"Pollution in Graz"

Homework Assignment 3

7. November 2018

Original data: https://luftdaten.info/

1 Tasks (25pt)

The topic of this assignment is to visualize the sensor data and match it to its geographical origin on a map. You can find a reference output (.html Download) in the Wiki[1]!

1.1 Functions (5 pt)

The goal of this exercise is to learn how to use visualization tools and learn how to structure functions. This means that the majority of your code should be implemented as functions. The only exception is when parsing the command line parameters (extracting the parameters). Checking to see if the parameters are present (e.g. if the input file is present) however should be implemented in a separate function. Your program then should consist of a number of functions that can be called with appropriate arguments. As a guideline you should implement every individual functionality in a separate function. Whenever you find the same code twice (or even more frequently) then it is a good sign that this should also be implemented as a function. Your program should contain at least 10 usefully constructed functions. For every function you will receive 0.5p (for a max of 5p).

1.2 Command Line Parameters (3p)

Your program should be able to work with a variety of different command line parameters. Use the `argparse` package[2] to implement this functionality. Implement the following parameters:

`-i --input:` Path to the input file (the dataset)

- Data type: String
- Optional: No

`-o --output:` Path to the output file

- Data type: String
- Optional: No

`-s --sensors:` Names of the desired sensors

- Data type: List of integers
- Optional: Yes
- Default value: `[ ]` (empty list)

[1] https://palme.iicm.tugraz.at/wiki/Info1BM#.C3.9Cbungs_Assignments
[2] https://docs.python.org/3.7/library/argparse.html

`--width:` Width of the map

- Data type: Integer
- Optional: Yes
- Default value: 800

`--height:` Height of the map

- Data type: Integer
- Optional: Yes
- Default value: 800

`--zoom:` Zoom of the map

- Data type: Integer
- Optional: Yes
- Default value: 13

For all parameters that are given as strings, make sure to work case-insensitive. Make sure that every possibility of lowercase and uppercase combinations works!

1.3 Basic I/O Checks (1p)

Check to make sure that the file related parameters are valid. Implement the following checks:

(i) Verify that the input file exists and is indeed a file.
(ii) Verify that the desired output path exists.

Should one of the tests not be passed provide an error message and end the program with `exit()`. Alternatively you could also raise an Exception[3] raisen.

1.4 Parsing the Dataset

1.4.1 Parsing(1p)

The dataset consists of the output from the last assignment. You may create this yourself or use the reference version found in theWiki [4]. Parse the dataset into your program. Eventually you can use the same code as in the last assignment. However, you have to pay attention since the columns (header) may have changed!

Parse the dataset line by line and save the values. The command line parameter `--sensors` works as a filter. Skip the rows with sensors (IDs) that are not explicitly wanted. Should the parameter note be present (set to the default value = []) then save all sensors.

1.4.2 Verification (1p)

Verify the dataset once again after it has been parsed and read into the program. Check to make sure that all desired sensors have been found. Should a sensor have no entries, then give a notice/warning to the user. Your program should however continue to work. The only exception is when all sensors have not been found! Then you should end your program with `exit()` or an Exception.

For the next preparation step you should verify that the order of the values for every sensor is in chronological order. This is identical to the requirements of the last assignment. However, this time make sure to use the `numpy.argsort()`[5] function!

[3] https://docs.python.org/3/tutorial/errors.html
[4] https://palme.iicm.tugraz.at/wiki/Info1BM#.C3.9Cbungs_Assignments
[5] https://docs.scipy.org/doc/numpy-1.15.4/reference/generated/numpy.argsort.html?highlight=argsort#numpy.argsort

1.5 Plotting on the Map (5p)

To create a map use the `folium` package[6]. Make sure to learn more about the package and when necessary install it (`pip install folium`). When you create a map there are some necessary parameters. The width and height of the map (in pixels) are to be used as parsed in the command line parameters `--width` and `--height` and `zoom_start` for the parameter `--zoom`. As (`tiles`) use `"OpenStreetMap"`. You can write this directly in your program. Lastly, your map also needs a center (location). For this step calculate the median of the different lengths and widths of all the desired sensors.

For every sensors create a marker (`folium.Marker`) and place it on the map. Save thie map at the end of your program with u `--output` and the name `map.html`.

1.6 Bokeh Popups on Markers (9p)

For every marker on the map (for every desired sensor) create a pop-up which when clicked shows up when clicking on the marker. For this step use the Bokeh package[7]. The plot should look like your outputs in the last assignment but with edited values (smooth). Examples can be found in the Wiki. The choice of colors is free for you to choose but should work with the standards for people that are colorblind. In the provided code for the last assignment the Matplotlib function `fill_between()` was used. Bokeh with Band[8] has similar functionalities. In addition to Band plot the values also with `line()`. If this step is not implemented the axes will not be adjusted correctly.

Make sure to use the Bokeh HoverTool[9] in order to get a Mouseover Effect with the pop-ups. This should (like in the reference) show the date, P1 and P2 values and the current mouse position. Make sure that the date is displayed as a date (and not a large number)! When creating HoverTools make sure to set the parameter `mode` to `"vline"` and the parameter `point_policy` to `"follow_mouse"`. This will make your result look like that of the reference.

In order to make sure that the HoverTool functions correctly connect only on the line from P1 (that you have plotted in addition to Band). This can be achieved with the HoverTool parameter `renderers()`.

The height and width of the Bokeh Plots should be calculated as follows:

- Height = `--height` Parameter / 4
- Width = `--width` Parameter / 2.5

When you create the Folium Popup you have to input the values for height and width again. Make sure to set these values to the values of the Bokeh plots +25 in order to prevent scrollbars.

You can use the following code to create the Folium pop-up using the Bokeh plot. This implementation must then additionally be connected with the marker.

```
from bokeh.resources import CDN  # You will need these imports
from bokeh.embed import file_html

# bokeh_plot is the variable where you have stored the plot you want to display in the popup
# replace <width> and <height> with your values. Make sure they are integers!
popup = folium.Popup(
    folium.IFrame(
        folium.Html(
            file_html(
                bokeh_plot,
                CDN
            ),
            script=True
        ),
        height=<height>,
        width=<width>)
)
```

[6] https://github.com/python-visualization/folium
[7] https://bokeh.pydata.org/en/latest/
[8] https://bokeh.pydata.org/en/latest/docs/user_guide/examples/plotting_band.html
[9] https://bokeh.pydata.org/en/latest/docs/user_guide/examples/tools_hover_tooltip_formatting.html

Hint: To test your program you can save each indvidual Popup plot. However, make sure to remove these when you uploading your assignment.

1.7 Bonus: Snowball Marker (3p)

Create an additional marker on the map. This should mark any place on the map where you had a snowball fight over the Christmas holidays. It does not necessarily need to be in Graz! When possible, include an evidence photo in the Pop-up. Should there be a lack of snow then you can create cookie markers that show where you ate Christmas cookies. You can use the Websize https://www.latlong.net/ to calculate the length and width of each place.

2 Restrictions

- Do not add any bloat to your submission
- Use built-in functions where possible
- Only use/import explicitly allowed packages
- Do not use any command line arguments
  - You may use extra parameters. However, your program should also work without their use!

2.1 Allowed Packages

In general: `argparse`, `copy`, `collections`, `os`, `sys`
Math: `numpy`, `pandas`
Visualization: `matplotlib`, `seaborn`, `bokeh`, `folium`

3 File Headers

All Python source files have to contain a comment header with the following information:

- Author: – Your name
- MatNr: – Your matriculate number
- Description: – Purpose of the file
- Comments: – Any comments as to why if you deviate from the task or restrictions

Please copy and paste the example and change its contents to your data. (Depending on your PDF-viewer you may have to fix the whitespaces!)
Example Header:

```
##########################################################################
# Author:      [FIRST NAME] [LAST NAME]
# MatNr:       [MAT NR]
# Description: [SHORTDESCRIPTION]
# Comments:    [ANY RELEVANT COMMENTS.
#              CAN BE MULTILINE]
##########################################################################
```

4 Coding Standard

This lecture follows the official PEP 8 Standard[10]. it defines basic formalities as to how your code should look like. In particular, please look at the following aspects:

Language. Programming is done in English. This is to ensure someone else at the other end of the world can read your code. Please make sure all sources you submit are thus written in English. This covers both variable names and comments!

Spaces, not tabs. Indent your files with 4 spaces instead of tabs. PEP 8 forces this in Python 3 (whilst allowing more freedom in Python 2). Most editors have an option to indent with 4 spaces when you press the tab button!

Descriptive names. Use descriptive names for variables where possible! Whilst a simple i can be enough for a simple single loop code can become very messy really fast. If a variable has no purpose at all, use a single underscore (_) for its name. Should you be uncertain if a name is descriptive enough, simply as a comment describing the variable.

120 **characters line-limit** PEP 8 suggests a maximum line length of 79 characters. A different established limit proposes 120 characters. In this lecture, we thus use the wider limit to allow for more freedom. The maximum number of characters is 120 including indentations! Note that this is also applies to comments!

5 Automated Tests

Your code will be tested by an automated program. Please make sure your submission follows all the restrictions defined in this document. In particular concentrate on the predefined names for functions and variables!
If your submission fails any of the automated tests, we will look into your code to ensure the reason was not on our end and to evaluate which parts of your code are correct and awards points accordingly.

[10] `https://www.python.org/dev/peps/pep-0008/`

6 Submission

6.1 Deadline

15^{th} of January 2019 at 23:59:59.

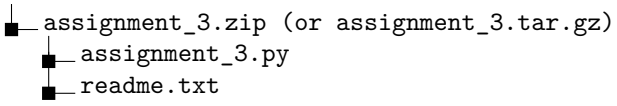
Any submission handed in too late will be ignored with the obvious exception of emergencies. In case the submission system is globally unreachable at the deadline it will be pushed back for 24 hours. If for whatever reason you are unable to submit your work, contact your tutor *PRIOR* to the deadline.

6.2 Uploading your submission

Assignment submissions in this course will always be archives. You are allowed to use .zip or .tar.gz formats.
In addition to your Python source files, please also pack a *readme.txt*. The file is mandatory but its contents are optional. In this file please write down how much time you spent on the assignment and where you ran into issues. Your feedback allows for immediate adjustments to the lecture and tutor sessions.
Upload your work to the Palme website. Before you do please double-check the following aspects:

- File names and folder structure (see below)
- Comment headers in every source code file
- Coding standard upheld

6.3 Your submission file



The dataset and output files should NOT be uploaded!