And now the fun stuff. Or the intro to the fun stuff.

**The scientific and numerical data structures**

Although Python was not originally designed for scientific data, its community-based and free nature allowed for the development of packages and libraries that handle it nicely & efficiently.

On top of the data type and collections we already defined, we need specific ones that can handle numerical computations…. Efficiently:

So, we need classes and data structures that can handle:

* Large amounts of data, but hopefully little space on storage
* Multiple dimensions arrays
* Mathematical and statistical operations. And efficient ones that can be applied to a whole array.
* And last but not least, as we all learn the hard way, meta data and data attributes

In this workshop, we will use these basic packages of scientific and numerical computing: math, scipy, numpy, pandas, xarray and matplotlib. These packages are also referred as libraries or modules.

**Math**

This is the basic of the basic. Scientific oriented languages have these functions integrated, but for python you have to load them at the beginning of your program.

So, we start by importing the package. We use the import command and the name of the package. We can also give it a nick name if we add “as nickname”. And also, as we saw before we can do from package import class.

In all cases you get the same object.

Now let’s try a little example, run the next code. Note the use of pi, math. And m. all calls to math classes. Also, that format is the method on the string, and floor and degrees is a method of math.

**SciPy**

Scipy is both a library and an ecosystem. Scipy is a library that contains core numerical routines for scientific computing and at the same time, in combination with other packages like numpy and pandas to make a whole ecosystem for scientific and numerical computing.

So, we are not going to study scipy in itself, because it is integrated to the next modules.

**Numpy**

Numpy is the basic scientific module in python. Not the math one, but scientific – because of the amount of data and the structure needed. And this means two big things: the use of multi-dimensional array objects (which basic python doesn’t have), and functions to broadcast and manipulate such arrays.

Let’s execute the code below to import the module numpy and to define two instances of the class numpy array

Print the objects, just to see how they look. Print the type as well.

Now, let’s review the functions. Ones, make an array of 1s of the size 3 rows and 5 columns, and can you guess what the next function does? A range is a numerical range into a numerical array, and then it is reshaped with the reshape method into an array of 3 rows and 5 columns. Note the numbers going from 0 to 14.

Now, modify the code to access an element of an array, you use the following sintax. b[row,column], and it does extend to multidimensional arrays. And remember the square brackets and that indices start in zero.

Now, let’s try a couple of attributes and methods of numpy array objects. Try them.

Ndim is the number of dimensions and shape…well, the shape of the array.

Now, the methods. Max, obviously gives you the maximum value of the array. But if you specify the axis (and again this start as the indices from zero), it applies the method to a single axis. Duh. Same for the method sum.

Ok, now let’s try some operations between our arrays. The easiest are the elementwise operations. Try a+b for example. That means each element is added to the corresponding element on the other array. Of course they need to be the same shape.

And indexing. We already know the basic indexing on lists. Numpy is just an extension of that. And just remember that the indices start with zero.

Execute the next code and let’s see what happens. First we have a new numpy array c. this is how it actually looks.

To print the entire second row, we use c[1], no need to add the columns.

To print the first column though we need to use : to indicate all rows.

Now, add code below to get the requested result.

# print the second element in the first row

print(c[0,1])

# print the last two elements of the second column

print(c[-2:,1])

# print the element last column

print(c[-1,-1])

# use the syntaxis c[[r1,r2],[c1,c2]] to print first and last elements of the array

print(c[[0,2],[0,4]])

ok, now let’s talk about **pandas**.

Sometimes, we need a bit more than just a numerical array. Sometimes we do need labels or to combine numerical data with text like a place where a measurement was taken. And this is where the pandas module comes in. its best attributes are the 2-d table data structure and the capacity to index this data (which can be of any type) by either numerical indices or by labels.

Pandas has two main data types, a series and a dataframe. The series is like a list, but with indices that can be strings or labels.

Series are used to define dataframes, which are tables in which the columns are those series.

Lets give it a try. Let’s execute the next cell. First importing the Pandas module, and then defining a series object called s, based on a range 0 to 4, and giving them indices a to e.

Now, lets use this one and other series to build a pandas dataframe. First a dictionary is define, with keys being the column labels, and items being the series. Finally a data frame is created with the function DataFrame.

Might seem complicated right now, but it is basically creating something similar to a excel sheet, or a csv file. Which by the way is what dataframes are very useful for, in case you have excel or csv data file you want to read (or save).

Another way to create a dataframe is from a 2-d numpy array. Just giving the labels for indices and columns.

Finally, to access data in the dataframe can be done as attributes or by index. And you could just add column indexing as dictionaries.

We’re not going into details of pandas, but we wanted to talk about it because the next package xarray builds on numpy and pandas. It kind of takes a pandas dataframe – a labeled table – and make it multidimensional as a numpy array. Sort of.

We are going to be really going into xarray in the next section. But just a as brief intro, xarray is a sophisticated scientific computing module. Its objects models netcdf file formats, in the sense that allows multiple data to be combined together and these data can be multidimensional and indexed by labels (like lat, lot, time, or anything else), not numerical indices. And it can also have metadata and attributes. You can see now why we like it.