

# NumPy 2

Week 3, day 5

# Numpy

- `import numpy`
- `ndarray`:
  - N-dimensional matrices, copy by default
  - Fixed type from hierarchy
  - Creat from shape or matrix-like object
- `ufuncs` : functors for element-by-element transformation of matrices

# Initializing ndarrays

- Matrix-like functions copy contents
- Shape-based functions:
  - zeros: create an array of all zeros
  - ones: create an array of all ones
  - identity: create a zero array with 1s on diagonal
  - fromfunction: create array with elements set by func (i, j, k ...)

# Changing size & shape

- reshape method
- Assign direct to shape member
- resize methods
- Functional programming ufuncs

# Internal memory layout

- Often irrelevant but actually linear
- Order:
  - “last index varies first”
  - for i in dimension1:  
for j in dimension2:  
for k in dimension3: ...

0 (0,0)	1 (0,1)	2 (0,2)
3 (1,0)	4 (1,1)	5 (1,2)
6 (2,0)	7 (2,1)	8 (2,2)
9 (3,0)	10 (3,1)	11 (3,2)

C

# Indexing and slicing

- Slice notation works just like sequences
- `seq [start:stop:step]`
- If arguments not supplied, assumed to be 0, end, and 1
- `array [slice, slice, slice ...]`
- If no slice provided, assume everything

# Operations & ufuncs

- Not matrix algebra but element by element operations
- Input & output all of the same size but ...

1	7	8
9	0	1
3	2	2

 + 

0	7	1
9	3	1
0	4	7



1	14	9
18	3	2
3	6	9

# Operations on arrays of unequal size

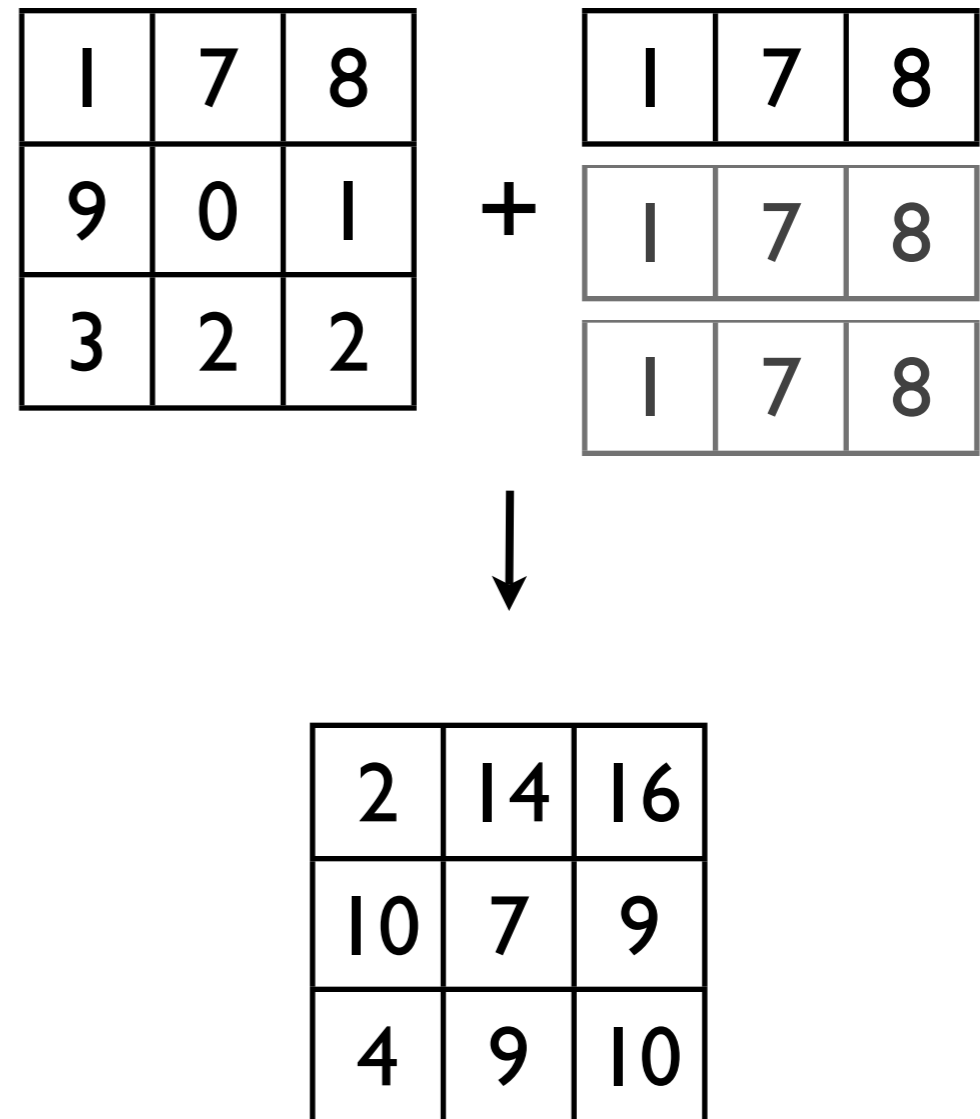
```
>>> a = numpy.array ([[1, 2, 3], [1, 2, 3], [1, 2, 3]])
>>> a
array([[1, 2, 3],
       [1, 2, 3],
       [1, 2, 3]])

>>> b = numpy.array ([[1, 2, 3], [1, 2, 3], [1, 2, 3]])
>>> a + b
array([[2, 4, 6],
       [2, 4, 6],
       [2, 4, 6]])

>>> c = numpy.array ([1, 2, 3])
>>> a + c
array([[2, 4, 6],
       [2, 4, 6],
       [2, 4, 6]])
```

# Operations & ufuncs

- In fact, arrays of different sizes are allowable if the smaller arrays can be replicated to match shapes
- Duplicate last dimension



# ndarray members

- `dtype`: type object
- `dtype.type`: string representation of type
- `ndim`: number of dimensions
- `size`: number of elements

# ndarray methods

- `tolist()`: return as a nested list
- `tofile (file [, sep], [, format])`: write to file name or object with entries separated by `sep` (default binary) and data in string format `format` (default `%s`)
- `dump (file)`: store (pickle) into file
- `dumps ()`: return pickled string of array

# ndarray methods 2

- `astype (type)`: return array converted to type
- `copy ()`: return copy of array
- `fill (val)`: fill array with value
- `transpose ([shape])`: transpose / flip array

# ndarray methods 3

- `choose (arr)`:  
select elements  
from array by  
indices
- `sort`  
(`axis=-1`)
- `nonzero()`:  
return list of  
nonzero indices
- magic methods

```
>>> a = array  
([0,3,2,1])  
  
>>> a.choose(  
    [0,1,2,3],  
    [10,11,12,13],  
    [20,21,22,23],  
    [30,31,32,33]  
)  
  
array([ 0, 31, 22, 13])
```

# ndarray methods 4

- `max()`: return maximum value
- `argmax()`: return indices of maximum
- `min()`, `argmin()`
- `clip(min=, max=)`: return array where values are transformed to within min and max
- `sum`, `mean`, `var`, `std`, `cov`,  
`median`: dispersal measures

# Numpy functions

- `fromstring()`: reverse of `tostring()`
- `fromfile()`: reverse of `tofile()`
- `load()`: reverse of `dump()`
- `concatenate (arr_seq, axis)`:  
append arrays
- `correlate, convolve, dot`  
`( [shape] )`: transpose / flip array