Mandatory 1: Log Analyzer

02807 Computational Tools for Data Science

Submission opens:	**08:00 Saturday, September 22, 2018**
Deadline:	**20:00 Sunday, September 30, 2018**
Individual:	**This assignment must be completed individually** (see collaboration policy on course page)

Exercise

In this mandatory assignment, the goal is to make a log file analyzer that can parse and analyze log files from webservers in python. The log files will be in the Common Log Format described on `https://en.wikipedia.org/wiki/Common_Log_Format`. Each line in such a file corresponds to a request to a webserver.

To solve this exercise, you must download the file testdata.zip from the course page. In here there is a file called LogAnalyzer.py where we have predefined a class LogAnalyzer with a number of currently (almost) empty functions. For each function, there is a comment in the file describing what it is supposed to do.

We recommend you to implement the functions one-by-one in the order described and ensure the function works before you continue to the next one.

If you are in doubt about exactly what some of the functions are supposed to do, you should look at the sample test data and see if you can figure it out that way before you ask for help. The file sample_test.py calls the different functions of the LogAnalyzer. When you run this with your solution, you are supposed to get the output shown in expected_output.txt.

Submission You must submit your solution on CodeJudge **before** the deadline (any submissions made after the deadline will **not** be counted). Your latest submission before the deadline will be used for grading thus you **must** ensure your best solution is uploaded as the latest. Submission on CodeJudge will not be available before the submission opens time stated in the top.

Grading In order to obtain full points for this assignment, you must implement all the described functions in a correct and efficient way. From submission opens until the deadline you can submit your code on CodeJudge and see how well it scores. An entirely correct solution gets 100 points. Partially correct/efficient solutions will get a score between 0 and 100. On CodeJudge you can see the results of all the tests, but all except the already provided sample test data will be hidden.

Correctness. A subset of the tests will be relatively small and only ensure your functions return the correct values. To solve these, you do generally not have to have clever solutions.

Efficiency. Another subset of the tests will be of a size such that "naive"[1] solutions (like scanning all log lines over and over again for each operation) will not work within the time limits. In other words, to solve these tests, you may have to come up with more efficient solutions. (This is mostly relevant for the functions: parse, numberOfStatusCode, countRequestsInTimeRange, resourcesWithQueryParam, and resourcesWithAllQueryParams)

The score you obtain in this mandatory assignment will count towards your final grade.

[1] `https://wcipeg.com/wiki/Naive_algorithm`

Competition If you are up for an extra challenge, there will be a competition among those interested to make the fastest possible solution. This is completely voluntarily, and will **not** count towards your final grade – instead there will be a small prize for the fastest solution.

The problem in the competition is exactly like the mandatory part and the tests will be of the same type (except there may be more requests in the log file and more queries in the tests), so everybody can join the competition with their solution without extra effort. We will simply sum the number of solved test cases, and in the case of a draw the sum of the running time of all passed tests will decide who got the fastest solution.

To join the competition, simply also submit your best solution to the "Competition" exercise on CodeJudge (you must still submit to the mandatory exercise as well). A live scoreboard will show the current standings on CodeJudge.