Programming in Python[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Academic year 2024/25, I semester

Lecture XV: NumPy arrays

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!)

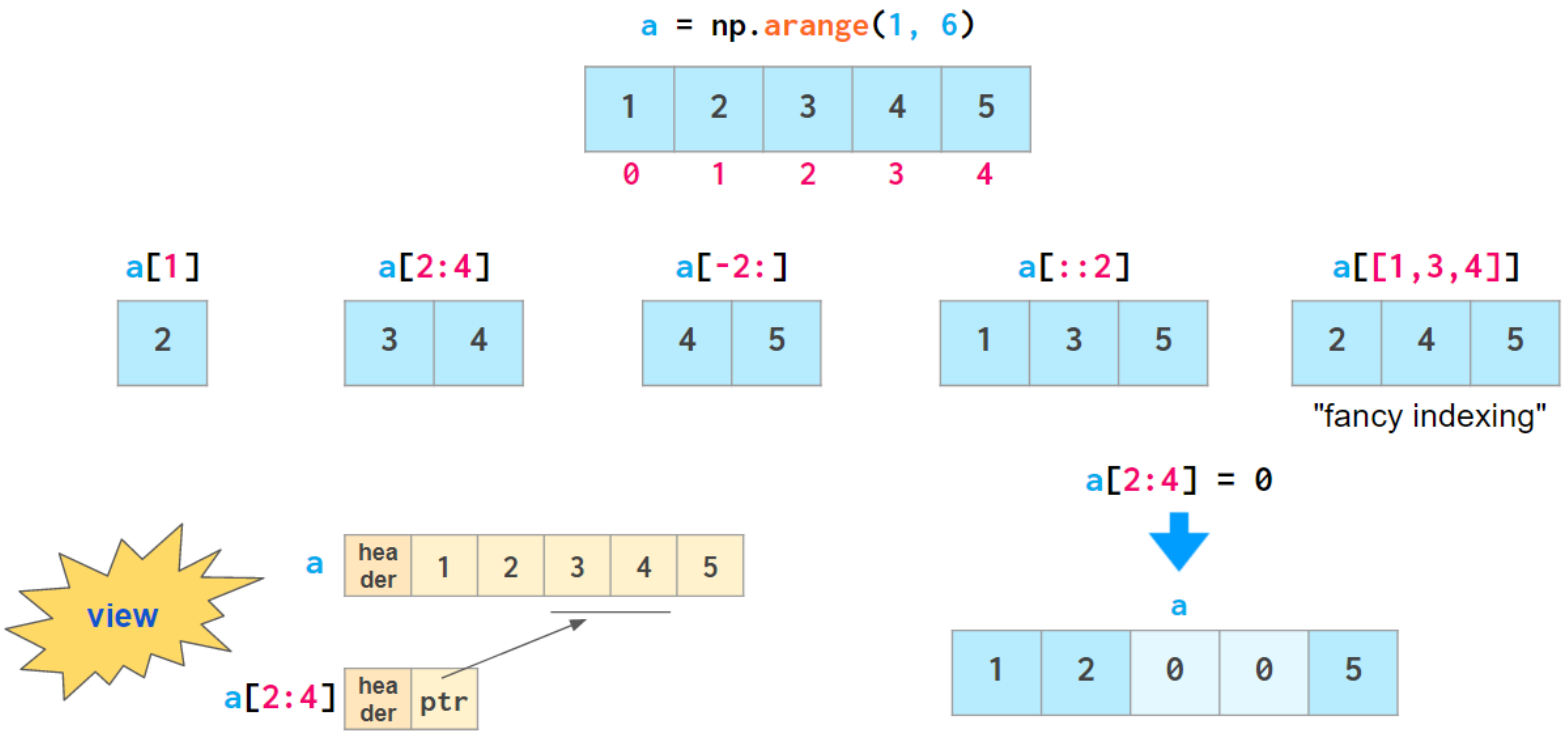
Don't remove, select

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.]])
```

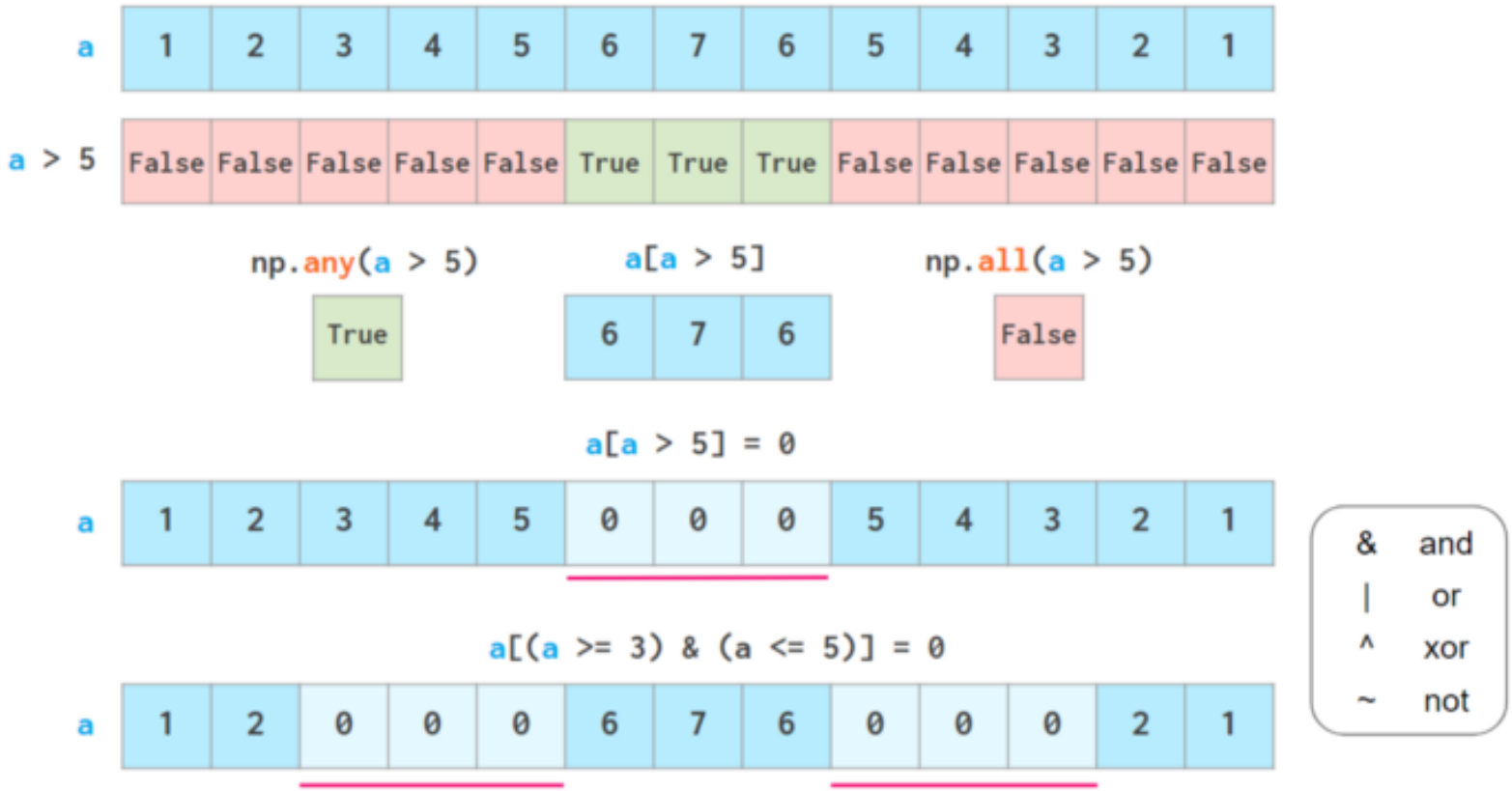
```
>>> x = a[:, 1].copy()
>>> x[1] = 100
>>> x
array([  0., 100.])
>>> a
array([[1., 0., 1.],
       [1., 1., 1.]])
```

Indexing is powerful



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

Warning! Assignment works differently from lists


```
>>> np = np.array([1,2,3,4,5])
>>> lst = [1,2,3,4,5]
>>> np[2:4] = 0
>>> np
array([1, 2, 0, 0, 5])
>>> lst[2:4] = 0 # Error!
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only assign an iterable
>>> lst[2:4] = [0,0]
>>> lst
[1, 2, 0, 0, 5]
>>> lst[2:4] = [0,0,0]
>>> lst
[1, 2, 0, 0, 0, 5]
>>> np[2:4] = [0,0]
>>> np[2:4] = [0,0,0] # Error!
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not broadcast input array from shape (3,) into
↪  shape (2,)
```

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])
```

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.