Lab 08 - Regular expressions

Script Languages

Learning goals

1. Use regular expressions.

Exercises

1. Preparation

1. Download the ini config file and the log file from ePortal.

2. Reading config file

1. Implement a dataclass for storing the content of the config file. Prepare a storage for values from particular sections of the config file
    - [Display] – a dictionary with display settings,
    - [LogFile] – a variable with filename,
    - [Config] – a dictionary with logging settings
2. Develop a function that reads configuration from config file. Exit application if the config file is not present.
    1. Read the content of the config file. Use regular expressions to analyse each line: recognize if it is section header (e.g. [Display]) or section content in the form of <parameter>=<value> (e.g. filename=web20200221.log).
    2. If any of the settings in the default lab6.config file is not present set it to arbitrarily chosen default value.
3. During the runtime read and process the content of the config file and instantiate the class from task 2. If required pass the created instance as a configuration to further functions as a parameter. Dynamically configure logging.

3. Reading log file

1. Develop a function that reads the content of the log file into memory. If file does not exist exit the application with proper message. Return a data structure containing all log lines.
2. Develop a function that analyses the content of the log file. Parse the content of each line using a regular expression. Use regular expressions to extract: IP address,

timestamp, HTTP request header, HTTP status code, size of the response. Return a list containing instances of classes representing extracted data.

3. Develop a set of functions for printing all requests sent from the given IP subnet.

    1. The IP should be passed as a parameter passed be the user. Use `argparse` (https://docs.python.org/3/library/argparse.html) module to get the required value.

    2. IP mask length is evaluated in the following manner: your index number modulo 16 plus 8 (e.g. student's index number: 224538, IP mask length: 224538 % 16 + 8 = 18). The index number should also obtained as a parameter passed be the user.

    3. Every number of lines (defined in the configuration file) stop and ask user to press Enter key.

    4. Don't put all the code in one function. Every function should have only one, single responsibility, e.g. extract a code to check if IP address belongs to the given IP subnet into a separate function.

4. Develop a set of functions to count a total number of bytes sent in response to requests of type filter defined in configuration file. Use regular expressions to identify requests. Print type of a request and a total number of bytes. Separate fields by the separator defined in the configuration file.

5. Install one of the code style checkers/formatters:

    1. `pycodestyle` (https://pypi.org/project/pycodestyle/)

    2. `black` (https://github.com/psf/black)

6. After finishing all other exercises run the chosen formatter command for every crated file. Create a separate file to resolve encountered problems.

4. Extended version

1. Develop a set of tests to verify your implementation.

2. Demonstrate the test coverage using `pytest-cov` (https://pypi.org/project/pytest-cov/) library.