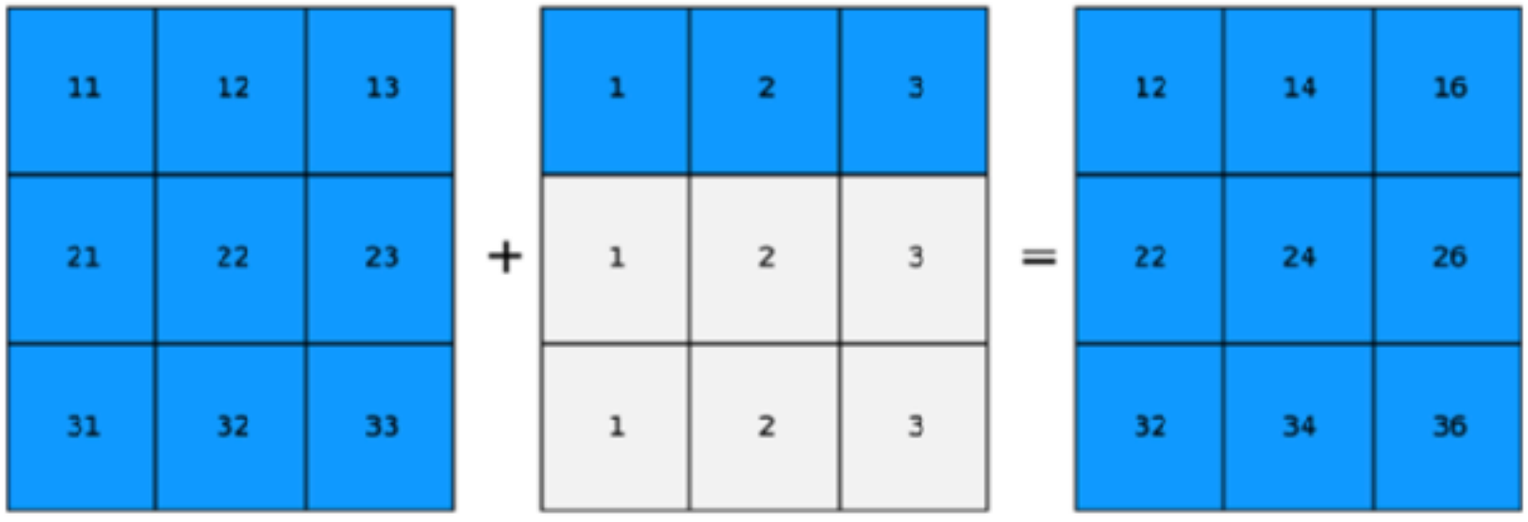
Function / method	Description
np.reshape, np.ndarray.reshape	Reshape an N-dimensional array. The total number of elements must remain the same.
np.ndarray.flatten	Create a copy of an N-dimensional array and reinterpret it as a one-dimensional array (that is, all dimensions are collapsed into one).
np.ravel,np.ndarray.ravel	Create a view (if possible, otherwise a copy) of an N-dimensional array in which it is interpreted as a one-dimensional array.
np.squeeze	Remove axes with length 1.
np.expand_dims, np.newaxis	Adds a new axis (dimension) of length 1 to an array, where np.newaxis is used with array indexing.
np.transpose, np.ndarray.transpose, np.ndarray.T	Transpose the array. The transpose operation corresponds to reversing (or more generally, permuting) the axes of the array.
np.hstack	Stack a list of arrays horizontally (along axis 1): For example, given a list of column vectors, append the columns to form a matrix.
np.vstack	Stack a list of arrays vertically (along axis 0): For example, given a list of row vectors, append the rows to form a matrix.
np.dstack	Stack arrays depth-wise (along axis 2).

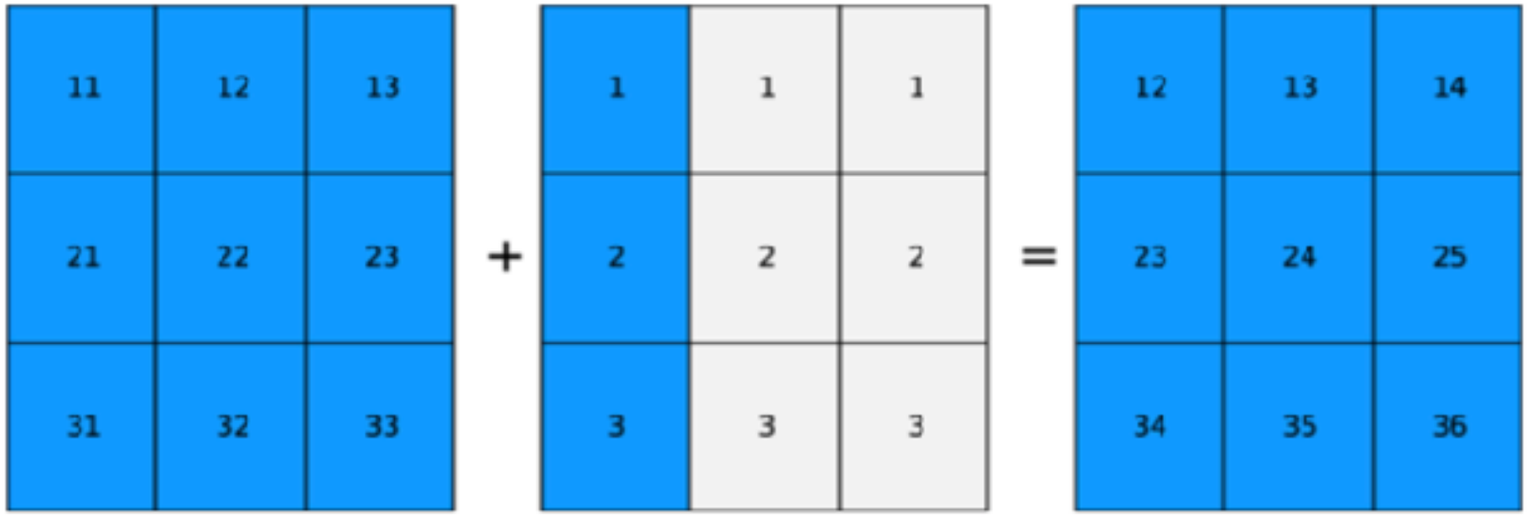
Redimensionnement des arrays

Function / method	Description
np.concatenate	Create a new array by appending arrays after each other, along a given axis.
np.resize	Resize an array. Creates a new copy of the original array, with the requested size. If necessary, the orignal array will repeated to fill up the new array.
np.append	Append an element to an array. Creates a new copy of the array.
np.insert	Insert a new element at a given position. Creates a new copy of the array.
np.delete	Delete an element at a given position. Creates a new copy of the array.

Opérations ‘vectorisées’ et broadcasting

- Le broadcasting consiste à rendre possibles des opérations sur les tableaux qui ne sont pas possible sinon.
- Par exemple, ajout d’un tableau 3x3 avec un tableau 1x3
- Règle de base : si dim=1 les lignes/colonnes sont recopiées pour atteindre la taille de l’autre tableau





Opérations arithmétiques et fonctions ‘vectorisées’

Operator	Operation
+, +=	Addition
-, -=	Subtraction
*, *=	Multiplication
/, /=	Division
//, //=	Integer division
**, **=	Exponentiation

Fonction	Description
np.cos, np.sin, np.tan	Trigonometric functions.
np.arccos, np.arcsin. np.arctan	Inverse trigonometric functions.
np.cosh, np.sinh, np.tanh	Hyperbolic trigonometric functions.
np.arccosh, np.arcsinh, np.arctanh	Inverse hyperbolic trigonometric functions.
np.sqrt	Square root.
np.exp	Exponential.
np.log, np.log2, np.log10	Logarithms of base e, 2, and 10, respectively.

Opérations mathématiques ‘élément par élément’

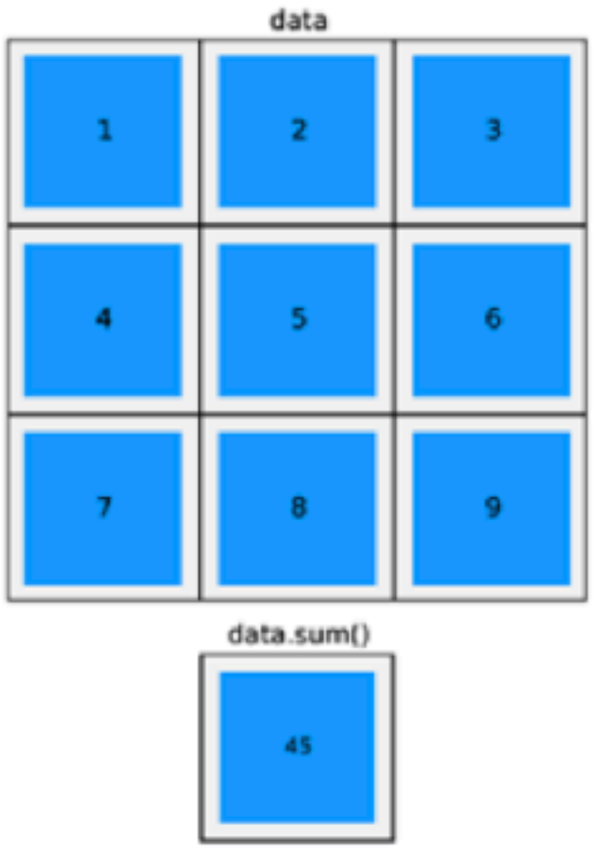
NumPy function	Description
`np.add, np.subtract, np.multiply, np.divide`	Addition, subtraction, multiplication and division of two NumPy arrays.
`np.power`	Raise first input argument to the power of the second input argument (applied elementwise).
`np.remainder`	The remainder of division.
`np.reciprocal`	The reciprocal (inverse) of each element.
`np.real, np.imag, np.conj`	The real part, imaginary part, and the complex conjugate of the elements in the input arrays.
`np.sign, np.abs`	The sign and the absolute value.
`np.floor, np.ceil, np.rint`	Convert to integer values.
`np.round`	Round to a given number of decimals.

Fonctions d'agrégation

- Ces fonctions condensent un array (ou une partie d'array) par un scalaire
- Par défaut, agrégation sur le tableau entier

NumPy Function	Description
`np.mean`	The average of all values in the array.
`np.std`	Standard deviation.
`np.var`	Variance.
`np.sum`	Sum of all elements.
`np.prod`	Product of all elements.
`np.cumsum`	Cumulative sum of all elements.
`np.cumprod`	Cumulative product of all elements.
`np.min, np.max`	The minimum / maximum value in an array.
`np.argmin, np.argmax`	The index of the minimum / maximum value in an array.
`np.all`	Return True if all elements in the argument array are nonzero.
`np.any`	Return True if any of the elements in the argument array is nonzero.

Choix des axes pour l'agrégation



Expressions booléennes

- Les opérateurs <,>,==, etc. Permettent de faire une comparaison terme à terme, et supportent le broadcasting
- Les résultats des opérateurs booléens peuvent être combinés avec les fonctions np.any ou np.all
- Cela permet d'éviter le recours à des branchements conditionnels et de tout écrire sous la forme d'expressions mathématiques :

```
In [191]: x = np.array([-2, -1, 0, 1, 2])
In [192]: x > 0
Out[192]: array([False, False, False,  True,  True], dtype=bool)
In [193]: 1 * (x > 0)
Out[193]: array([0, 0, 0, 1, 1])
In [194]: x * (x > 0)
Out[194]: array([0, 0, 0, 1, 2])
```

Expressions logiques et conditionnelles

Function	Description
`np.where`	Choose values from two arrays depending on the value of a condition array.
`np.choose`	Choose values from a list of arrays depending on the values of a given index array.
`np.select`	Choose values from a list of arrays depending on a list of conditions.
`np.nonzero`	Return an array with indices of nonzero elements.
`np.logical_and`	Perform and elementwise AND operation.
`np.logical_or, np.logical_xor`	Elementwise OR/XOR operations.
`np.logical_not`	Elementwise NOT operation (inverting).

Opérations sur des ensembles

Function	Description
`np.unique`	Create a new array with unique elements, where each value only appears once.
`np.in1d`	Test for the existence of an array of elements in another array.
`np.intersect1d`	Return an array with elements that are contained in two given arrays.
`np.setdiff1d`	Return an array with elements that are contained in one but not the other, of two given arrays.
`np.union1d`	Return an array with elements that are contained in either, or both, of two given arrays.

Opérations sur des arrays

Function	Description
`np.transpose`, `np.ndarray.transpose`, `np.ndarray.T`	The transpose (reverse axes) of an array.
`np.fliplr / np.flipud`	Reverse the elements in each row / column.
`np.rot90`	Rotate the elements along the first two axes by 90 degrees.
`np.sort`, `np.ndarray.sort`	Sort the element of an array along a given specified axis (which default to the last axis of the array). The np.ndarray method sort performs the sorting in place, modifying the input array.

Opérations matricielles

NumPy Function	Description
`np.dot`	Matrix multiplication (dot product) between two given arrays representing vectors, arrays, or tensors.
`np.inner`	Scalar multiplication (inner product) between two arrays representing vectors.
`np.cross`	The cross product between two arrays that represent vectors.
`np.tensordot`	Dot product along specified axes of multidimensional arrays.
`np.outer`	Outer product (tensor product of vectors) between two arrays representing vectors.
`np.kron`	Kronecker product (tensor product of matrices) between arrays representing matrices and higher-dimensional arrays.
`np.einsum`	Evaluates Einstein's summation convention for multidimensional arrays.

Références

Ouvrages utilisés pour préparer ce cours