NumPy

Efficient numerics in Python

Fernando Pérez

<Fernando.Perez@berkeley.edu>

John Hunter

<jdh2358@gmail.com>

Neuroscience, UC Berkeley
Tradelink

Joint Sciences Division, Claremont, CA
Oct 25, 2008

Python is a general programming language

Flexible and high level

- Basic numbers: ints, floats and complex.
- Lists:
  - flexible containers.
  - can hold any python object.
  - can grow and shrink.

```
In [12]: alist=[1,3.5,'hello world',
['a sublist',99]]
In [13]: alist[2]
Out[13]: 'hello world'
In [14]: alist.append('more')
In [15]: alist
Out[15]: [1, 3.5, 'hello world', ['a sublist', 99], 'more']
```

- math and cmath modules:
  - basic functionality (sqrt, ext, sin, cos, etc)

Is it good enough for scientific computing?

Efficient numerical processing?

- Compiler-specific numbers needed: ints (32, 64-bit), floats, etc.
- Homogeneous arrays of such elements.
- Easy arithmetic on entire arrays that is efficient.
- Comprehensive math library to operate on arrays.
- Common linear algebra support.
- Needed for:
  - Mathematics
  - Image processing
  - Data analysis
  - ... Just about anything remotely connected to scientific computing.

A tiny bit of history

1990s-2004: Numeric

- Started by Jim Hugunin, MIT grad student.
- Developed by many: national labs, academia, industry.
- Fast and light C code.
- Difficult to maintain and extend.

2004: Numarray

- Perry Greenfield and team (Hubble Space Telescope)
- Clean codebase, new ideas.
- Many new features and documentation.
- Some performace issues lingered

2005-future: NumPy: unification effort

- Led by Travis Oliphant, with Perry's full support.
- Best of Numeric and Numarray, into a new codebase
- The whole community rallied behind the effort.

Today: only NumPy for any new code!

NumPy: key ideas

- A flexible, efficient, multidimensional array object.
- Homogeneous elements
  - Supports all native types (ints, floats, etc).
  - Arbitrary user-defined types of fixed size.
  - Arbitrary Python objects can also be stored.
- Convenient syntax for high-level operations.
- Math library that operates on arrays.
- Basic scientific functionality:
  - Linear algebra
  - FFTs
  - Random number generation

NumPy: flexible arrays

- Array is a container of objects “of the same kind”: **homogeneous**.
- Concept of “kind” embodied in the data type, or **dtype**.
- Dtypes **can be user-defined** to be arbitrarily complex.

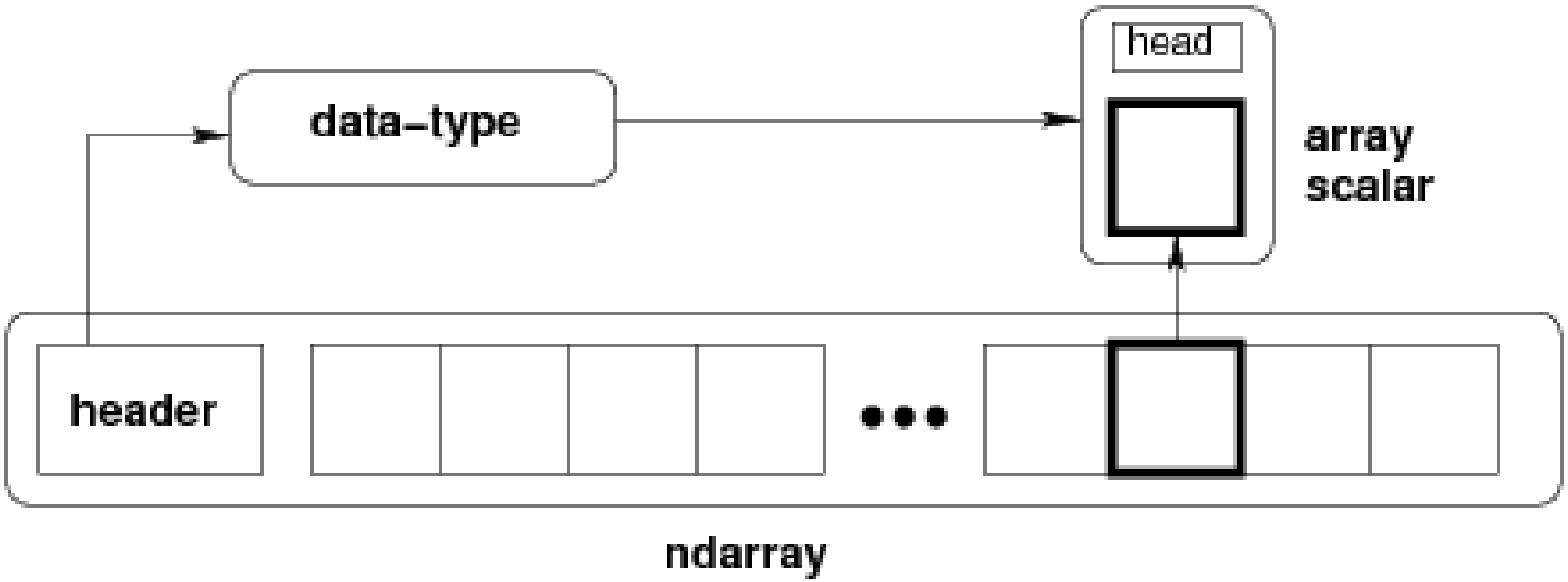
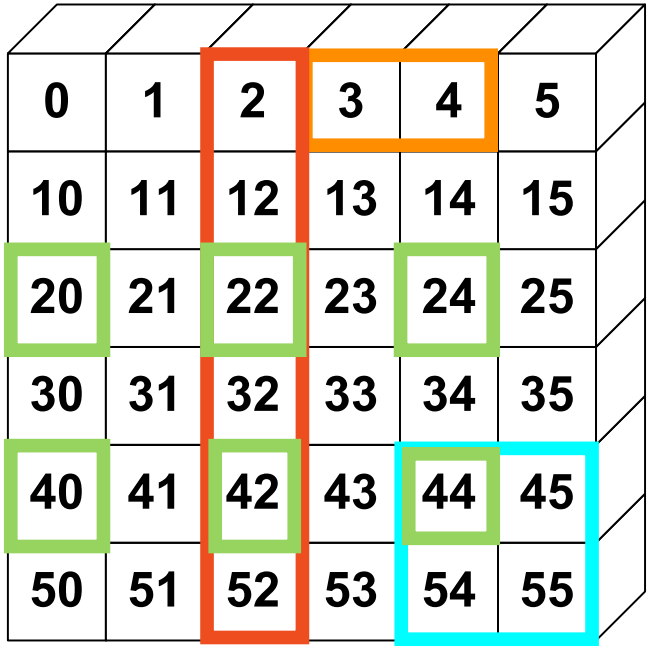


Image credit: T. Oliphant, Enthought Inc.

NumPy: indexing

Image credit: E. Jones, Enthought Inc.

```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,22,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```

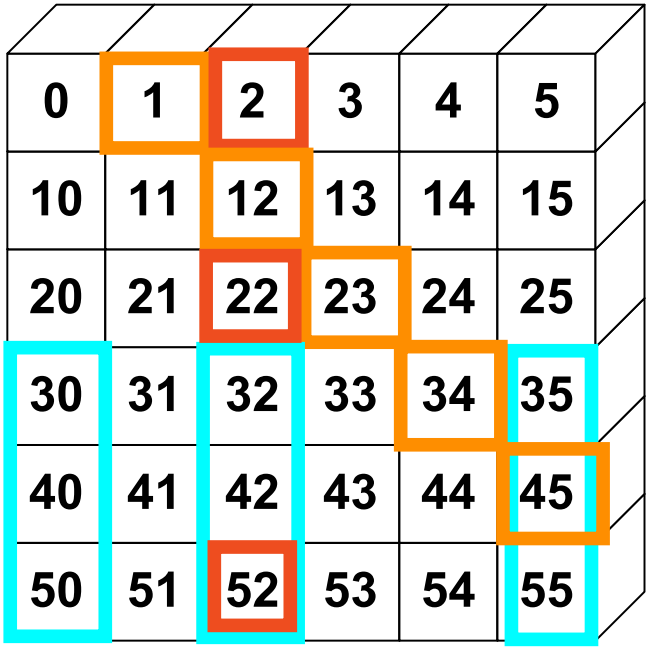


Slicing does not create copies of the array's contents

NumPy: fancy indexing

Image credit: E. Jones, Enthought Inc.

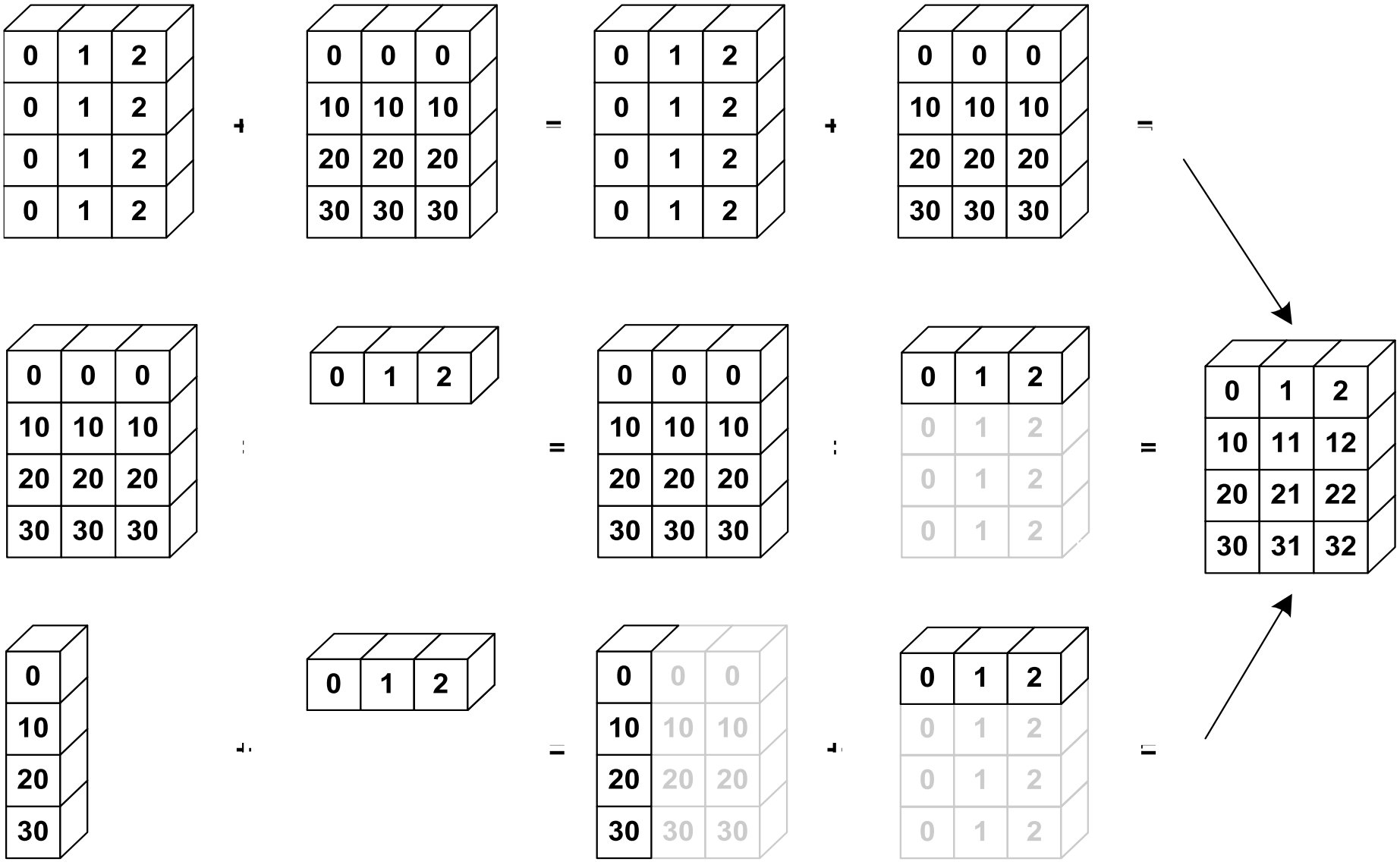
```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35],
       [40, 42, 45]])
       [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                 dtype=bool)
>>> a[mask,2]
array([2,22,52])
```



Unlike slicing, fancy indexing creates copies instead of views into original arrays.

NumPy: broadcasting

Image credit: E. Jones, Enthought Inc.



SciPy: numerical algorithms galore

- **linalg** : Linear algebra routines (including BLAS/LAPACK)
- **sparse** : Sparse Matrices (including UMFPACK, ARPACK,...)
- **fftpack** : Discrete Fourier Transform algorithms
- **cluster** : Vector Quantization / Kmeans
- **odr** : Orthogonal Distance Regression
- **special** : Special Functions (Airy, Bessel, etc).
- **stats** : Statistical Functions
- **optimize** : Optimization Tools
- **maxentropy** : Routines for fitting maximum entropy models
- **integrate** : Numerical Integration routines
- **ndimage** : n-dimensional image package
- **interpolate** : Interpolation Tools
- **signal** : Signal Processing Tools
- **io** : Data input and output