Final Exercise on Python NumPy

Reference: Python summer school Berlin

1. Construct an array from the list: `[1, 2, 3]` and assign it to the variable `"a"`.
2. Caste `"a"` into floats and assign the results to the variable `"b"`.
3. Create an array of int ranging from `0` through `10` (inclusive) and assign it to the variable `"a"`.
4. Create an array containing 7 evenly spaced numbers between `0` and 23 and assign it to the variable `"a"`.
5. Create an array with shape `(2,5,1,5,2,1)` containing only the number `5.0` and assign it to the variable `"a"`.
6. Eliminate all length-1 dimensions and assign it to the variable `"a"`. What is the shape of `"a"`?
7. Reshape the resulting array `"a"` from shape (2,5,5,2) to only 2D and assign it to the variable `"c"`.
8. Starting from `a=numpy.arange(10)` and `b=numpy.arange(5)` create the array `[0,1,2,3,4,4,3,2,1,0]` (**Note**: the variable `"a"` should change).
9. Create an array with shape `(3,3,12,3)` containing floats drawn from a normal distribution with mean 21 and sigma 4.5 (use `numpy.random.normal()`). Assign it to the variable `"a"`.
10. Calculate the mean, std, var, max, min, of `"a"`.
11. Do the same but not on all elements but only on the 2nd dimension.
12. Transform the array such that it is only 1d and assign the result to the variable `"a"`.
13. Remove all values smaller 15 and larger 26 and assign the result to the variable `"b"`.
14. Clip all values smaller 15 and larger 26 to `NaN` and assign the result to the variable `"b"`. (**Hint**: `numpy.clip`) Print `"b"` to see the result.
15. Calculate the sum of `"b"`. It should not be NaN! (**Hint**: `numpy.nansum`)
16. Print the mean of `"b"`. Can you guess why the result is `nan`?

What is NaN and what is its importance in Data Analysis?

NaN stands for Not-a-Number. NaN is an undefined or unrepresentable value, especially in floating-point calculations. For example 0/0 or the square root of a negative number would return an NaN result.

Let's check it out!

a. `Import numpy as np`

`np.inf` and `-np.inf` stand for positive and negative infinity.

b. Print `0 * np.inf` and `np.inf * 0` (either sign on both factors)
c. Print `0 * -np.inf` and `-np.inf * 0`
d. Print `np.sqrt(-1)`
e. Print `np.array([1,0,3])/0`. Note that dividing 0 by 0 gives `nan`.

So, what is its importance? Let's look at the following example:
As is known, Facebook uses the New Feeds to advertise various campaigns. Let's examine a Facebook campaign. Let's say Nestle is giving you a chance to win free

ice cream for a year. Clicking on Nestle's ad on Facebook opens a link to register for the sweepstake. According to Facebook, **Impressions** are the number of times a particular ad is displayed in a single day (so it measures ad delivery). Nestle's ad impressions was different in every day:
`impressions = np.array([0,11000,23000,55000,37000,33000,44500])`
The number of **clicks** (on the ad) per day:
`clicks= np.array([0,100,600,1000,2000,10000,15000])`
Assume you are a Data Analyst in Facebook and your job is to evaluate the effectiveness of this ad. **CTR** is click-through rate, it measures the effectiveness of any advertisements. It's calculated by: `CTR = clicks/impressions`.
As the first step you decided to calculate the mean effectiveness (i.e. the mean CTR).

a. Divide clicks by impressions and assign the result to the variable `"CTR"`.
b. Calculate the mean of `"CTR"`.

The mean CTR is `NaN` because the `"CTR"` array contains NaN or NaNs. Let's test it:

c. Print `np.argwhere(np.isnan(CTR))` to get the indices list of all `NaN` value in `"CTR"`.

So, in real life you can't just go to your boss and tell him that the CTR is nan. You'll have to deal with with the NaNs (and Null, inf and -inf). Later in this course you'll learn to handle missing values in data set, so you can perform calculations on data even some of which are missing.

17. Convert the NaN's in `"b"` to `0.0` (**Hint**: `numpy.nan_to_num`) and assign the result to `"c"`.
18. Save the array `"c"` and load it again (**Hint**: `numpy.savetxt` and `numpy.loadtxt`).

For more 100 Numpy exercises:

https://github.com/rougier/numpy-100