



PyQB

Monga

ndarr  
ay

Creation

Indexing

Vectorization

Array operations

Homework

# Programming in Python<sup>1</sup>

Mattia Monga

Dip. di Informatica  
Università degli Studi di Milano, Italia

[mattia.monga@unimi.it](mailto:mattia.monga@unimi.it)

Academic year 2020/21, II semester



PyQB

Monga

ndarr  
ay

Creation

Indexing

Vectorization

Array operations

Homework

# Lecture XI: NumPy arrays



# NumPy arrays

The most important data structure in NumPy is `ndarray`: a (usually fixed-size) sequence of same type elements, organized in one or more dimensions.

<https://numpy.org/doc/stable/reference/arrays.ndarray.html>

Implementation is based on byte arrays: accessing an element (all of the same byte-size) is virtually just the computation of an 'address'.

PyQB

Monga

`ndarray`

Creation

Indexing

Vectorization

Array operations

Homework





A ndarray has a dtype (the type of elements) and a shape (the length of the array on each dimensional **axis**). (Note the jargon: slightly different from linear algebra)

- Since appending is costly, normally they are pre-allocated (zeros, ones, arange, linspace, ...)
- vectorized operations can simplify code (no need for loops) and they are faster with big arrays
- vector indexing syntax (similar to R): very convenient (but you need to learn something new)



# All the elements must have the same size

This is actually a big limitation: the faster access comes with a price in flexibility.

```
>>> np.array(['', '', ''])
array(['', '', ''], dtype='<U1')
>>> np.array(['a', 'bb', 'ccc'])
array(['a', 'bb', 'ccc'], dtype='<U3')
>>> np.array(['a', 'bb', 'cccccccccccccccccccccccc'])
array(['a', 'bb', 'cccccccccccccccccccccccc'], dtype='<U21')
```

PyQB

Monga

ndarray

Creation

Indexing

Vectorization

Array operations

Homework



# Usually the length is not changed

The best use of arrays is to avoid a change in their length, that can be costly. Thus, they are normally **preallocated** at creation:

- `np.array([1,2,3])`
- `np.zeros(2)`, `np.zeros(2, float)`, `np.ones(2)`
- `np.empty((2,3))` six not meaningful float values
- `np.arange(1, 5)` be careful with floats:  

```
>>> np.arange(0.4, 0.8, 0.1)
array([0.4, 0.5, 0.6, 0.7])
>>> np.arange(0.5, 0.8, 0.1)
array([0.5, 0.6, 0.7, 0.8])
```
- `np.linspace(0.5, 0.8, 3)` with this the length is easier to predict

You can concatenate arrays with `np.concatenate` (be careful with the shapes!)

PyQB

Monga

ndarray

Creation

Indexing

Vectorization

Array operations

Homework



# Don't remove, select

In general you don't remove elements but select them. Be careful: if you don't make an explicit **copy** you get a “view” and possibly side-effects.

```
>>> a = np.ones((2,3))
>>> a
array([[1., 1., 1.],
       [1., 1., 1.]])
>>> x = a[:, 1]
>>> x
array([1., 1.])
>>> x[0] = 0
>>> x
array([0., 1.])
>>> a
array([[1., 0., 1.],
       [1., 1., 1.]])
```

```
>>> x = a[:, 1].copy()
>>> x[1] = 100
>>> x
array([ 0., 100.])
>>> a
array([[1., 0., 1.],
       [1., 1., 1.]])
```

PyQB

Monga

ndarray

Creation

Indexing

Vectorization

Array operations

Homework



# Indexing is powerful

```
a = np.arange(1, 6)
```

1	2	3	4	5
0	1	2	3	4

`a[1]`

2
---

`a[2:4]`

3	4
---	---

`a[-2:]`

4	5
---	---

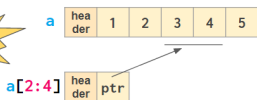
`a[::2]`

1	3	5
---	---	---

`a[[1,3,4]]`

2	4	5
---	---	---

"fancy indexing"



`a[2:4] = 0`

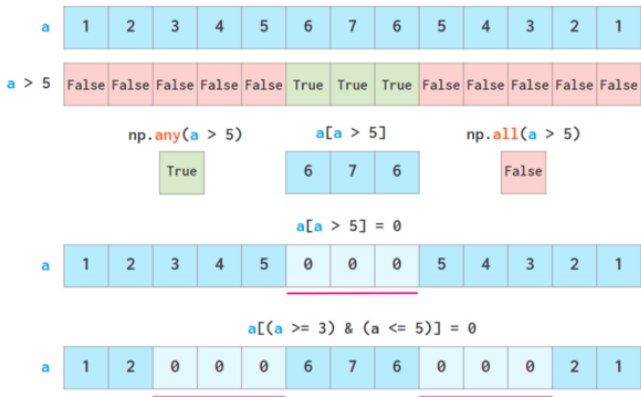


`a`

1	2	0	0	5
---	---	---	---	---

Picture from "NumPy Illustrated: The Visual Guide to NumPy", highly recommended

# Indexing is powerful



Picture from "NumPy Illustrated: The Visual Guide to NumPy", highly recommended



# The highest power: vectorization

Most of the basic mathematical function are **vectorized**: no need for loops! This is both convenient and faster!

```
>>> a = np.array([1,2,3,4])
>>> a + 1
array([2, 3, 4, 5])
>>> a ** 2
array([ 1,  4,  9, 16])
>>> np.exp(a)
array([ 2.71828183,  7.3890561 , 20.08553692,
        ↪ 54.59815003])
```

PyQB

Monga

ndarray

Creation

Indexing

Vectorization

Array operations

Homework



# Array operations

On arrays you have many “aggregate” operations.

```
>>> a
array([1, 2, 3, 4])
>>> a.sum()
10
>>> a.max()
4
>>> a.argmin()
0
>>> a.mean()
2.5
```

Remember to look at [dir](#) or the online documentation.

PyQB

Monga

ndarray

Creation

Indexing

Vectorization

Array operations

Homework



PyQB

Monga

ndarr  
ay

Creation

Indexing

Vectorization

Array operations

Homework

<https://classroom.github.com/a/C0132nII>