Code Documentation

- General Structure of the Code:
    - The first cell contains all packages to be loaded.
    - The second cell contains all code that needs to be run only once, which is the "stuff for ad on-the-fly download" and all functions.
    - The third cell contains the code that is actually executing the parsing.
    - The last cells are some trial code to test specific files to adjust the functions. I always copied the function I was working on down there, adjust it according to what I wanted and then replace the actual function in cell 2.
- How the overall code works:
    - Every outlet has its function called "parse_*outlet"*.
    - There is the overall function "parse", which delegates the content to the correct function. Outlets can be added to the code by writing a new function for the outlet and adding it to the "parse"-function.
    - The same is true for the "change_date"-function along with the "change_date_*outlets*"-functions.
- How the third cell works:
    - There is one loop for each month (starts in line 662).
    - There is one loop for every outlet (starts in line 672).
    - There is one block for regular data processing (starts in line 674).
    - There are two blocks for different on-the-fly-scripts (so that the data is downloaded again) (start in lines 723 and 768).
- How to run the code:
    - At the beginning of the third cell, there are variables to be specified:
        1. "year" (line 622) (takes only one value),
        2. "months" (line 623) (a list, you can give it several values, for the values to fill in, please see the comment in the code)
        3. "onthefly" (line 624) (True/False; when True: the data is downloaded again)
        4. "outlets" (line 671) (a list, possible values are equivalent to the names of the folders)
    - Lines 630 to 659 are used to direct the script to the correct folder. This needs to be adjusted according to the machines' folder-structure.
- What to look at when testing code:
    - After letting a code run for a month I first checked the "fulltext"-file. Here you can get an easy overview if there are still files for which the code produces no output (which means, the homepage-structure is different).
    - Then I checked randomly files and see if special parts of the text are included, such as bold or italic text, second headings, text with links behind etc. If missing, add xpaths for it to the code. To make it easier to track later I always had a comments list to remind me what xpath does what and in which file I found the error.