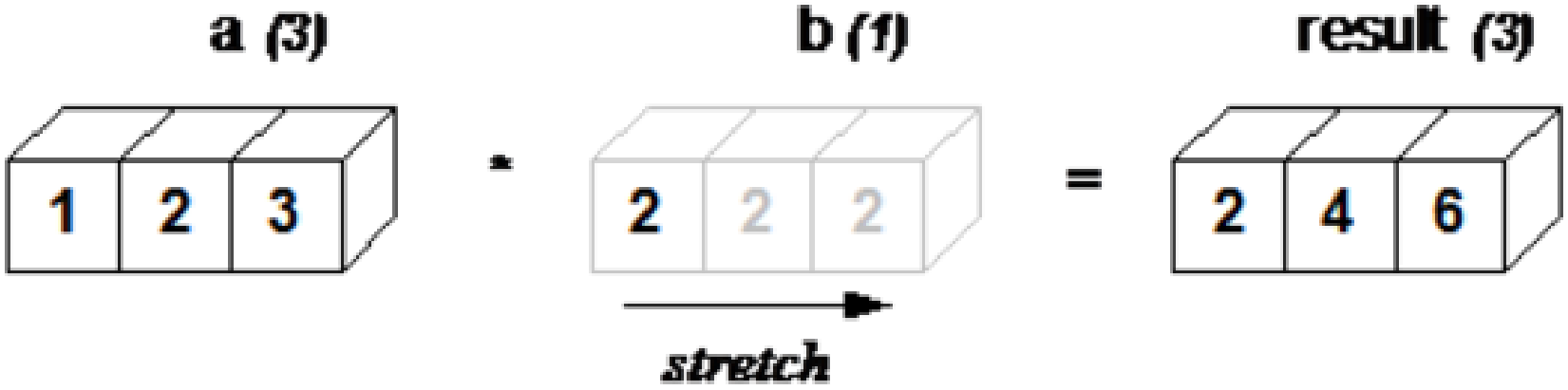
Array Broadcasting

- Broadcasting describes how numpy treats arrays with different shapes during arithmetic operations
- The smaller array is "broadcast" across the larger array so that they have compatible shapes

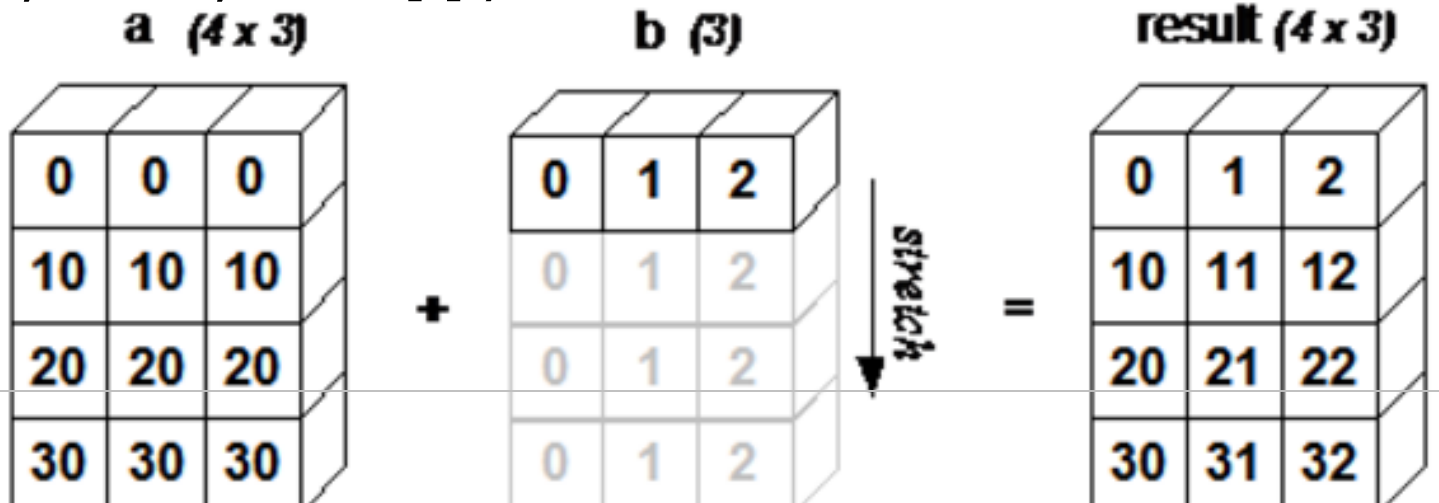
Array Broadcasting

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
>>> b = 2
>>> a * b
array([2,  4,  6])
```



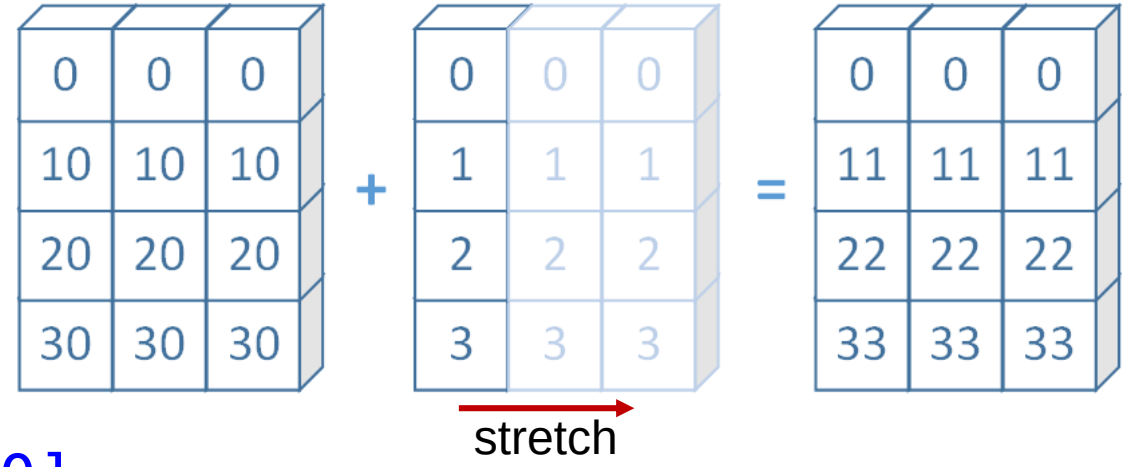
Array Broadcasting

```
>>> a = np.array([[ 0,  0,  0],
...               [10, 10, 10],
...               [20, 20, 20],
...               [30, 30, 30]])
>>> b = np.array([0, 1, 2])
>>> a + b
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```



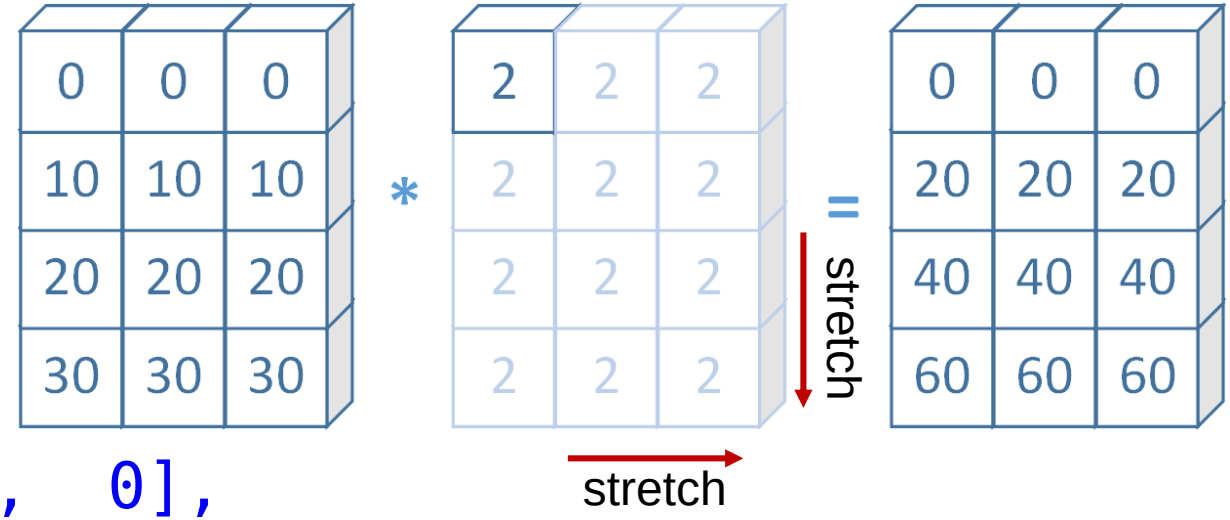
Array Broadcasting

```
>>> a = array([[ 0,  0,  0],
...           [10, 10, 10],
...           [20, 20, 20],
...           [30, 30, 30]])
>>> b = np.array([0,1,2,3]).reshape(4,1)
>>> b
array([[0],
       [1],
       [2],
       [3]])
>>> a + b
array([[ 0,  0,  0],
       [11, 11, 11],
       [22, 22, 22],
       [33, 33, 33]])
```



Array Broadcasting

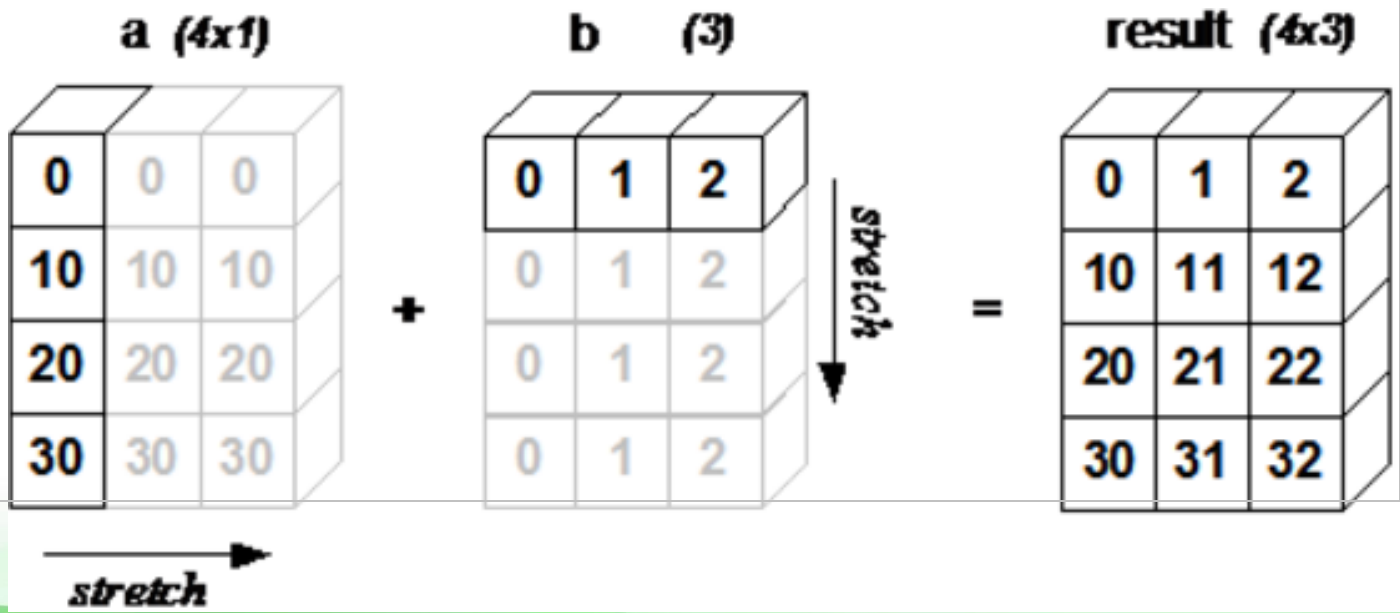
```
>>> a = array([[ 0,  0,  0],
...           [10, 10, 10],
...           [20, 20, 20],
...           [30, 30, 30]])
>>> b = 2
```



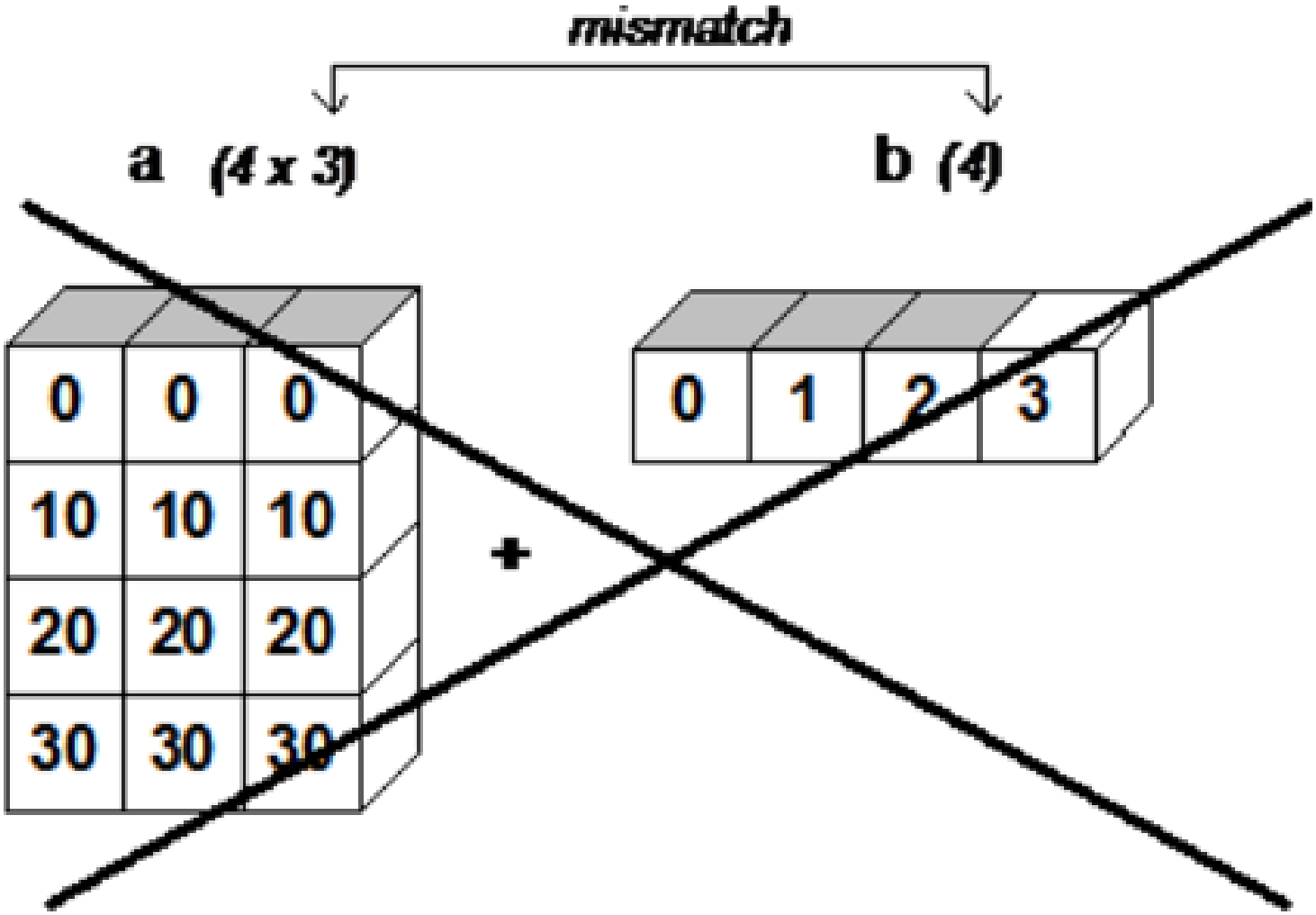

```
>>> a * b
array([[ 0,  0,  0],
       [20, 20, 20],
       [40, 40, 40],
       [60, 60, 60]])
```

Array Broadcasting

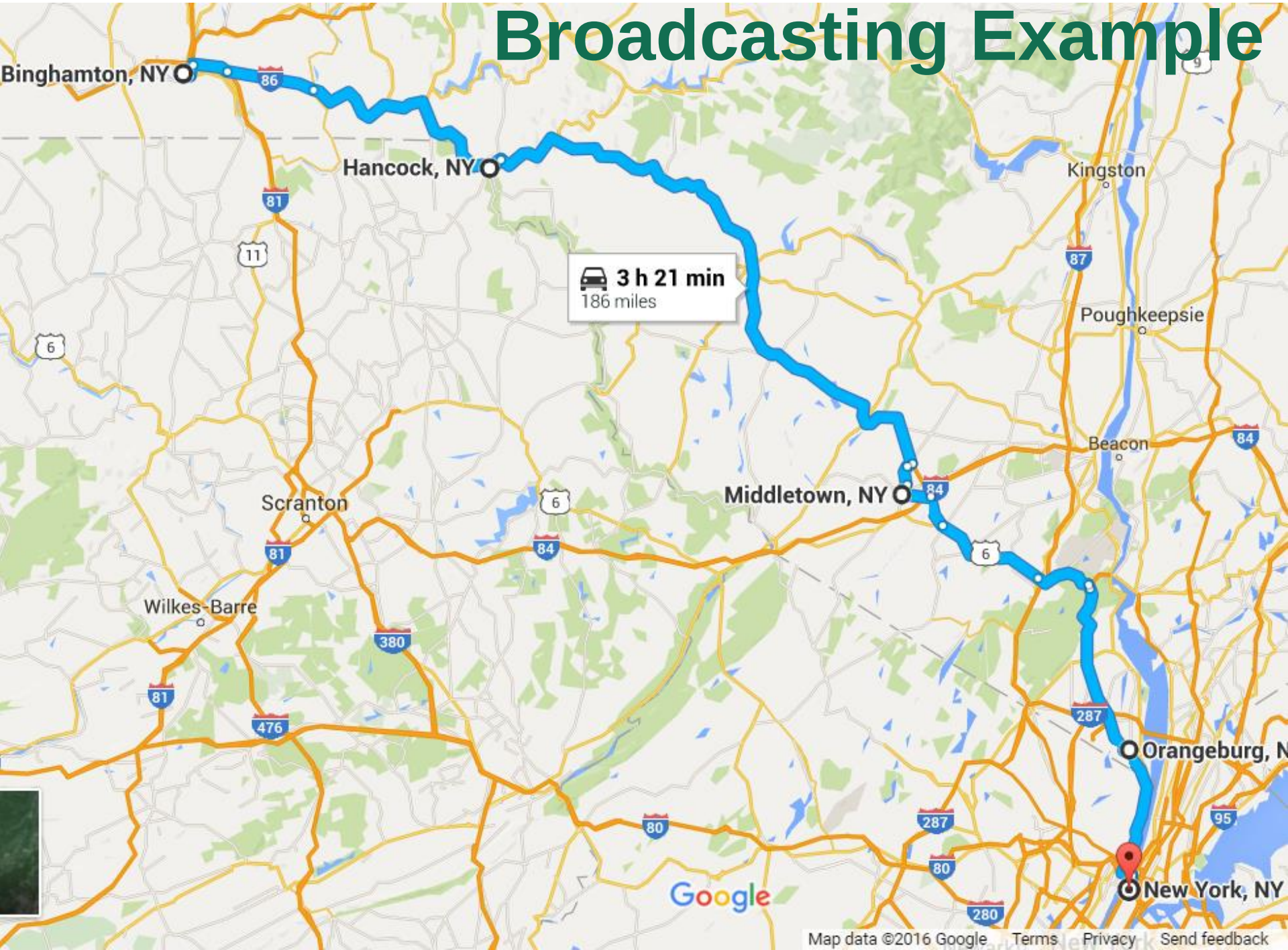
```
>>> a = np.array([0, 10, 20, 30]).reshape(4,1)
>>> b = np.array([0, 1, 2])
>>> a + b
array([[ 0,   1,   2],
       [10,  11,  12],
       [20,  21,  22],
       [30,  31,  32]])
```



Array Broadcasting



Broadcasting Example



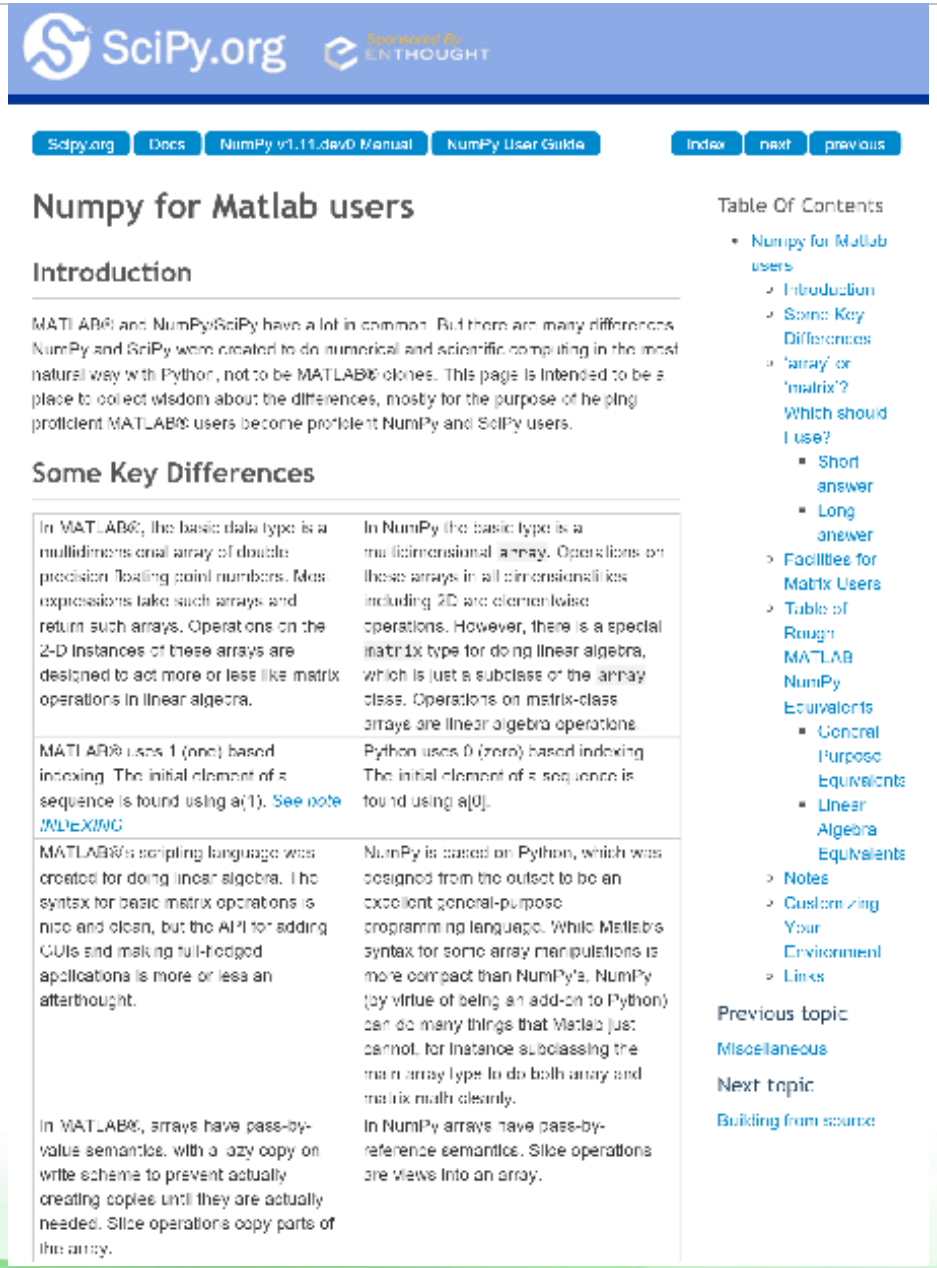
Broadcasting Example

- Calculate the distances between any two locations

(Binghamton, Hancock, Middletown, Orangeburg, NYC)

```
>>> miles = np.array([0, 43, 120, 168, 186])
>>> miles.reshape(5,1)
array([[  0],
       [ 43],
       [120],
       [168],
       [186]])
>>> dis_array = miles - miles.reshape(5,1)
>>> dis_array
array([[   0,   43,  120,  168,  186],
       [ -43,    0,   77,  125,  143],
       [-120,  -77,    0,   48,   66],
       [-168, -125,  -48,    0,   18],
       [-186, -143,  -66,  -18,    0]])
```

NumPy for Matlab Users



SciPy.org — Sponsored By ENTHOUGHT

SciPy.org | Docs | NumPy v1.11.dev0 Manual | NumPy User Guide | index | next | previous

Numpy for Matlab users

Introduction

MATLAB® and NumPy/SciPy have a lot in common. But there are many differences. NumPy and SciPy were created to do numerical and scientific computing in the most natural way with Python, not to be MATLAB® clones. This page is intended to be a place to collect wisdom about the differences, mostly for the purpose of helping proficient MATLAB® users become proficient NumPy and SciPy users.

Some Key Differences

In MATLAB®, the basic data type is a multidimensional array of double precision floating point numbers. Most expressions take such arrays and return such arrays. Operations on the 2-D instances of these arrays are designed to act more or less like matrix operations in linear algebra.	In NumPy the basic type is a multidimensional array. Operations on these arrays in all dimensionalities including 2D are elementwise operations. However, there is a special matrix type for doing linear algebra, which is just a subclass of the array class. Operations on matrix-class arrays are linear algebra operations.
MATLAB® uses 1 (one) based indexing. The initial element of a sequence is found using a(1). See note INDEXING	Python uses 0 (zero) based indexing. The initial element of a sequence is found using a[0].
MATLAB®'s scripting language was created for doing linear algebra. The syntax for basic matrix operations is nice and clean, but the API for adding GUIs and making full-fledged applications is more or less an afterthought.	NumPy is based on Python, which was designed from the outset to be an excellent general-purpose programming language. While Matlab's syntax for some array manipulations is more compact than NumPy's, NumPy (by virtue of being an add-on to Python) can do many things that Matlab just cannot, for instance subclassing the main array type to do both array and matrix math cleanly.
In MATLAB®, arrays have pass-by-value semantics, with a lazy copy on write scheme to prevent actually creating copies until they are actually needed. Slice operations copy parts of the array.	In NumPy arrays have pass-by-reference semantics. Slice operations are views into an array.

Table Of Contents

- Numpy for Matlab users
  - Introduction
  - Some Key Differences
  - 'array' or 'matrix'? Which should I use?
    - Short answer
    - Long answer
  - Facilities for Matrix Users
  - Table of Rough MATLAB NumPy Equivalents
    - General Purpose Equivalents
    - Linear Algebra Equivalents
  - Notes
  - Customizing Your Environment
  - Links

Previous topic

Miscellaneous

Next topic

Building from source

Performance Comparison Between Python and NumPy