# ← Project review - APP1 Bootcamp. Day08

| | |
|---|---|
| Type of project | Individual |
| Duration | 30 min |
| Passed Peer Reviews | 0/3 |

## Git project

ssh://git@repos-ssh.21-school.ru:2289/students/Python_Bootcamp._Day_08.ID_87714...

Copy link Open

## Student

greita@student.21-school.ru level 9

## About

### Introduction

The methodology of School 21 makes sense only if peer-to-peer reviews are done seriously.
Please read all guidelines carefully before starting the review.
- Please, stay courteous, polite, respectful and constructive in all communications during t
his review.

- Highlight possible malfunctions of the work done by the person and take the time to discuss and debate it.
- Keep in mind that sometimes there can be differences in interpretation of the tasks and the scope of features. Please, stay open-minded to the vision of the other.
- If you have not finished the project yet, it is compulsory to read the entire instruction before starting the review.

### Guidelines

- Evaluate only the files that are in src folder on the GIT repository of the student or group.
- Ensure to start reviewing a group project only when the team is present in full.
- Use special flags in the checklist to report, for example, an “empty work” if repository does not contain the work of the student (or group) in the src folder of the develop branch, or “cheat” in case of cheating or if the student (or group) are unable to explain their work at any time during review as well as if one of the points below is not met. However, except for cheating cases, you are encouraged to continue reviewing the project to identify the problems that caused the situation in order to avoid them at the next review.
- Doublecheck that the GIT repository is the one corresponding to the student or the group.
- Meticulously check that nothing malicious has been used to mislead you.
- In controversial cases, remember that the checklist determines only the general order of the check. The final decision on project evaluation remains with the reviewer.

## Main part

### Exercise 00 - I Know Kung Fu

- File "fight.py" is present

- Agent's health is only decreasing when it is hit


### Exercise 01 - A Squid On A Stick

- Files "crawl.py" and "server.py" are present
- When started, server starts listening on port 8888
- Both server and client code are asynchronous and are using async/await paradigm for I/O
- Client accepts a list of URLs as command line arguments (or via text file with a list of URLs)
- Server receives JSON list of URLs via POST request to HTTP endpoint `/api/v1/tasks/` and i

mmediately responds with 201 Created
- On task creation a task object returned by server includes field "status" set to "running" and a random "id" based on UUID4
- Server asynchronously processes URLs and collects HTTP codes from responses
- Client keeps periodically (no more frequently that once per second) querying endpoint `/api/v1/tasks/{received_task_id}` until server returns "status" equal to "ready"
- Server returns a list of HTTP codes that can be matched with corresponding URLs
- Client prints out tab-separated HTTP response code and corresponding URL for every entry


### Exercise 02 - DejaVu

- File "server_cached.py" is present
- URL crawling result is cached in Redis and is returned from it instead of querying the same URL again
- For every incoming domain a view counter is increased in Redis, and it works properly if the same URL is submitted multiple times in one request or in multiple subsequent requests
- All the code to work with cache is asynchoronous and is using async/await paradigm for I/O

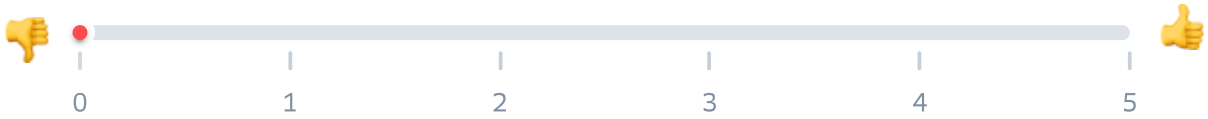

## Bonus part

### Exercise 00 - Bonus part

- Neo successfully fights off multiple agents at once


### Exercise 02 - Bonus part

- Server code includes specific logic for cleaning up entries from cache if they are not submitted again during the configured timeout




## Feedback

### Fails

Invalid compilation | Forbidden functions | Cheat | Code style | Crash | Empty work | Leaks

### Comment

Leave a comment...

Review