2_Arrays_and_matrices

January 27, 2018

A brief introduction to working with arrays of numbers in Python.
Last Modified: 27 Jan 2018 Humans Responsible: The Prickly Pythons

1 Lists and Numpy arrays

```
In [1]: # Python has built-in 'lists':
        x = [1, 2, 3]

In [2]: x

Out[2]: [1, 2, 3]

In [3]: # But they don't work as you would expect
        x*2

Out[3]: [1, 2, 3, 1, 2, 3]

In [4]: # And most vector-operations fail
        x*2.3

        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-4-c4e56f154bc0> in <module>()
          1 # And most vector-operations fail
    ----> 2 x*2.3


        TypeError: can't multiply sequence by non-int of type 'float'


In [5]: # Numpy: The module for scientific computing with python
        import numpy as np

In [6]: x = np.array([1,2,3])

In [7]: x

Out[7]: array([1, 2, 3])

In [8]: type(x)

Out[8]: numpy.ndarray
```

```
In [9]: x.nbytes # Number of Bytes required to store x
```

```
Out[9]: 24
```

```
In [10]: # Now you can manipulate your array in every way thinkable:
         x*2.3
```

```
Out[10]: array([ 2.3,  4.6,  6.9])
```

```
In [11]: x/100.
```

```
Out[11]: array([ 0.01,  0.02,  0.03])
```

```
In [12]: x**2
```

```
Out[12]: array([1, 4, 9])
```

```
In [13]: # numpy arrays can be combined with lists
         y = [10.,10,10]
         x+y
```

```
Out[13]: array([ 11.,  12.,  13.])
```

```
In [14]: x/y
```

```
Out[14]: array([ 0.1,  0.2,  0.3])
```

2 Matrix and vector operations

Let's define three quantities

```
In [15]: x = np.array([1,2,3])
         y = np.array([1,1,1])
         A = np.array([[2,3,5],[7,11,13],[17,19,23]]) # Setting up a matrix

         print(x)
         print(y)
         print(A)
```

```
[1 2 3]
[1 1 1]
[[ 2  3  5]
 [ 7 11 13]
 [17 19 23]]
```

2.0.1 Operation 1:

The dot (or scalar) product, $\mathbf{x} \cdot \mathbf{y}$, is

```
In [16]: np.dot(x,y)
```

```
Out[16]: 6
```

2.0.2 Operation 2:

The cross (or vector) product, $\mathbf{x} \times \mathbf{y}$, is

```
In [17]: np.cross(x,y)
```

```
Out[17]: array([-1,  2, -1])
```

2.0.3 Operation 3:

The element-wise multiplication, sometimes known as a Hadamard product, $\mathbf{x} \circ \mathbf{y}$, is

```
In [18]: x*y
```

```
Out[18]: array([1, 2, 3])
```

2.0.4 Operation 4:

This is $A\mathbf{x}$

```
In [19]: np.dot(A,x)
```

```
Out[19]: array([ 23,  68, 124])
```

This is $\mathbf{x}^{\mathrm{T}}A$

```
In [20]: np.dot(x,A)
```

```
Out[20]: array([ 67,  82, 100])
```

Note: `np.dot(A,x)` is not the same as `A*x`. `A*x` gives the element-wise product of x with each rows of A

```
In [21]: A*x
```

```
Out[21]: array([[ 2,  6, 15],
                [ 7, 22, 39],
                [17, 38, 69]])
```

```
In [22]: x*A
```

```
Out[22]: array([[ 2,  6, 15],
                [ 7, 22, 39],
                [17, 38, 69]])
```

2.0.5 Operation 5:

This is multiplication of the two marices, AA

```
In [23]: np.dot(A,A)
```

```
Out[23]: array([[110, 134, 164],
                [312, 389, 477],
                [558, 697, 861]])
```

This is element-wise multiplication of A with itself

```
In [24]: A*A
```

```
Out[24]: array([[  4,   9,  25],
                [ 49, 121, 169],
                [289, 361, 529]])
```

3 Accessing elements in an array

```
In [25]: # How to access an element in an array:
         x

Out[25]: array([1, 2, 3])

In [26]: x[0]

Out[26]: 1

In [27]: # Normal parentheses are for callable functions only:
         x(0)


         ---------------------------------------------------------------------------

         TypeError                                 Traceback (most recent call last)

         <ipython-input-27-108f7c32fc4b> in <module>()
           1 # Normal parentheses are for callable functions only:
     ----> 2 x(0)


         TypeError: 'numpy.ndarray' object is not callable


In [28]: # Slicing
         x[0:2]

Out[28]: array([1, 2])
```

4 Dictionary

```
In [29]: # A dictionary can contain a combination of lists and arrays:
         d = {'Your desired label':[1,2,3],'A':A}

In [30]: d['Your desired label']

Out[30]: [1, 2, 3]

In [31]: d['A']

Out[31]: array([[ 2,  3,  5],
                [ 7, 11, 13],
                [17, 19, 23]])

In [ ]:
```