Numpy cheat sheet

HowTo:

- This cheat sheet is supposed to give a short broad overview over important numpy functions and provides hyperlinks to the full documentation of each function. If you are completely new to numpy you might want to check out **this** short tutorial

General array properties:

- ndarray.shape Returns the shape of given array (as **tuple**). Useful for debugging and to create new arrays of this shape.
- ndarray.size Returns the product of the array dimensions (as **scalar**). Might be useful for reshaping.

Creating arrays:

- np.zeros(shape) Creates an array with specified shape (**tuple**) filled with zeros
- np.ones(shape) Creates an array with specified shape (**tuple**) filled with ones
- np.zeros_like(array) Creates an array with the shape of the inputarray (**ndarray**) filled with zeros
- np.arange(start, stop, stepsize) Creates an array by counting from start until stop with specified stepsize (all **int**) start and stepsize are optional
- np.linspace(start, stop, numelements) Creates an array by creating as many elements as numelements between start and the stoppoint (all **int**)
- np.meshgrid(x1, x2, ..., xN) Creates N N-D arrays, each containing a coordinate grid for the respective dimension, from N 1-D coordinate arrays.

Array handling:

- np.reshape(array, newshape) Reshapes the array to specified newshape (**tuple**)
- np.vstack(tuple) Stacks two arrays (as **tuple**) vertically (horizontal equivalent: np.hstack(tuple))
- np.concatenate((array1,array2,...), axis = 0) Concatenates arrays with exact same shape - except for the dimension of "axis".
- np.tile(array, numberofrepetitions) Creates an array by repeating the specified array numberofrepetitions times (**tuple**)
- ndarray.T Transpose the numpy array

Calculating with numpy arrays:

- np.dot(array1,array2) Dot product of two arrays
- Elementwise multiplication → array1 * array2
- All other mathematical operations
- Masking: "array < scalar" returns a boolean mask of where all array values are smaller then the scalar

Array indexing

- $\underbrace{[2}_{\text{start}}\,:\,\underbrace{8]}_{\text{end}}$
  start|end
- [2 : 8 : 2]
  start|end|stepsize → can be used for subsampling
- [2 : 8 , :]
  start|end|2nd dimension
- [... , −1]
  all other dims|last dimension
- Negative values can be used to walk backwards or to access the last elements of an array
- [:,1] returns an array with dim 1 while [:, 1 : 2] returns the same array with dim 2

Statistics

- np.amax(array)/ np.amin(array) calculates the max/min value of an array
- np.argmax(array)/ np.argmin(array) returns the index of the max/min value of an array
- np.mean(array,axis)/ np.std(array,axis) calculates the mean/standard deviation of an array along specified axis

Important functions from other libraries

- scipy.signal.convolve(array,kernel)/ scipy.signal.correlate(array,kernel) N-dimensional convolution/correlation of an array with specified kernel. You might want to use np.pad(array,pad_width) to resize your array after convolving
- Use pickle to save (pickle.dump(data, open(filename, "wb"))) or load (pickle.load(open(filename, "rb"))) any kind of object

Other pointers:

- Setting up python via (Ana-/Mini-)conda for Windows/macOS/Linux
- Setting up a virtual environment Recommended!
- Pycharm is an easy to use IDE for python. Can be assessed in the cip-pool by typing <addpackage pycharm> in the terminal