University of Zurich[UZH]

Department of Physics

Scientific Programming with Python

Data Structures Exercises

June 25, 2025

Exercises 1 & 2 from
Stéfan van der Walt

Before you start:
- create a suitable directory for this exercise
- download the zipped material from `http://www.physik.uzh.ch/~python/python/lecture_data/`

Exercise 1: Exploring the NumPy Data Structure Further

This exercise has the goal to explore the NumPy data structure a bit further.
Generate two 2×2 arrays by

```
>>>>>> x = np.array([[1, 2], [3, 4]], order='C', dtype=np.uint8)
>>>>>> y = np.array([[1, 2], [3, 4]], order='F', dtype=np.uint8)
```

which give you a C- and a F-contiguous array, respectively.
Check the order of the elements and bytes in the data by looking at the strings representing the data behind the arrays with `x.tobytes(order='A')` or, easier to read, `[hex(el) for el in x.tobytes(order='A')]`.
The parameter `order` can have values {`'C'`, `'F'`, `'A'`} and specifies the order that should be used when returning the result (`'A'` stands for any and means to use the order of the input array).

Now define these arrays as of data type `uint32`. Consider again the order of the bytes in the data.
Is the order of type Little Endian (least significant byte first) oder Big (most significant byte first) Endian?
What happens if you switch to signed integers (*i.e.* `int32`)?
What happens if you switch the sign of the values in the input array?

Exercise 2: Fancy Indexing with NumPy

Create an $n \times n$ array and try to construct a one-dimensional array containing all the diagonal elements with fancy indexing.

Create a 10×5 array with random numbers between 0 and 1. Construct the (one-dimensional) array returning the values of the array closest to 0.66 for each row using fancy indexing. Do the same for the columns.
Tip: Make use of `np.abs` and `np.argmin`, and check their documentation.

Predict and verify the shape of the arrays resulting from the following slicing operations:

```
>>>>>> x = np.empty((12, 7, 5))
>>>>>> idx0 = np.zeros((2, 9)).astype(int)
>>>>>> idx1 = np.zeros((2, 1)).astype(int)
>>>>>> idx2 = np.zeros((1, 1)).astype(int)
>>>>>> x[idx0, idx1, idx2]
```

Exercise 3: Random Samples in NumPy

Create an array of Gaussian distributed random variables x_{ij} (`np.random.randn`) with dimension 100×1000 (i is the row index, j is the column index). The element x_{ij} should be distributed according to a Gaussian distribution of mean μ_i and standard deviation σ_j, with $\mu_i = i \cdot 10^{-2}$ and $\sigma_j = j \cdot 10^{-3}$.

In a second step, calculate the mean per column and the standard deviation per row. What do you observe?

In a third step, calculate the average per row of all entries that are above 0.7.
Tip: Leverage `np.where` and you will likely have to use a for loop in this last step (Why?).

Exercise 4: Pandas meet Zebras

You find the position data of several zebras (Equus burchelli) in northern Botswana (Source: movebank.org) in the file `ZebraBotswana.txt`.
The data consists of the date and time of the measurement in unix format (*i.e.* seconds since 1970-1-1), the longitude and latitude of the measured position (in degrees) and the number of the corresponding zebra.

Use Pandas to read in the data (`pd.read_csv`).

For each zebra, calculate the distance between two measurements (the distance of one degree in latitude corresponds to about 111.3 km and in longitude at this latitude to about 104.6 km).

Calculate the distance covered by a zebra during each day and then calculate the daily average and its standard deviation as well as the daily minimum and maximum.
Tip: Calculate the day of each measurement by dividing the Unix timestamp by $24 \cdot 60 \cdot 60$ and transform it into an integer. You can store it as an additional column in the data frame using the function `map`. This allows you to group the measurements per day.

Exercise 5: Pandas & Financial Analysis

The file `FinanceData.csv` contains as time series the adjusted closing prices of the six highest-weighted stocks (IBM, Goldman Sachs, 3M, Boeing, Chevron and United Technologies; VISA is not included due to its initial public offering only in 2008) in the Dow Jones Industrial (DJI) index plus the index itself. The first row is the date in unix format.

Use Pandas to read in the data (`pd.read_csv`).

Calculate the daily increase or decrease of the adjusted closing prices in percent.

Calculate the mean and the standard deviation of these changes over the full time range.

Calculate the mean and the standard deviation of these changes on a rolling basis (*e.g.* based on the last 30 days).

Determine the correlation of the movements between the different time series over the full time range.

Determine the correlation of the movements between the different time series again on a rolling basis.

Exercise 6: Swiss Postal Codes

The file `ch_postal_codes_utf8.csv` contains a list of all the postal codes in Switzerland with their coordinates.

Use Pandas to read in the data (`pd.read_csv`).

Which canton has the highest number of postal codes? And which city?

Calculate the distance between every pair of post codes. (The distance of one degree in latitude corresponds to about 111.3 km and in longitude at this latitude to about 76.0 km; or you can leverage the Haversine formula, *cf.* https://en.wikipedia.org/wiki/Haversine_formula).
Tip: Leverage the broadcasting approach. You should be able to do it without any for loop. And be careful: The values in the file are in degrees.

What is the maximum distance between two places? What is the minimum distance if you neglect zeros?

Try to determine the minimum and maximum distance of all the places within a canton or between two different cantons without redoing the distance calculation.
Tip: Use fancy indexing.

Try to repeat the distance caluculation for France (`fr_postal_codes_utf8.csv`). What does (likely) happen?

Exercise 7: Address book application using protocol buffers

You're working on an address book application and are asked to modify the existing .proto file `addressbook.proto` to include the option to store the address of a given person in addition to the other information already described in the .proto file.
First, extend the .proto file to include the new field (with a reasonable type).
Then, use the protocol buffer compiler to compile the updated .proto file.
After that, read the data from the file `person`, update it by adding a value for the address, and save it to a different file.
Finally, read the new file back using Python to cross-check that it actually contains the address you specified.

Additional Exercise: Data Handling

Store the results obtained in Exercises 4, 5 or 6 in one of the discussed ways (Pickle, JSON, SQL database, MongoDB).