Lumicks Python Developer Assignment

Rules

- We give you the assignment on Friday, and you have until Monday morning 10.00am (Amsterdam time, CET/CEST) to submit the assignment.
- Please don't feel obliged to put more than, say, 2-3 hours of work in the assignment. If you can't finish it within that time: no problem. We just want something substantial to discuss with you. If the code doesn't run yet, that's OK. If it's incomplete, that's also OK.
- The assignment should be submitted as a full Git repository. Either push your project up to GitHub, BitBucket, or any repository hosting service that is accessible to us; or compress your local repository, and send it to us by email.
- Feel free to consult any online resources during the assignment. This is not an exam.
- The code must run on Python 3.6 or later. We will use either a Windows or Linux machine to evaluate the assignment; make sure it runs on at least one of these platforms.
- If you use any third-party libraries, make sure they're installable for us through `conda` or `pip`.

What we look for

- Clean, readable code
- Well-documented code (but don't overdo it: documentation should be useful and concise)
- Clear Git commit history

Assignment

Background

One of the instruments we build at Lumicks features a *confocal scanner*. The scanner consists of a laser beam that scans over the sample, and a sensor that measures the amount of light that comes back. Typically, we scan over the sample in a line-by-line fashion, starting at the top-left:

```
START
v
------------->
------------->
------------->
------------->
             ^
             END
```

Now, because of the nature of this scanning process, the data that comes back from the sensor has to be mapped onto the pixels of an image in a special way. Say we're receiving sensor data at a rate of 1,000 samples per second. The first 3 samples might correspond to the top-left pixel in the image; the next 3 to the second pixel; etc. Additionally, we may want to ignore some of the samples altogether: when the scanner moves from one line to the next, it takes some time to 'brake' and reverse direction, during which time we'd rather ignore all the light that comes back completely.

In the data files that we write, we have decided to store the full sensor data anyway, but we write an additional channel with data that encodes how to process it:

- Samples that should be ignored are marked with a `0`. These samples are not part of any pixel in the image.
- Samples that combine to form one image pixel, are marked with `1` for the first (N-1) samples ("accumulate"), and `2` for the last sample in the image pixel ("end-of-pixel"). So for three samples that, added together, form one pixel, we would see mapping values of `1 1 2`.

So say we have an image that consists of a single line of 3 pixels. The following dataset:

```
sensor:   1  2  1  3  4  2  2  1  0  1  0  0  3  1
mapping:  0  0  1  1  2  1  1  2  1  1  2  0  0  0
```

would then map onto the following pixel values: 8, 5, 1.

The goal

Solve the following problem:

- Load an HDF5 file containing the data streams described above. (See below for more information on this file format, and how to read it.)
- Reconstitute an image from each data file, and store it as a PNG file.
- Take the middle horizontal line of the image, fit a Gaussian to the pixel values, and output the σ (sigma) parameter of the Gaussian (a measure for its width).

The idea would be to write a module with a user-facing API. And then create a Jupyter Notebook which will serve as a tutorial for users to learn how to use this functionality. You are free to decide which parts you would like to put in the notebook and which parts you wish to include in the module.

About the HDF5 data file

We provide a couple of example data files to you in a format called HDF5. In case you haven't encountered such files before: it's an open format for structured binary data, widely used across academia and industry.

Reading HDF5 files is easy enough in Python, fortunately, if you use the *h5py* package. For documentation on *h5py*, see https://docs.h5py.org.

The HDF5 dataset for this assignment is structured as follows:

- `data/sensor` (i.e., the `sensor` dataset in the `data` group): 1D array of all the samples read from the sensor during the scan.
- `data/mapping`: 1D array of all corresponding mapping values, as described above.

Additionally, the HDF5 file contains some attributes with metadata:

- `data/image_width`: number of pixels along the horizontal axis of the image.
- `data/image_height`: number of pixels along the vertical axis of the image.