Primer/cheat-sheet on NumPy arrays

First off: NumPy arrays are most definitely sorcery of the highest level. The wizards who made NumPy took great pains to do a lot of stuff, so you don't have to. I have seen some of you treat them like a C array or a list of lists. Please don't do that - it makes the wizards sad! So here are some examples of what these beasts can do:

- Indexing!
  1. When you want to select all entries in M between a and b (up to, but excluding the entry at index b), you can write M[a:b].
  2. If a is the beginning or b is the end, why write it? Just do M[:b] or M[a:]. Imagine what M[:] does.
  3. Your arrays have multiple dimensions? Just separate indexes by commas e.g. M[a:b,c:d] (whenever some uses M[a:b][c:d], a puppy cries)
  4. Only want every second entry? We can do that with M[a:b:2] or M[::2] if we want every second entry from the entire array. Now consider what M[::-1] would do!
- Arrays in all shapes and sizes
  1. If you have an array A, you can use A.shape to get its size along each dimension. A matrix might for instance have the shape (3,3) and a crazy five-dimensional "thing" might be (3,4,2,5,1337)
  2. If you're not satisfied with the shape of an array, you can either just change its shape attribute, use the np.reshape function or A.reshape method.
  3. Just getting the values of an array in a long list is apparently so common that you can use np.flatten just to do that.
  4. Glueing arrays together is possible through functions/methods such as np.concatenate, np.hstack and np.vstack.
- Operators. Have you ever grown tired of using lots of for-loops to do something to some arrays? Chances are there is an operator for that.
  1. +, -, *, / and even ** does exactly what you would expect for two arrays of equal size. So I can add, subtract, multiply and divide two array A and B elementwise with A+B, A-B, A*B, A/B (but be careful with zeroes) or even do exponentiation with A**B.
  2. Logical operators like ==, <, >, <=, >=, "and" and "or" of course works as well. They compare the two arrays elementwise and give you an array of Trues and Falses (boolean array). To test whether all elements of a boolean array is true use np.all(array) or np.any(array) if you just want to see if at least one element is true.
  3. Matrix multiplication is a thing, and we're gonna use it in this course. Fortunately NumPy support that as well. Computing the matrix product between A and B? As simple as A@B. Transposing is just A.T.
  4. Imagine I wanted to subtract a number c from every element in an array A. Is it as simple as doing A-c? Of course it is and it works for every operator above!
  5. Could I do the same with a vector b? In fact I can, but I have to be careful! If b is a row-vector, A-b subtracts b from every row in A. If b is a column vector A-b subtracts from every column. This behaviour is called broadcasting and is equally powerful and dangerous. It can simplify computations or give you complete nonsense (usually when you don't intend to use it)
- Other fun stuff
  - Calling functions on your array: By now you shouldn't be surprised that calling np.sqrt, np.cos or np.min on your array simply.... well.. works.. Also there is an np.sum, take a wild guess what it does.
  - You remember linear algebra? Yeah the wizards took that course as well. The functions np.linalg.solve, np.linalg.inv and np.linalg.eig solves equation systems, computes inverses and finds eigenvalues as you would expect.
  - We will probably expect you to do it manually the first couple of times, but after that you will just use NumPy to compute basic statistical quantities like np.mean and np.var.

  - Did you think broadcasting was crazy? (for the record it is) Well consider now that many built-in functions in NumPy like np.sum, np.mean and np.norm will work along any axis. That means you can get them to take the norm of each row/column, add all numbers, add all rows or columns and compute the mean row, column or matrix (if you have some three dimensional array).

If you want more info look no further than the very extensive and actually not terribly written documentation at **https://docs.scipy.org/doc/numpy/reference/index.html**

There is probably some cool stuff that I forgot to mention, so feel free to comment anything you discover.

Let me just finish off by saying that builtin NumPy functions are either implemented in C or through the LAPACK FORTRAN library (which depending on your system and platform might do matrix operations with BLAS implemented directly in your CPU). This means that when it comes to speed, nothing you implement in python will ever come close to the speed of NumPy.

So throw out those tired old for-loops and start cooking up weird and wonderful matrix/vector operation which will get the job done and send a thankful thought to those who came before us and made sure that np.dot will correctly multiply an N-dimensional array with an M-dimensional array.