Multi-Dimensional Arrays

- Rules for arrays of any number of dimensions:

1. Numbers directly separated by commas are in separate columns,
2. Innermost bracketed groups separated by commas are in different rows,
3. Second-innermost bracketed groups separated by commas are on different "pages",
4. and so on….

zero commas, one number → one column

```
c = np.array([[[1], [2]], [[3], [4]]])
```

```
array([[[1],
        [2]],

       [[3],
        [4]]])
```

one comma, two groups → two rows

one comma, two groups → two pages

… pages, rows, columns

```
c.shape
c.ndim
```

```
(2L, 2L, 1L)
3
```

What is the relationship between `c.ndim` and `c.shape`? (Write a Python expression involving them that is always `True`)

Indexing Multi-Dimensional Arrays

- Rules for arrays of any number of dimensions:

1. Last index selects the column,
2. Second to last index selects the row,
3. Third to last index selects the page,
4. and so on….

```
c = np.random.rand(3)
```

```
array([ 0.5664878 ,  0.8279805 ,  0.04935606])
```

c[1]

```
c = np.random.rand(2, 3)
```

```
array([[ 0.26515998,  0.16943515,  0.7184211 ],
       [ 0.2401759 ,  0.05123935,  0.45890346]])
```

c[0,2]

c[1,0]

```
c = np.random.rand(2, 2, 3)
```

```
array([[[ 0.21282285,  0.85749608,  0.26390172],
        [ 0.55234231,  0.76093889,  0.53984949]],

       [[ 0.95962417,  0.93538958,  0.11380988],
        [ 0.04033667,  0.95009481,  0.77611468]]])
```

c[1, 0, 2]

Slicing and Fancy Indexing

```
x  = np.arange(6)
```

```
array([0, 1, 2, 3, 4, 5])
```

How can I use indexing to get:

```
array([0, 5])
```

```
x[::5]
x[0::5]
```

Slicing

```
x[[0, 5]]
x[np.array([0,-1])]
```

Fancy Indexing (index list or array)

```
x[x%5==0]
x[np.logical_or(x<1, x>=5)]
```

Boolean masking

Copies vs Views

```
x  = np.arange(6)
```

```
array([0, 1, 2, 3, 4, 5])
```

```
y = x[::5]
y[0] = 10
print y
print x
```

Slicing creates a *view*.

```
array([10, 5])
array([10, 1, 2, 3, 4, 5])
```

```
y = x[[0,5]]
y[0] = 10
print y
print x
```

Fancy indexing creates a *copy*.

```
array([10, 5])
array([0, 1, 2, 3, 4, 5])
```

Copies vs Views

```
x  = np.arange(10)

y  = np.reshape((2,5))
y  = np.reshape(2,-1)
```

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

```
x[0] = 10
print y
```

```
array([[10, 1, 2, 3, 4],
       [ 5, 6, 7, 8, 9]])
```

Broadcasting

- Numpy has a rule for performing operations between arrays of different shapes

My attempt at stating it simply:

If dimension *i* of array `a` is 1 and dimension *i* of array `b` is *n*, then `a` is copied along the i^{th} dimension *n* times.

Now the dimensions of the two arrays match and the operation can be performed.)