

# Prerequisite

## 1.Third party dependencies

Since this is a server-client architecture, there are five third party libraries needed as follows:

- (1)tornado – for server side (support restful api)
- (2)pandas – for server side(read from database and do the statistics summary)
- (3)sqlalchemy – for server side(connect, read, write to database)
- (4)requests – for client side(request data from server side)

### On Windows:

(1)To install the third party libraries globally make sure Administrator Access obtained and run:  
`python3 setup.py install`

### On Linux and Mac:

(1)activate a virtual python3 environment through `virtualenv`, then run:  
`python3 setup.py install`

a. Please start the server first by:  
`python3 server.py`

b.Then run the client by:  
`python3 a3.py`

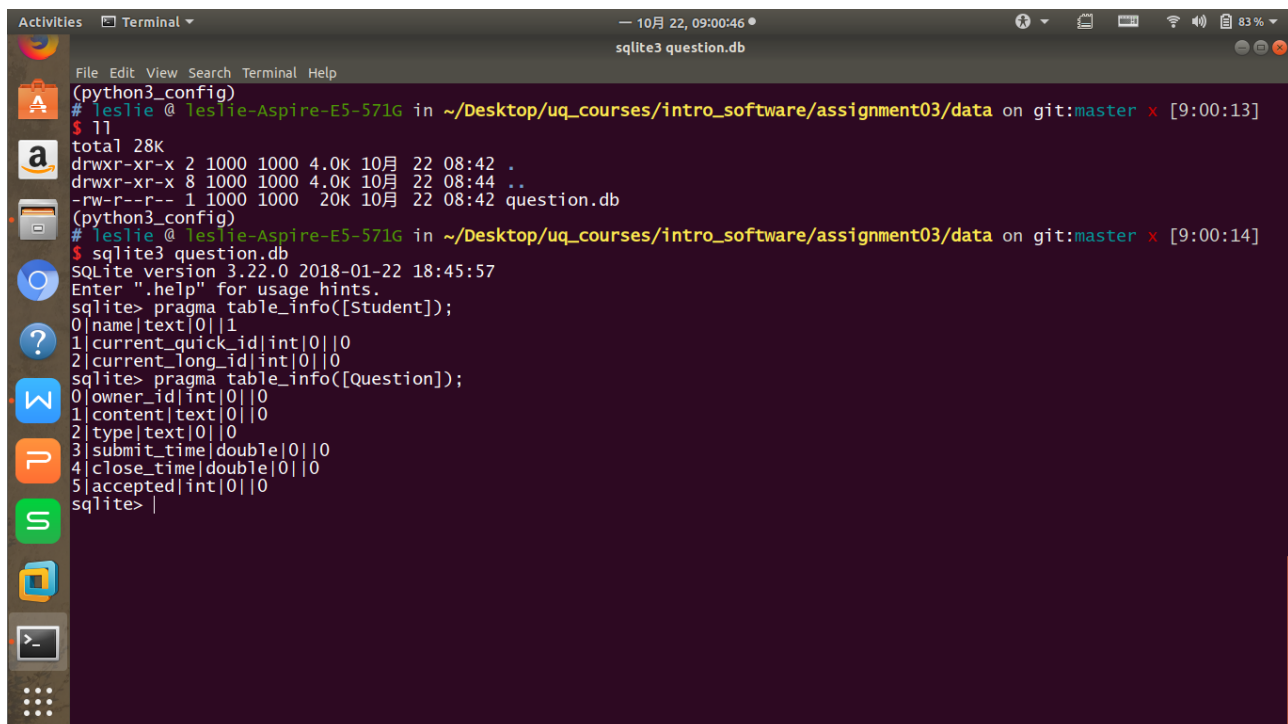
c.(optinal) there is a one-step script that starts the sever and client which is the `start.py`, simply run:  
`python3 start.py`

## 2.The restful apis for server are as below:

- (1) URL:/data  
METHOD: GET  
URL Params:  
Required:  
question\_type[str]

```
        example: question_type='quick'
(2) URL:/accept
    METHOD: POST
    Data Params:
        Require:
            question_type[str]
            name[str]
        example:
            {
                question_type:'quick',
                name:'Leslie'
            }
(3)URL:/cancel
    METHOD:POST
    Data Params:
        Required:
            quetion_type[str]
            name[str]
        example:
            {
                question_type:'quick',
                name:'Leslie'
            }
(4)URL:/submit
    METHOD:POST
    Data Params:
        Required:
            name['str']
            content['str']
            question_type['str']
        example:
            {
                name:'Leslie',
                content: 'MyPytutor',
                question_type:'long',
            }
(5)URL:/statistics
    METHOD:POST
    Data Params:
        Required:
            question_type['str']
        example
            {
                question_type: 'long'
            }
```

### 3. Use the sqlite3 as the backend database with 2 tables

A terminal window titled 'Terminal' with a dark background. The top bar shows the date '10月 22, 09:00:46' and battery status '83%'. The terminal shows a file listing for a directory containing 'question.db'. Then, the 'sqlite3 question.db' command is executed, showing the SQLite version and a prompt. Two 'pragma table\_info' commands are used to inspect the 'Student' and 'Question' tables, displaying their column names, types, and constraints.

```
File Edit View Search Terminal Help
# leslie @ leslie-Aspire-E5-571G in ~/Desktop/uq_courses/intro_software/assignment03/data on git:master x [9:00:13]
$ ll
total 28k
drwxr-xr-x 2 1000 1000 4.0K 10月 22 08:42 .
drwxr-xr-x 8 1000 1000 4.0K 10月 22 08:44 ..
-rw-r--r-- 1 1000 1000 20K 10月 22 08:42 question.db
(python3_config)
# leslie @ leslie-Aspire-E5-571G in ~/Desktop/uq_courses/intro_software/assignment03/data on git:master x [9:00:14]
$ sqlite3 question.db
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> pragma table_info([Student]);
0|name|text|0||1
1|current_quick_id|int|0||0
2|current_long_id|int|0||0
sqlite> pragma table_info([Question]);
0|owner_id|int|0||0
1|content|text|0||0
2|type|text|0||0
3|submit_time|double|0||0
4|close_time|double|0||0
5|accepted|int|0||0
sqlite> |
```

#### Table Student:

name: str, to store the student name  
current\_quick\_id: int, null if student is currently not request for a question, otherwise it would be the rowid of the Question table  
current\_long\_id:int, null if student is currently not request for a question, otherwise it would be the rowid of the Question table

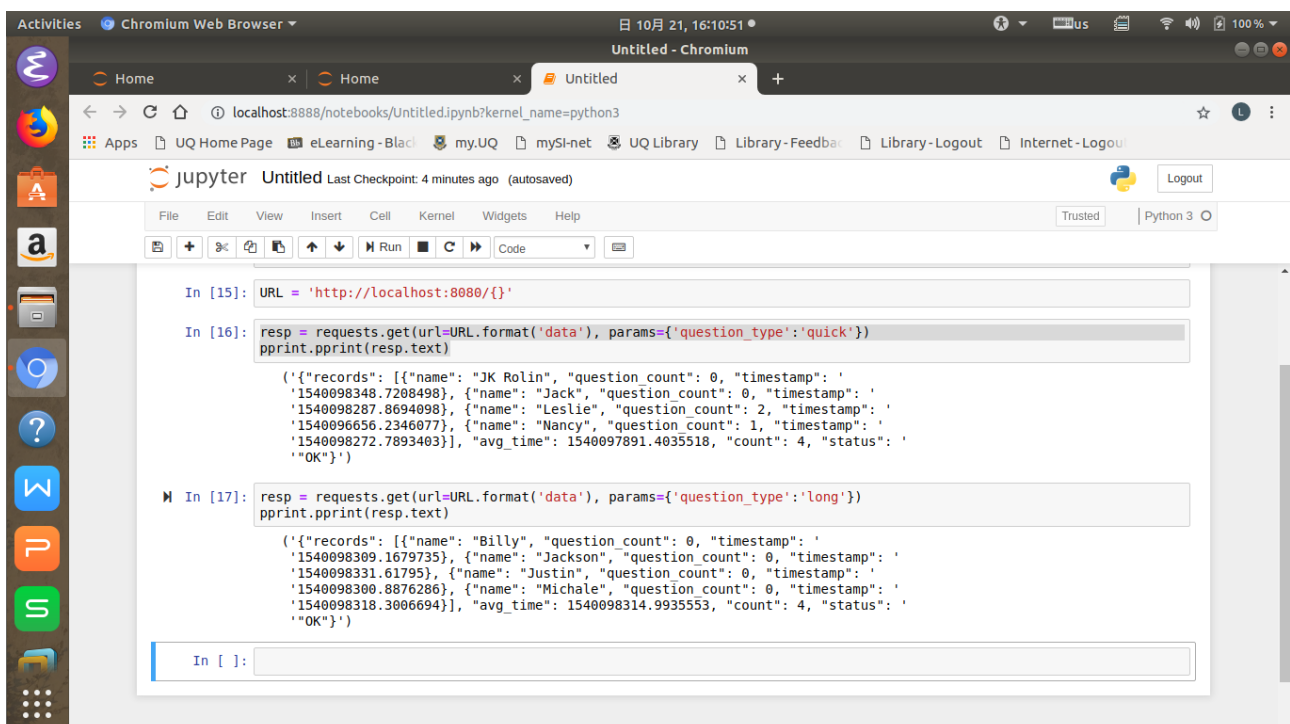
#### Table Question:

owner\_id: int, the rowid of the Student Table, indicates which Student owns this Question  
content: str, the content of the question  
type:str, question type, either quick or long  
submit\_time: double the timestamp when Student submit this question  
close\_time: double the timestamp when this question is accepted, if the question is canceled, the value will be null  
accept: int 0 or 1 when the value is 0 it means this question not yet been accepted it could either be this question is still waiting or been cancel.  
If close\_time is null and accepted is 0, it means this question is still waiting on request.

If the `close_time` is not null and `accepted` is 0, it means this question has been canceled.  
If `accepted` is 1, it means this question has been accepted.

### 3. Testing restful api examples

#### (1) /data



The screenshot shows a Jupyter Notebook interface in a Chromium Web Browser. The browser's address bar displays `localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3`. The notebook's title bar indicates it is 'Untitled' and 'Last Checkpoint: 4 minutes ago (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains three code cells:

```
In [15]: URL = 'http://localhost:8080/{}'
```

```
In [16]: resp = requests.get(url=URL.format('data'), params={'question_type':'quick'})
         pprint.pprint(resp.text)

({'records': [{'name': 'JK Rolin', 'question count': 0, 'timestamp': '1540098348.7208498'}, {'name': 'Jack', 'question count': 0, 'timestamp': '1540098287.8694098'}, {'name': 'Leslie', 'question count': 2, 'timestamp': '1540096656.2346077'}, {'name': 'Nancy', 'question count': 1, 'timestamp': '1540098272.7893403'}], 'avg_time': 1540097891.4035518, 'count': 4, 'status': 'OK'})
```

```
In [17]: resp = requests.get(url=URL.format('data'), params={'question_type':'long'})
         pprint.pprint(resp.text)

({'records': [{'name': 'Billy', 'question count': 0, 'timestamp': '1540098309.1679735'}, {'name': 'Jackson', 'question count': 0, 'timestamp': '1540098331.61795'}, {'name': 'Justin', 'question count': 0, 'timestamp': '1540098300.8876286'}, {'name': 'Michale', 'question count': 0, 'timestamp': '1540098318.3006694'}], 'avg_time': 1540098314.9935553, 'count': 4, 'status': 'OK'})
```

The bottom of the notebook shows an empty input cell labeled 'In [ ]:'.

summary

## Important

Individual assessment items must be solely your own work. While students are encouraged to have high-level conversation about problem they are trying to solve, you must not look at another student's code or copy from it. The university uses sophisticated anti-collusion measures to automatically detects similarity assignment submissions.

### short question

<2 mins with a tutor

Some example of short question

- Syntax errors
- Interpreting error output
- Assignment/MyPyTutor interpretation
- MyPyTutor submission issues

[Request Short Help](#)

An average wait time for 13 minutes ago for 4 students

| # | Name     | Question Asked | Time           |
|---|----------|----------------|----------------|
| 1 | Jack     | 0              | 7 minutes ago  |
| 2 | JK Rolin | 0              | 6 minutes ago  |
| 3 | Nancy    | 1              | 7 minutes ago  |
| 4 | Leslie   | 2              | 34 minutes ago |

### long question

>2 mins with a tutor

Some example of long question

- Open ended questions
- How to start a problem
- How to improve code
- Debugging
- Assignment help

[Request Long Help](#)

An average wait time for 6 minutes ago for 4 students

| # | Name    | Question Asked | Time          |
|---|---------|----------------|---------------|
| 1 | Justin  | 0              | 6 minutes ago |
| 2 | Billy   | 0              | 6 minutes ago |
| 3 | Michale | 0              | 6 minutes ago |
| 4 | Jackson | 0              | 6 minutes ago |

## (2)/statistics

Chromium Web Browser

日 10月 21, 16:21:05

Untitled - Chromium

localhost:8888/notebooks/Untitled.ipynb?kernel\_name=python3#

jupyter Untitled Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

```

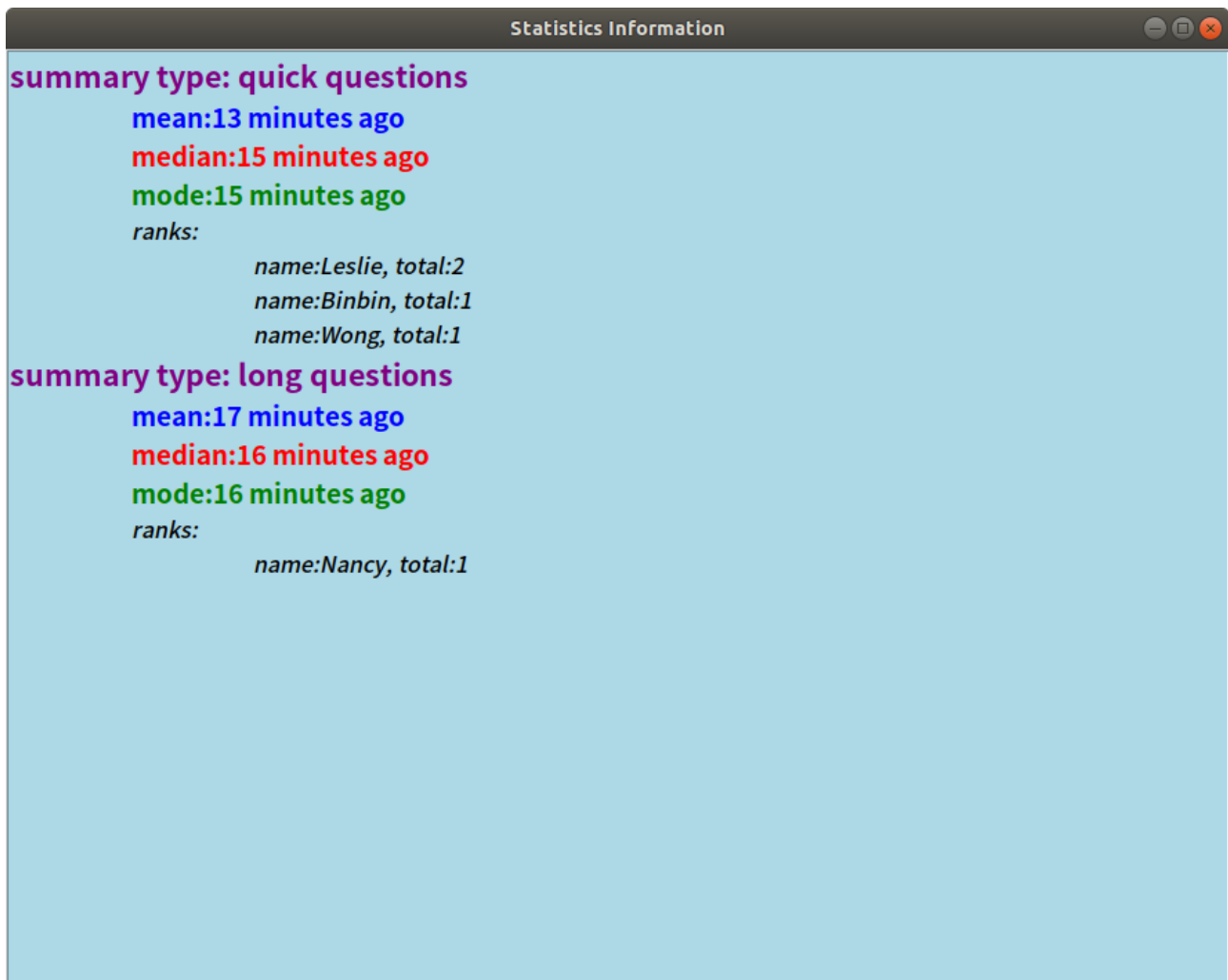
In [ ]: %pylab inline

In [ ]: import requests, pprint

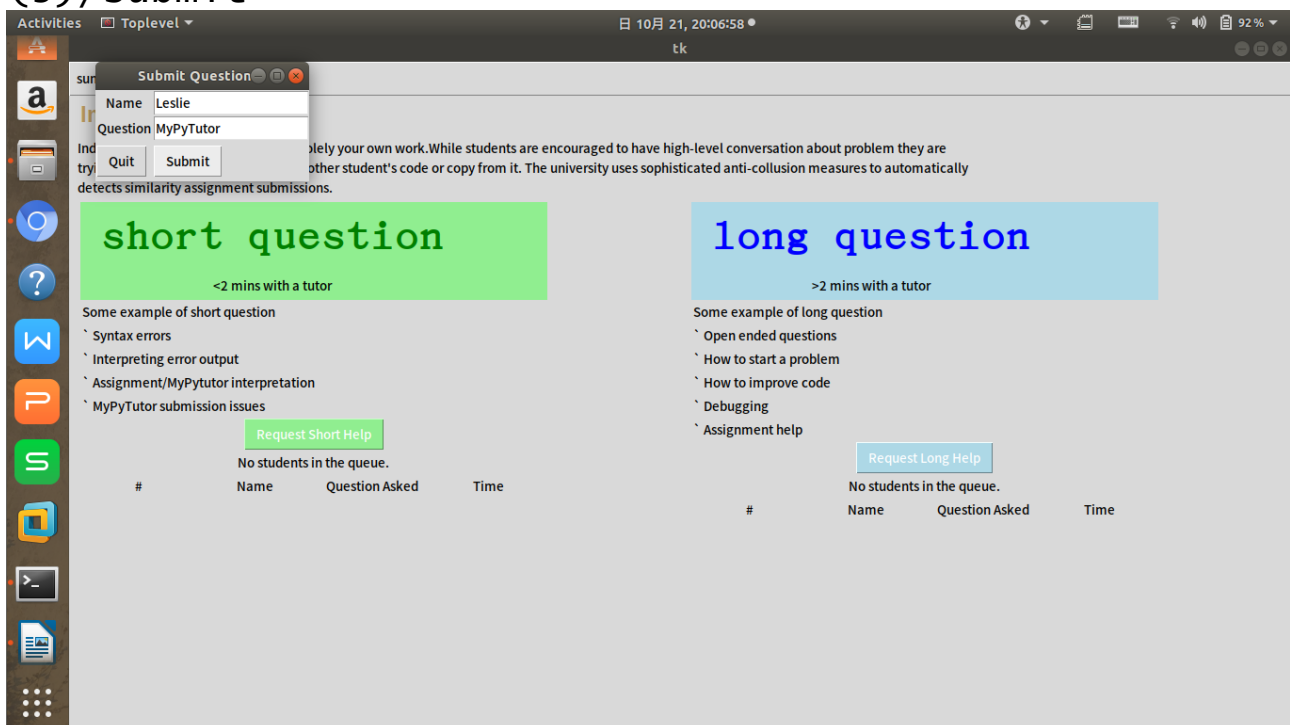
In [ ]: URL = 'http://localhost:8888/{}'

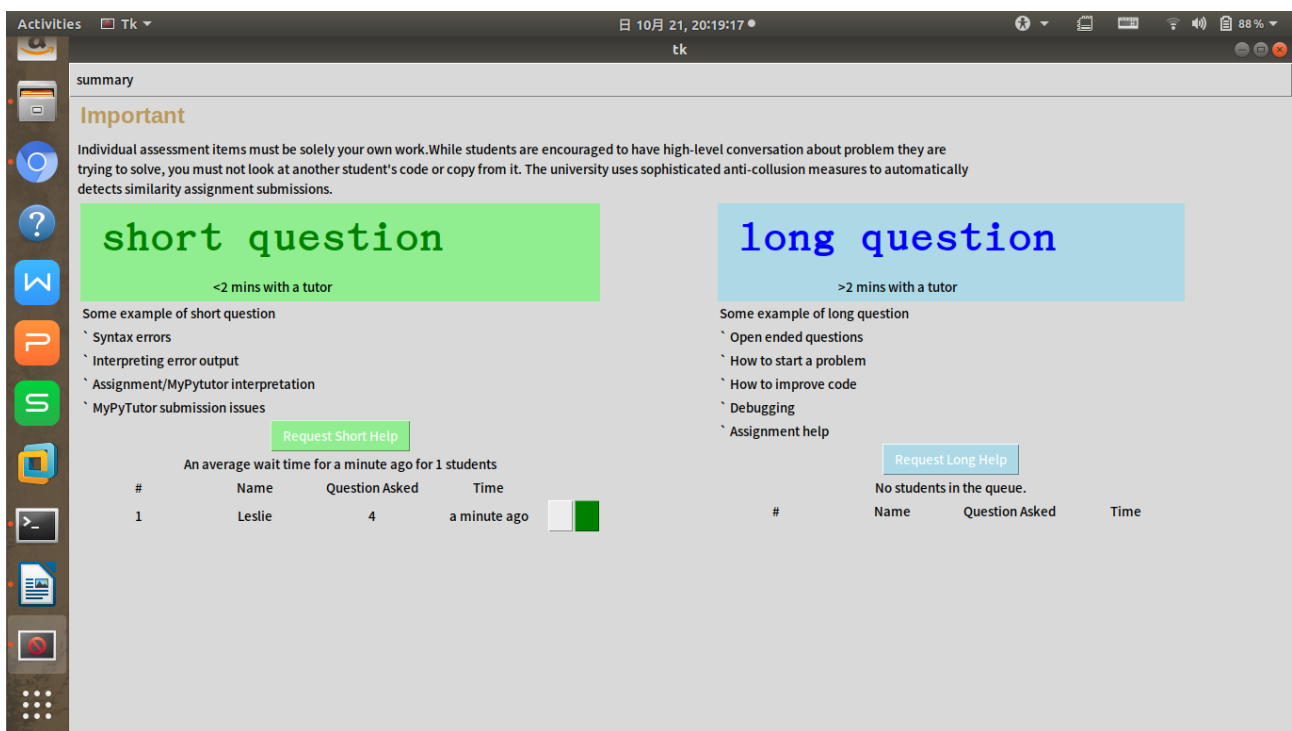
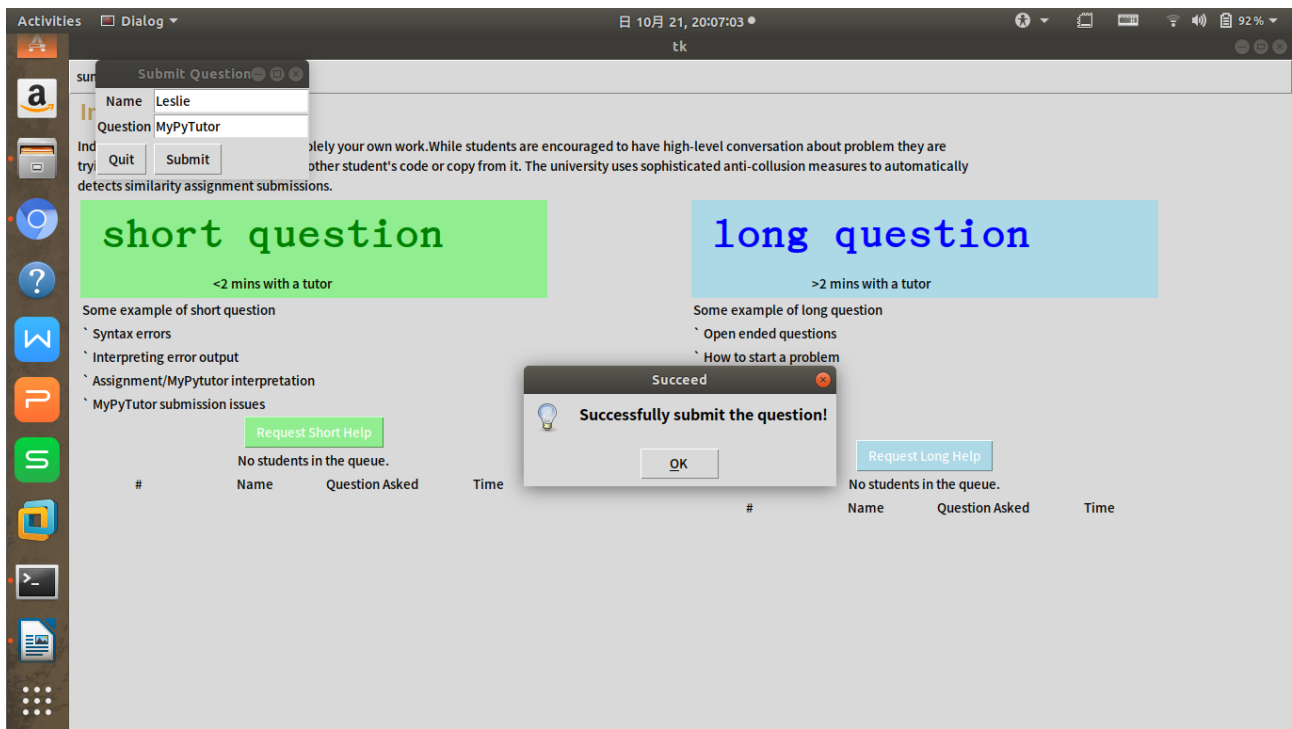
In [25]: resp = requests.get(url=URL.format('statistics'), params={'question_type': 'long'})
pprint.pprint(resp.text)
({'ranks': [{"total": 1, "name": "Nancy"}, {"total": 1, "name": "Binbin"}, {"total": 1, "name": "Wong"}], "mean": "16 minutes ago", "median": "15 minutes ago", "mode": "15 minutes ago", "status": "OK"})

In [26]: resp = requests.get(url=URL.format('statistics'), params={'question_type': 'quick'})
pprint.pprint(resp.text)
({'ranks': [{"total": 2, "name": "Leslie"}, {"total": 1, "name": "Binbin"}, {"total": 1, "name": "Wong"}], "mean": "13 minutes ago", "median": "15 minutes ago", "mode": "15 minutes ago", "status": "OK"})
  
```

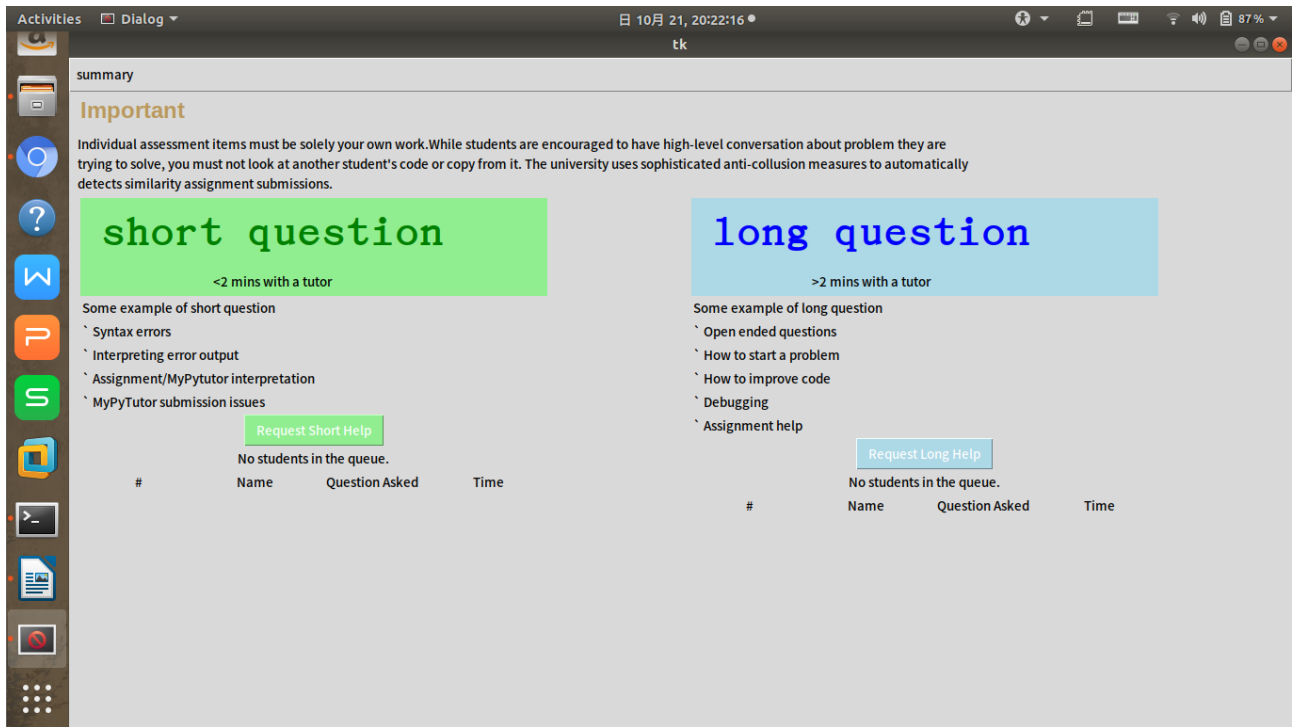


(3)/submit

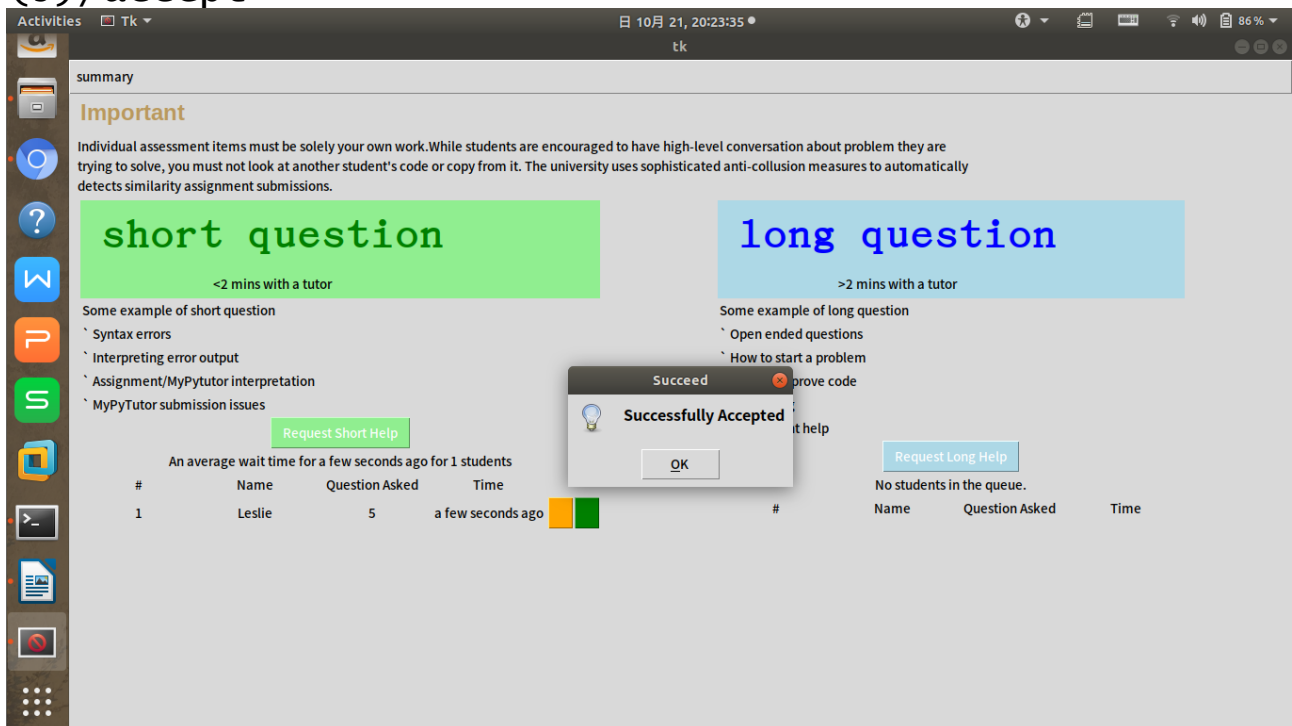




(4)/cancel



(5)/accept





ActivitiesDialog

日 10月 21, 20:23:40tk

summary

Important

Individual assessment items must be solely your own work. While students are encouraged to have high-level conversation about problem they are trying to solve, you must not look at another student's code or copy from it. The university uses sophisticated anti-collusion measures to automatically detects similarity assignment submissions.

short question

<2 mins with a tutor

Some example of short question

- Syntax errors
- Interpreting error output
- Assignment/MyPyTutor interpretation
- MyPyTutor submission issues

Request Short Help

No students in the queue.

| # | Name | Question Asked | Time |
|---|------|----------------|------|
|---|------|----------------|------|

long question

>2 mins with a tutor

Some example of long question

- Open ended questions
- How to start a problem
- How to improve code
- Debugging
- Assignment help

Request Long Help

No students in the queue.

| # | Name | Question Asked | Time |
|---|------|----------------|------|
|---|------|----------------|------|