Lab 05 - Files and dictionaries

Script Languages

Learning goals

1. Read and write text files.

2. Use dictionaries to store and retrieve data.

3. Create a new class using inheritance.

Exercises

1. Introduction

1. Download the example Apache access log file and save it in the project folder.

2. Review the page https://httpd.apache.org/docs/2.4/logs.html to learn the structure of the Access Log.

3. Write run() function that will call the remaining functions in this assignment. Call this function only if the file is run directly.

2. Using dictionaries

1. Write a function `read_log()` that:

    1. Reads the content of the log file.

    2. For each row creates an instance of a suitable class from the previous assignment. Store these instances in lists, in a dictionary modeling the whole log. Which one of the log entry fields should be used as a key?

    3. Return the dictionary as a result.

2. Write `ip_requests()` function. Create a class that will allow you to store the number of requests for the given IP address. Return its instance as a result.

3. Write `ip_find()` function to find IP addresses that issued:

    1. the biggest number of requests (if function is called as `ip_find()` or `if_find(most_active=True)`)

    2. the smallest number of requests (if called as `ip_find(most_active=False)`)

Remember that there could be many IP addresses with the same number of requests.

4. Write function `longest_request()` that will return the longest request line (e.g. `GET /index.html`) along with its IP address. If there are many such addresses, return arbitrarily chosen one.

5. Write `non_existent()` function that will return all request strings with HTTP result code „404 - Page not found". In the result each request string should occur once, even if there are many entries with the same request.

3. Extended version

WARNING - all the snippets below are not fully functional and only serve as a brief illustration of the desired outcome.

1. Create a class `AccessLog` that could be used instead of a dictionary in task 2.1. Implement a function which creates its instance taking a log file as a parameter. Use appropriate magic methods to make instance of `AccessLog` class behave like a dictionary e.g.:

```
>> access_log = get_access_log("log_file.txt”)
>> access_log['27.202.53.132']
```

2. Use appropriate magic methods in the class created in the task 2.2 to make it iterable. Implement `__len__` method.

```
>> ip_requests = IPRequests(...)
>> for ip in ip_requests:
>>    print(ip)

>> len(ip_requests)
>> 6
```

3. Modify functions from tasks 3-5. Implement them as methods within appropriate classes.