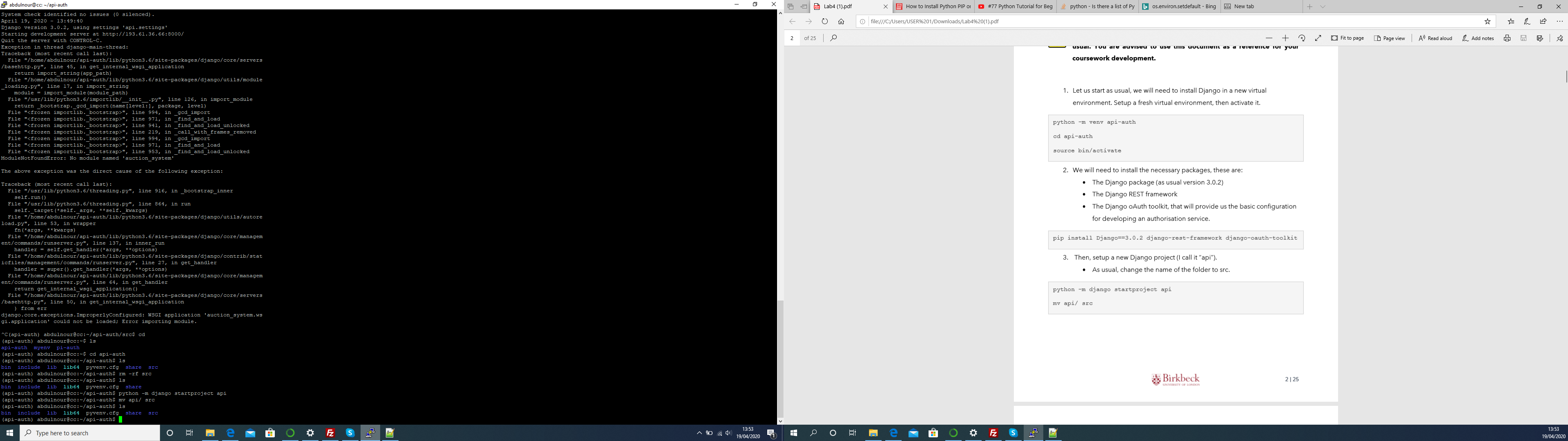
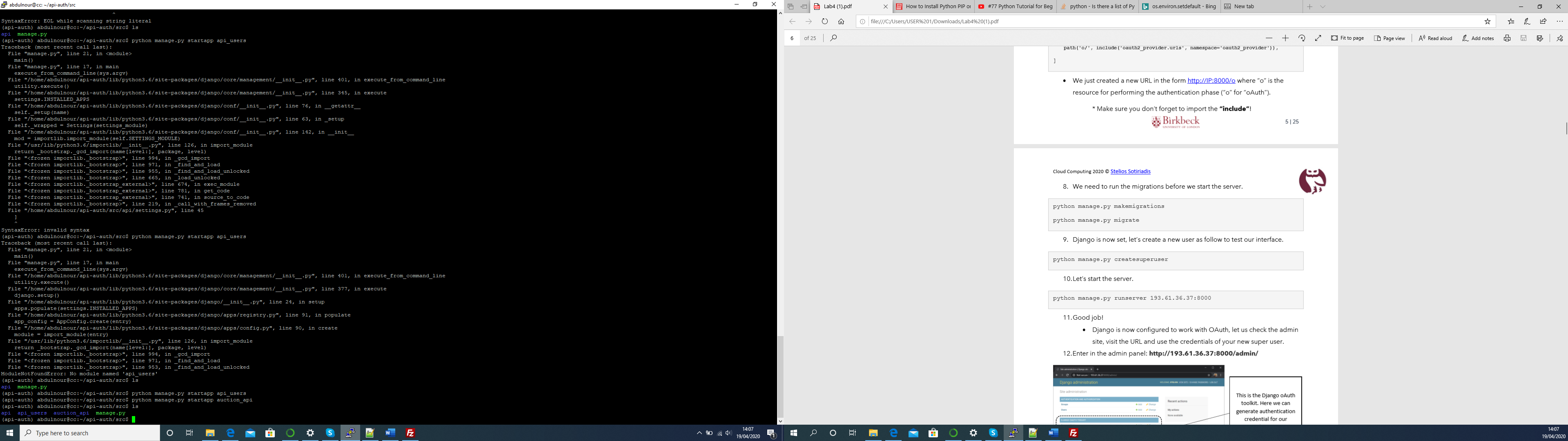
Cloud Computing Concepts coursework

This report will discuss my findings in the cloud computing coursework.

To create my setup I was required to create a virtual environment which I called “api-auth”. Inside this virtual environment I created a new project which I called api initially which I then renamed to src. Inside this src project I created two apps, one was called api\_users and the other auction\_api. Inside the virtual I environment I installed Django, the Django REST and Django oauth toolkit packages. Once I made these changes I then had to migrate and then create a superuser(admin) before running the server and deploying my applications. I also had to add my server to allowed hosts in settings.py

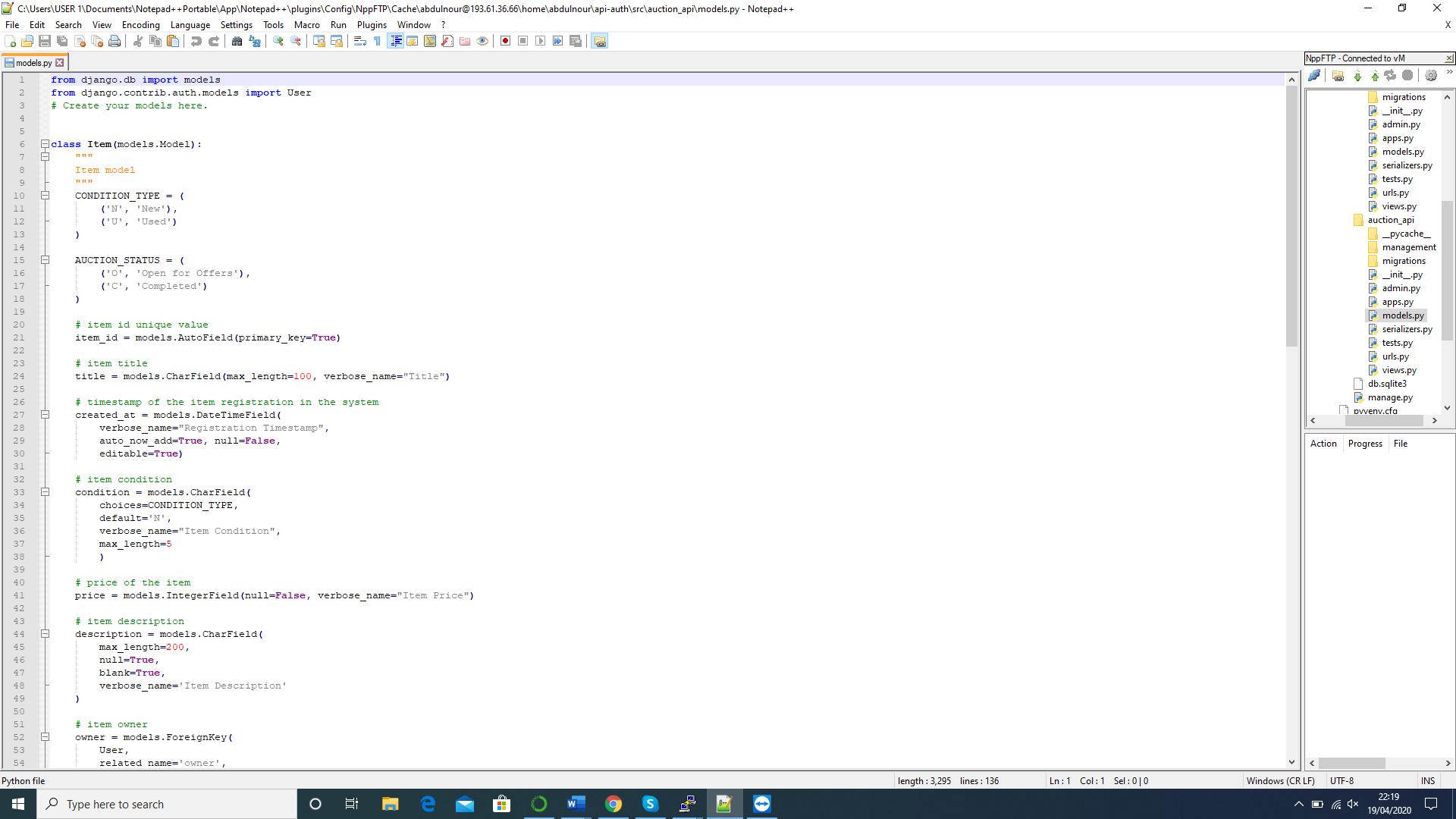




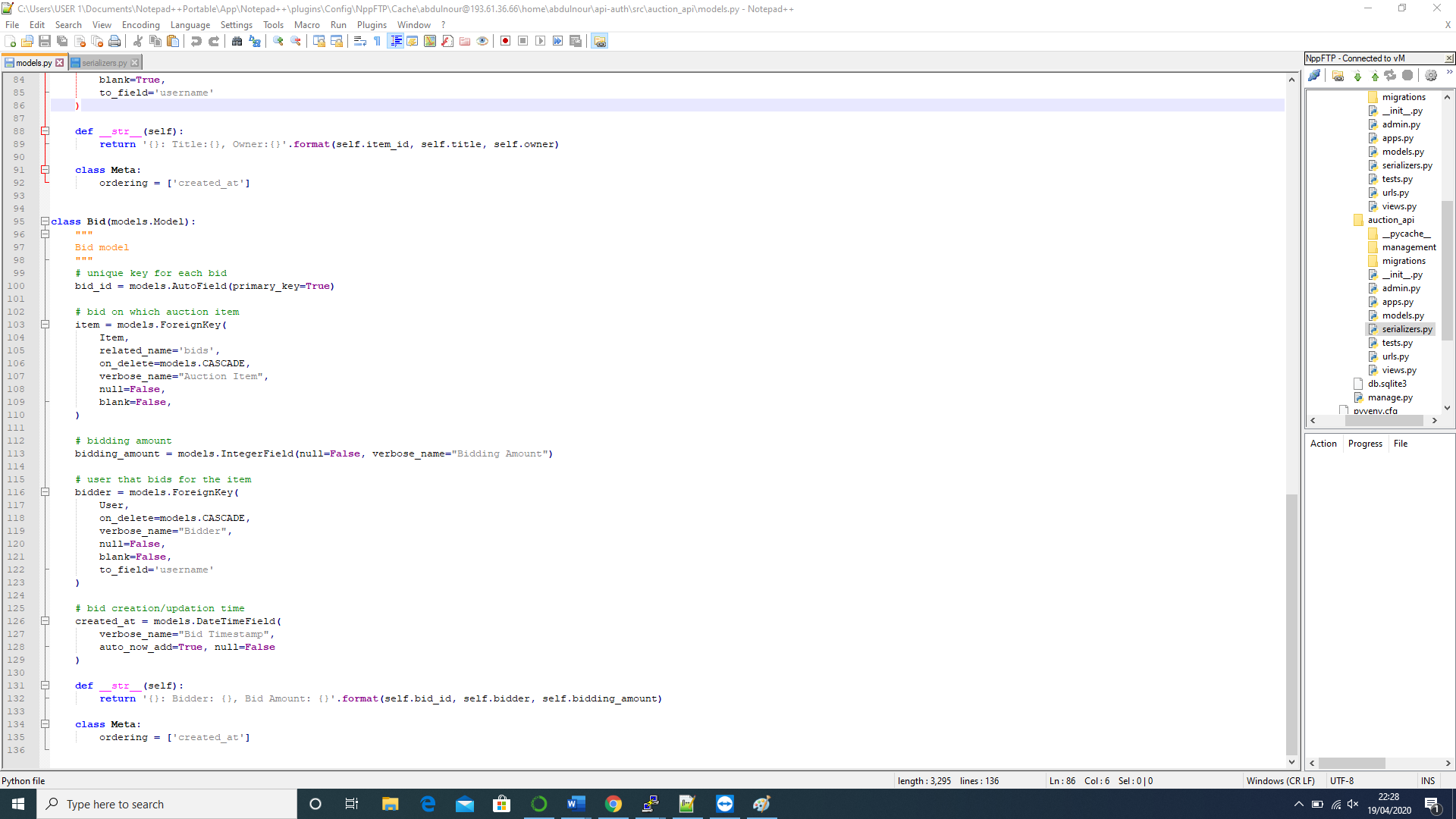
Database.

For my database design I have two tables one called bid and the other called item. Both tables have the contain the auction details.

The table Item as seen below has the fields = ['item\_id', 'title', 'owner', 'created\_at', 'condition','price', 'description', 'winner', 'auction\_status', 'expire\_time', 'bids']. This table contains details of the auction aswell as the information for the item.

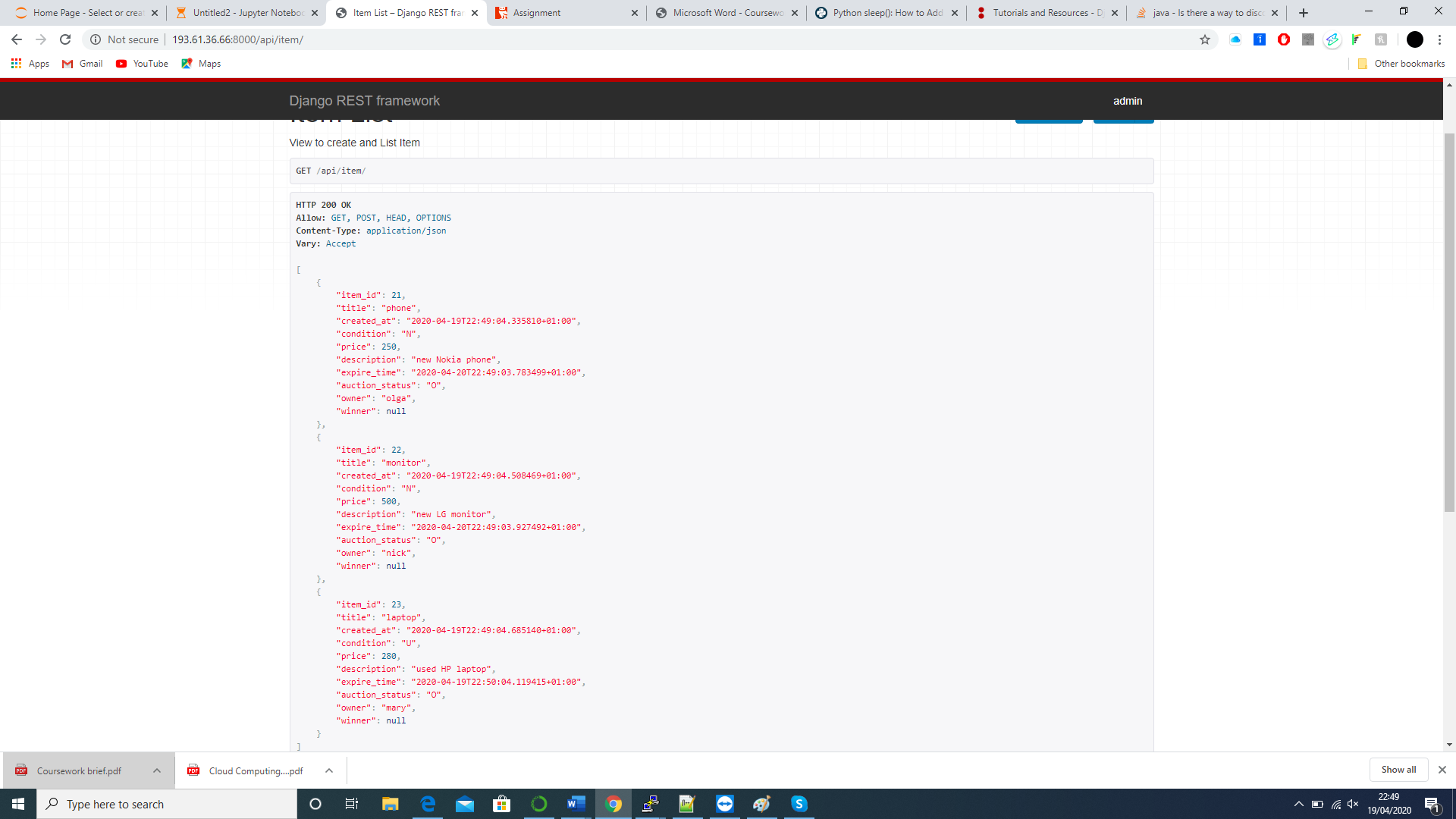


Below I have the Bid table where it contains information about which item was bidded on, the amount of the bid and the bidder.

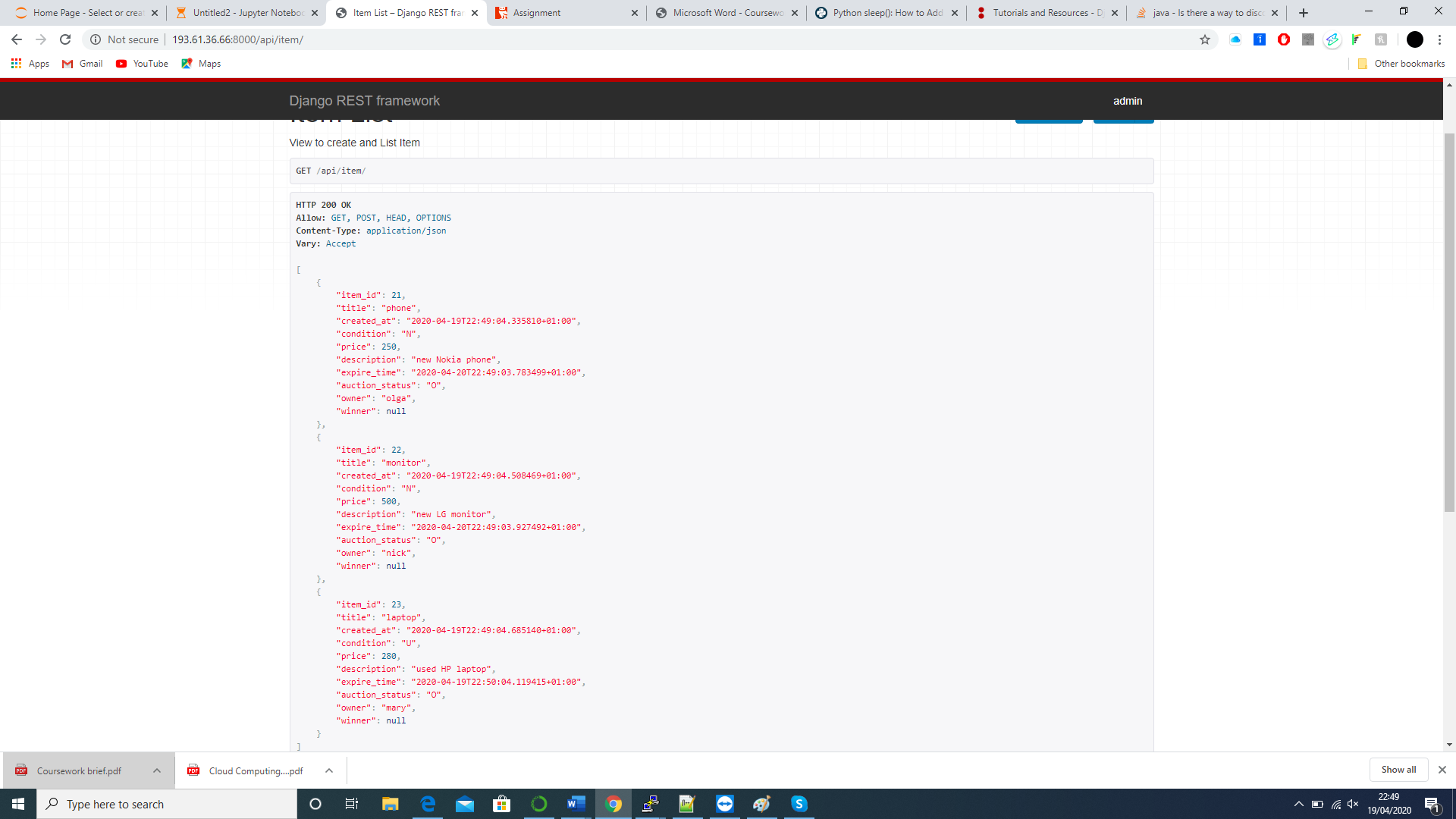


Api endpoints:

/api/item/ -- the allowed methods are: GET, POST. The GET method is to list the item. The POST method is to create the new item. To view an items details use the /api/item/<item\_id>.



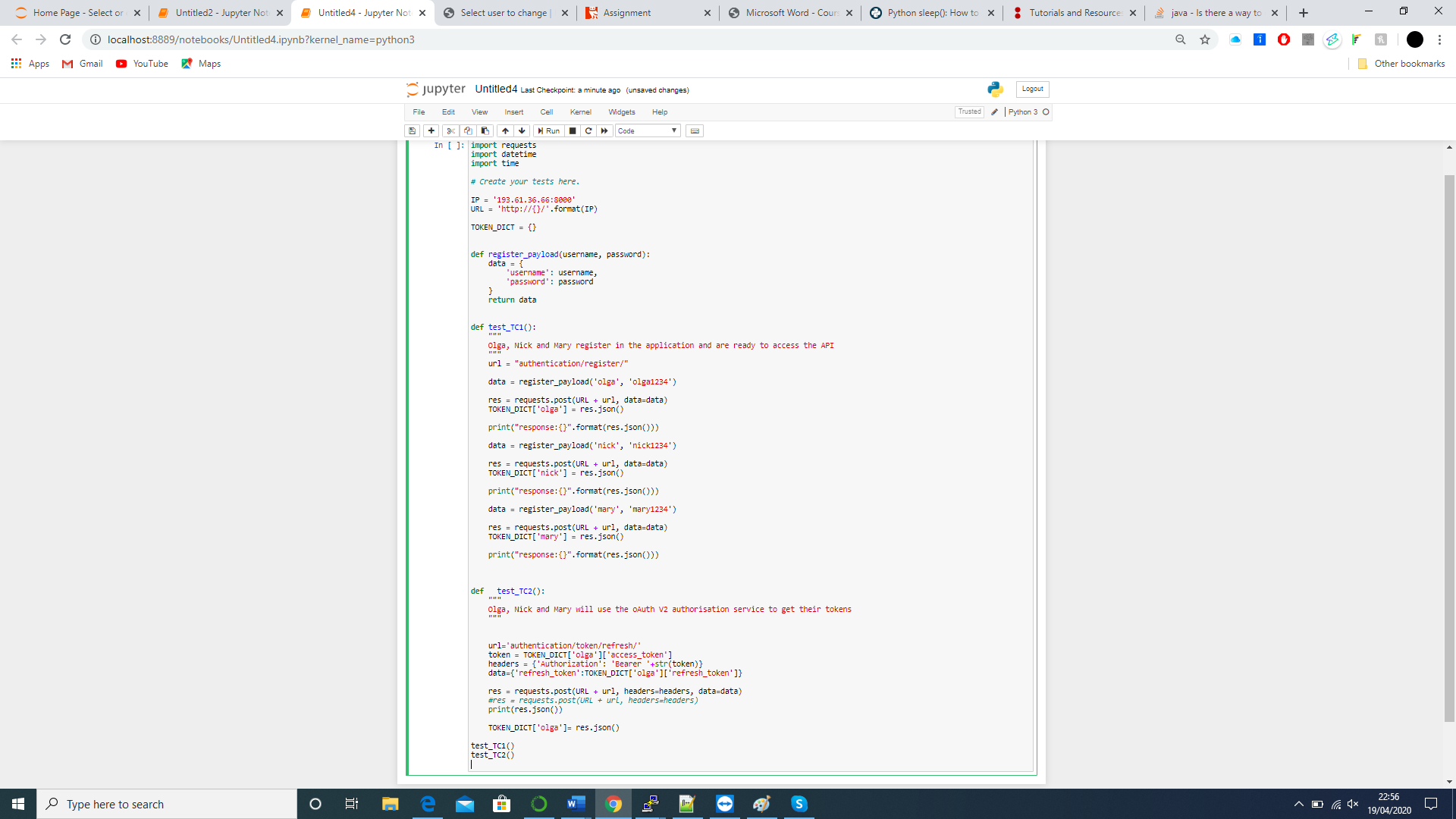
/api/bid/ -- the allowed methods are POST. The user can only create a bid on an item.

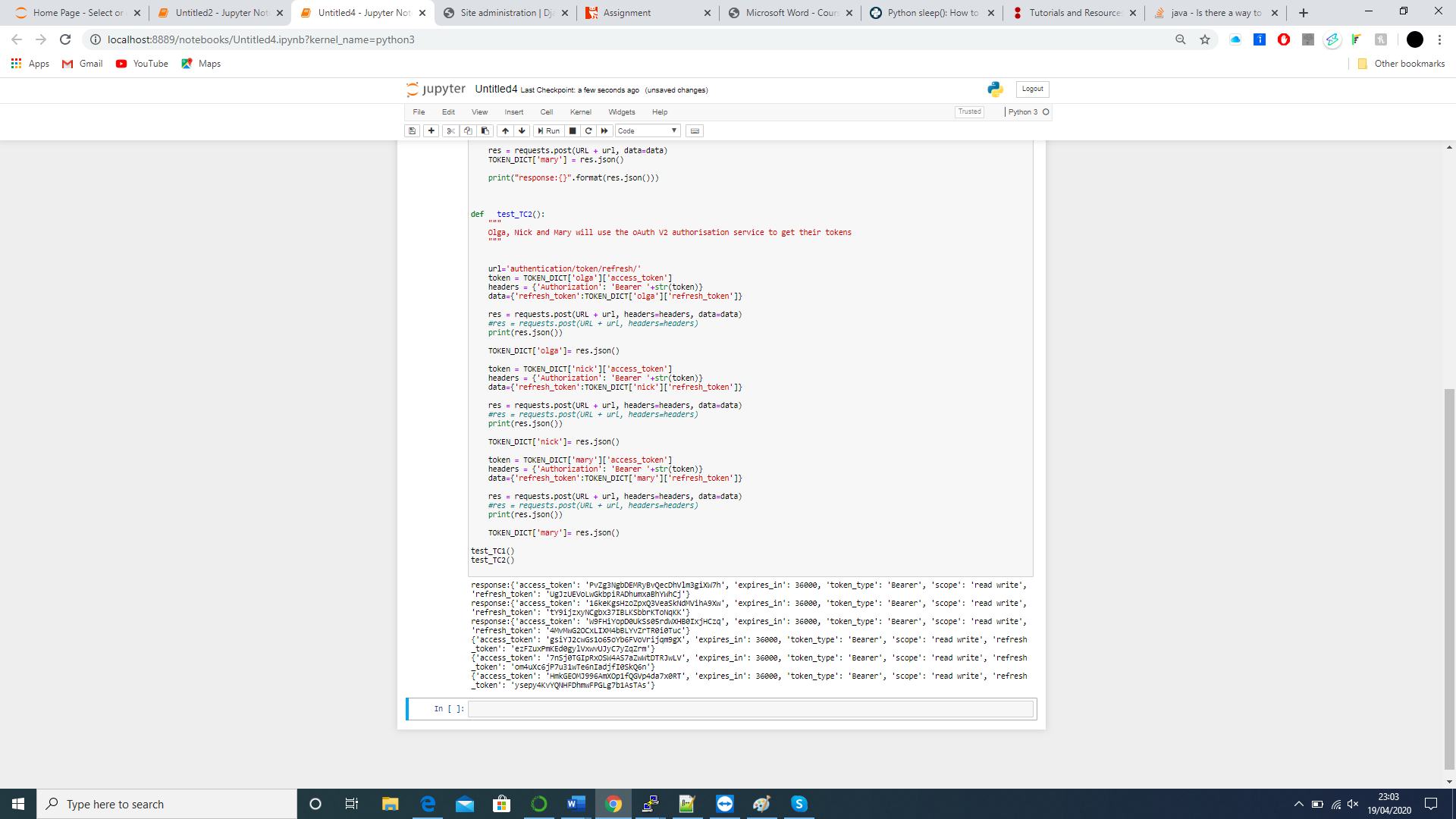


Development of a testing application.

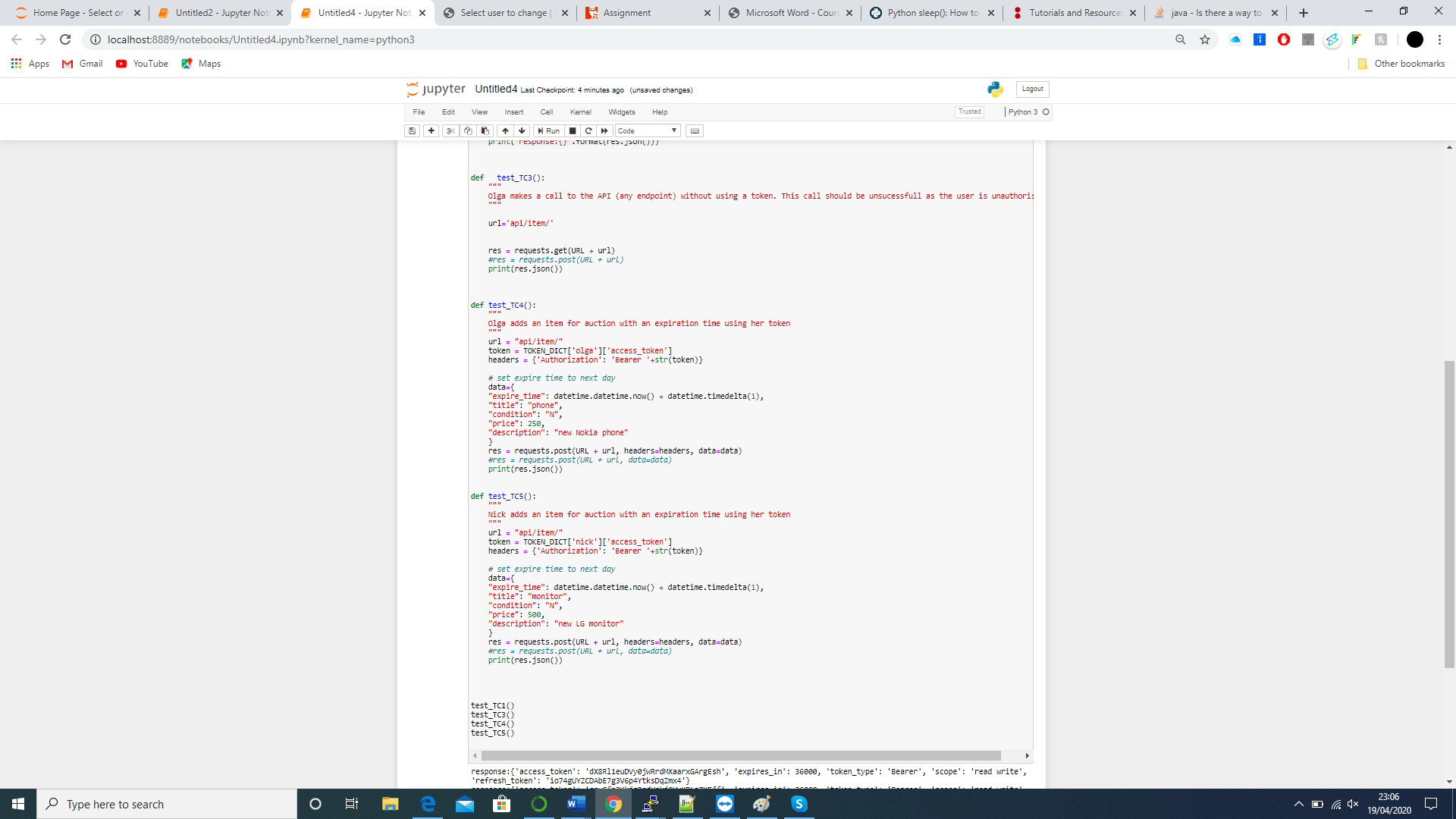
So for the testing application, I used jupyter labs for this which I ran from my desktop outside the virtual machine.

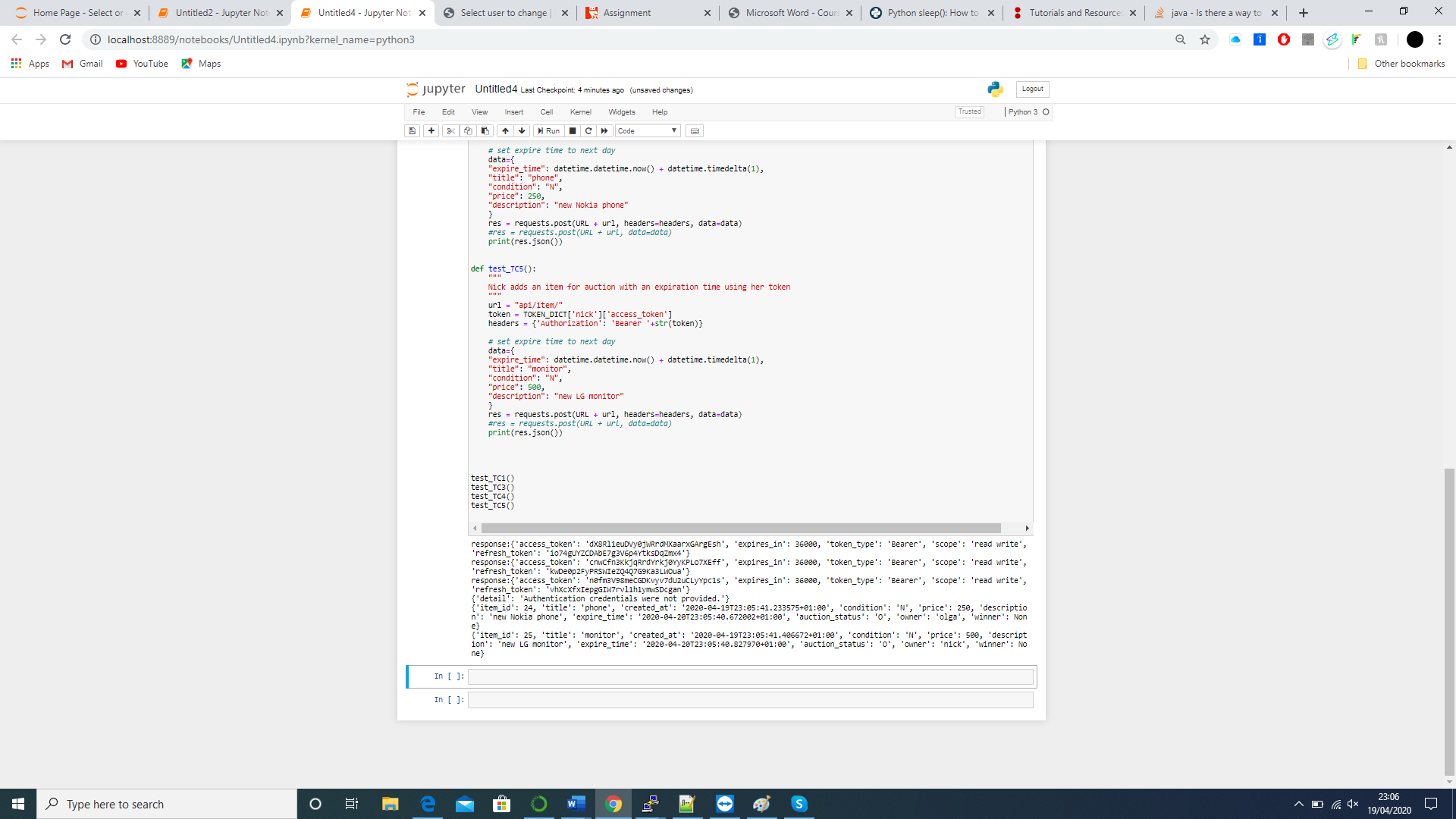
TC1 & TC2



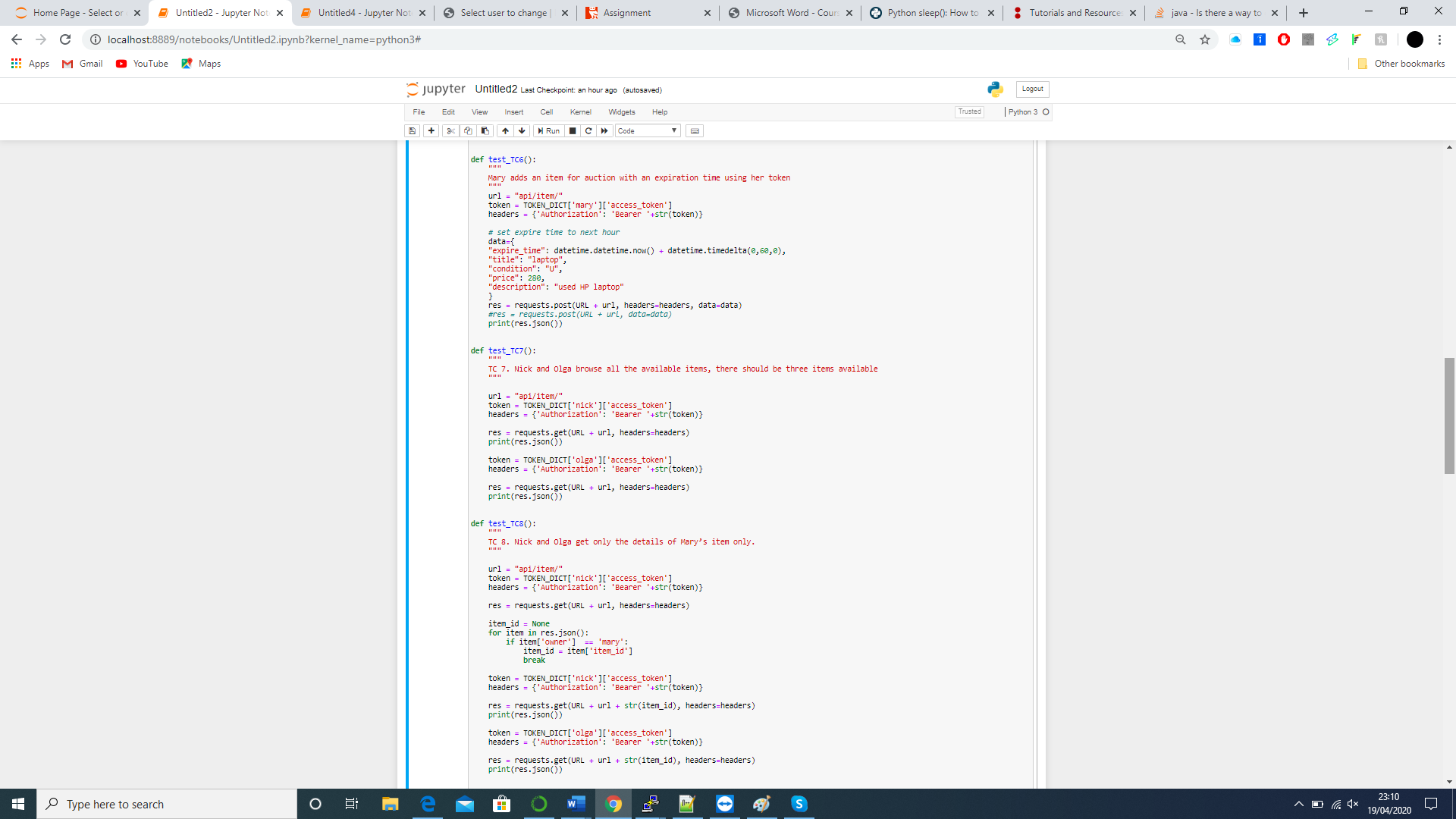


TC3, 4, 5

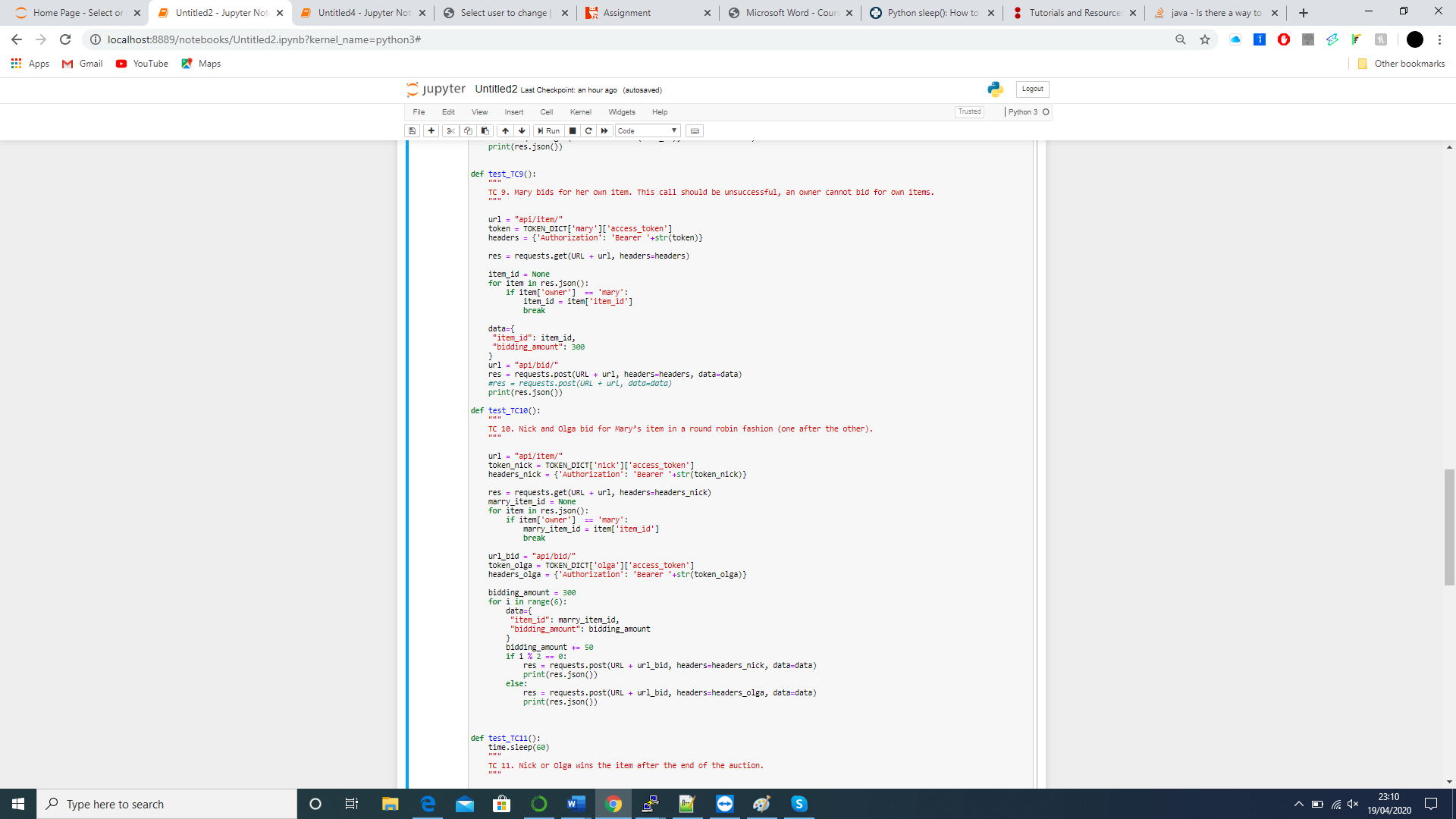




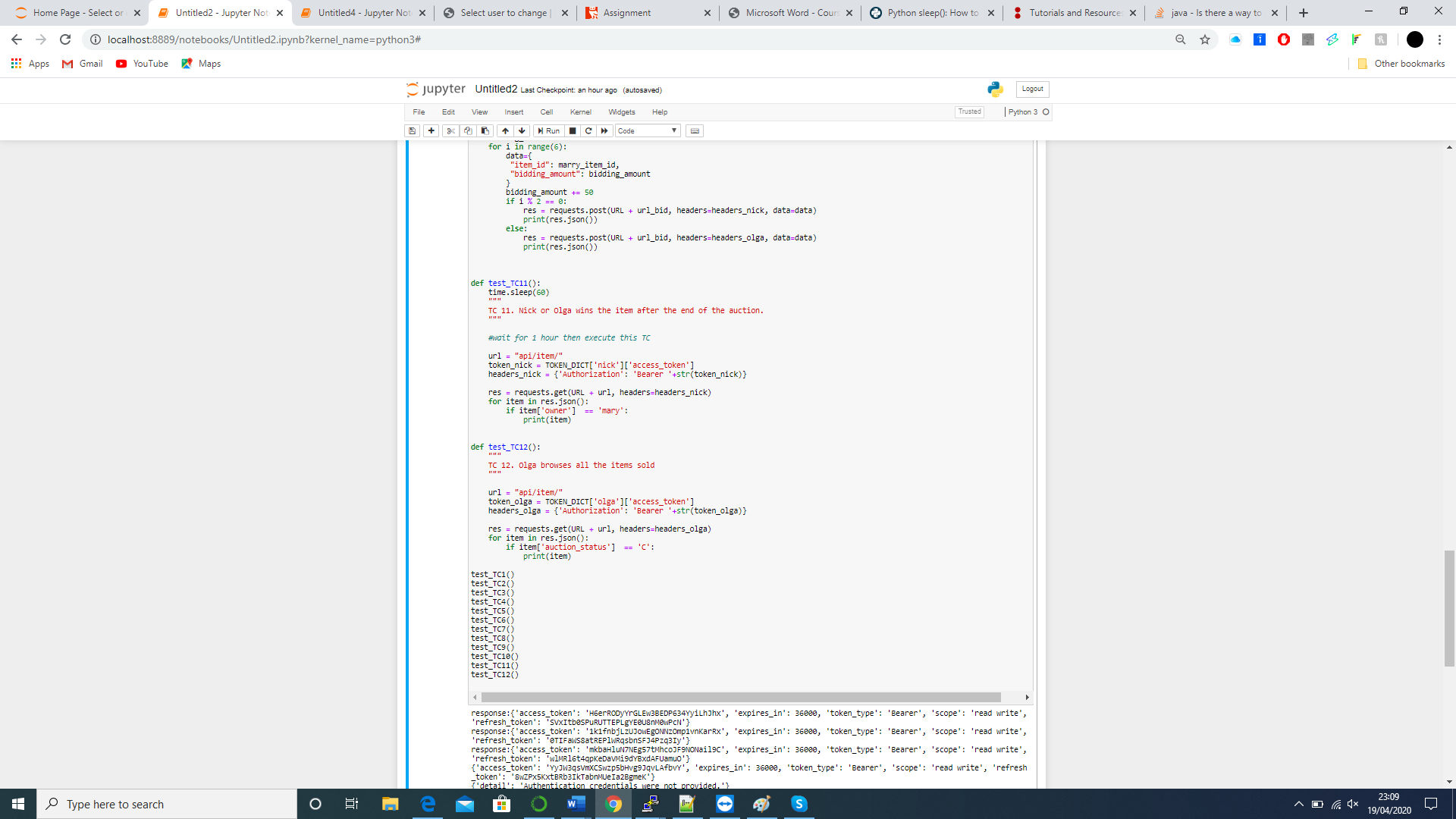
TC 6,7,8



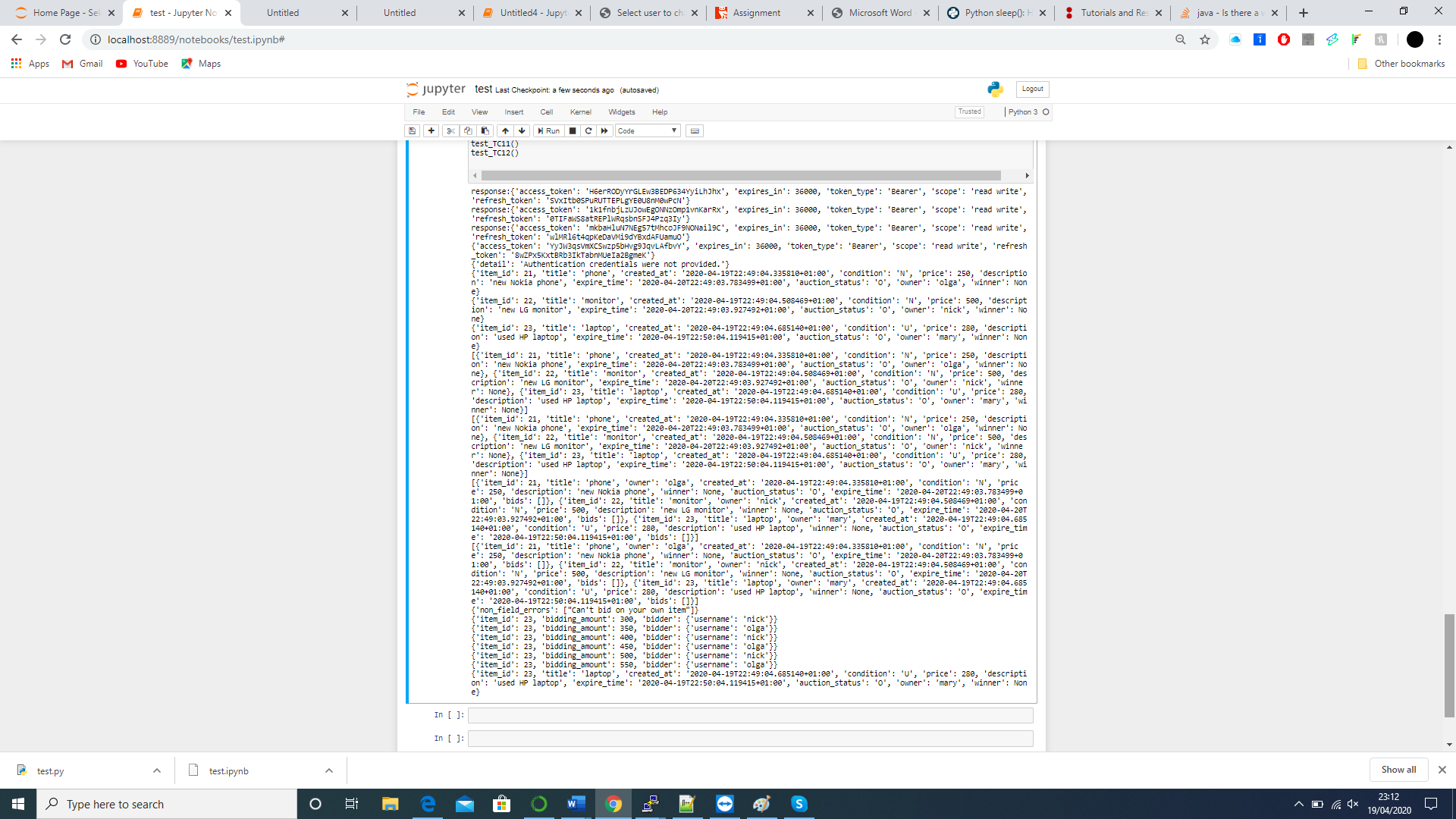
TC 9, 10

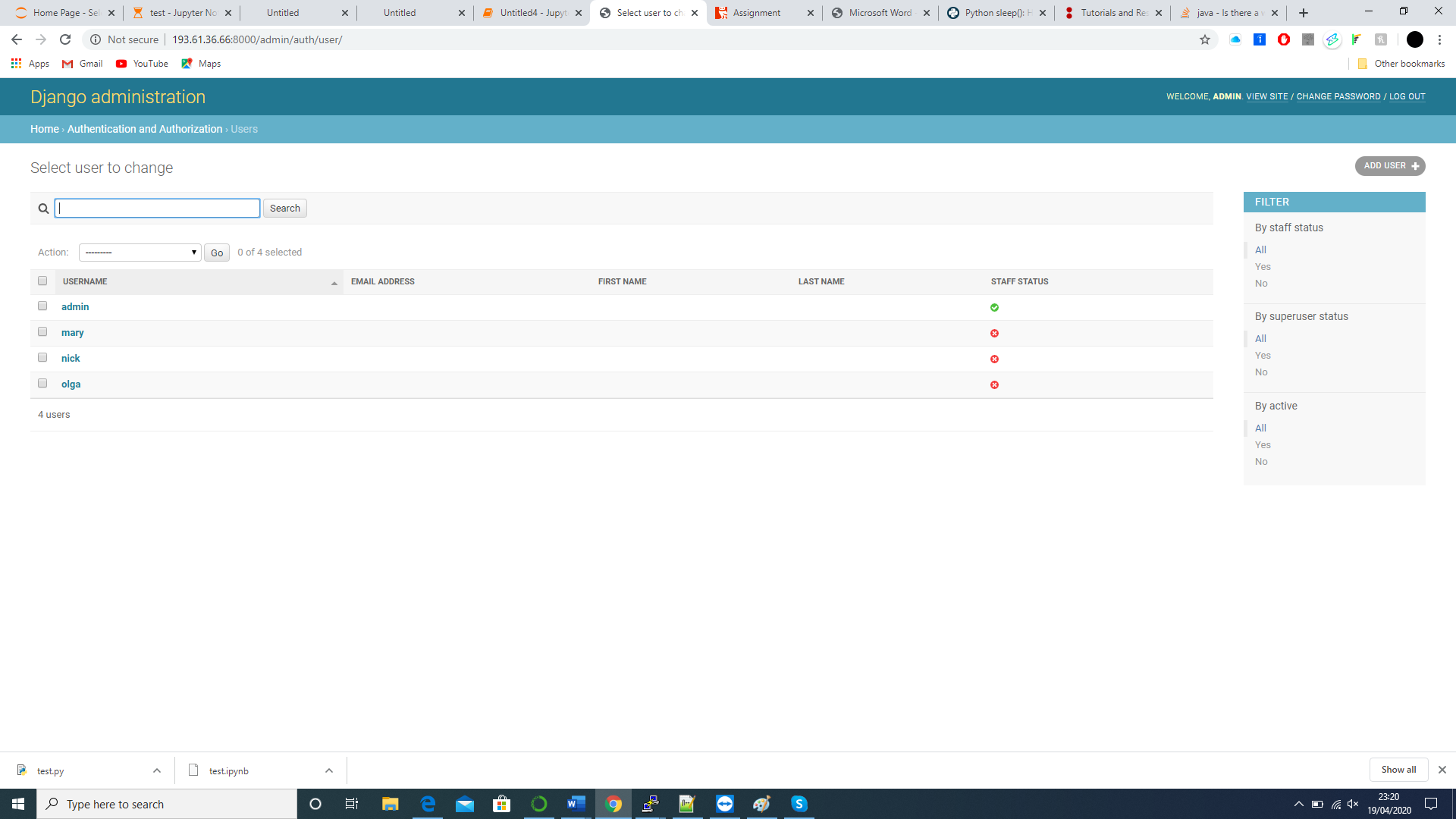


TC 11, 12

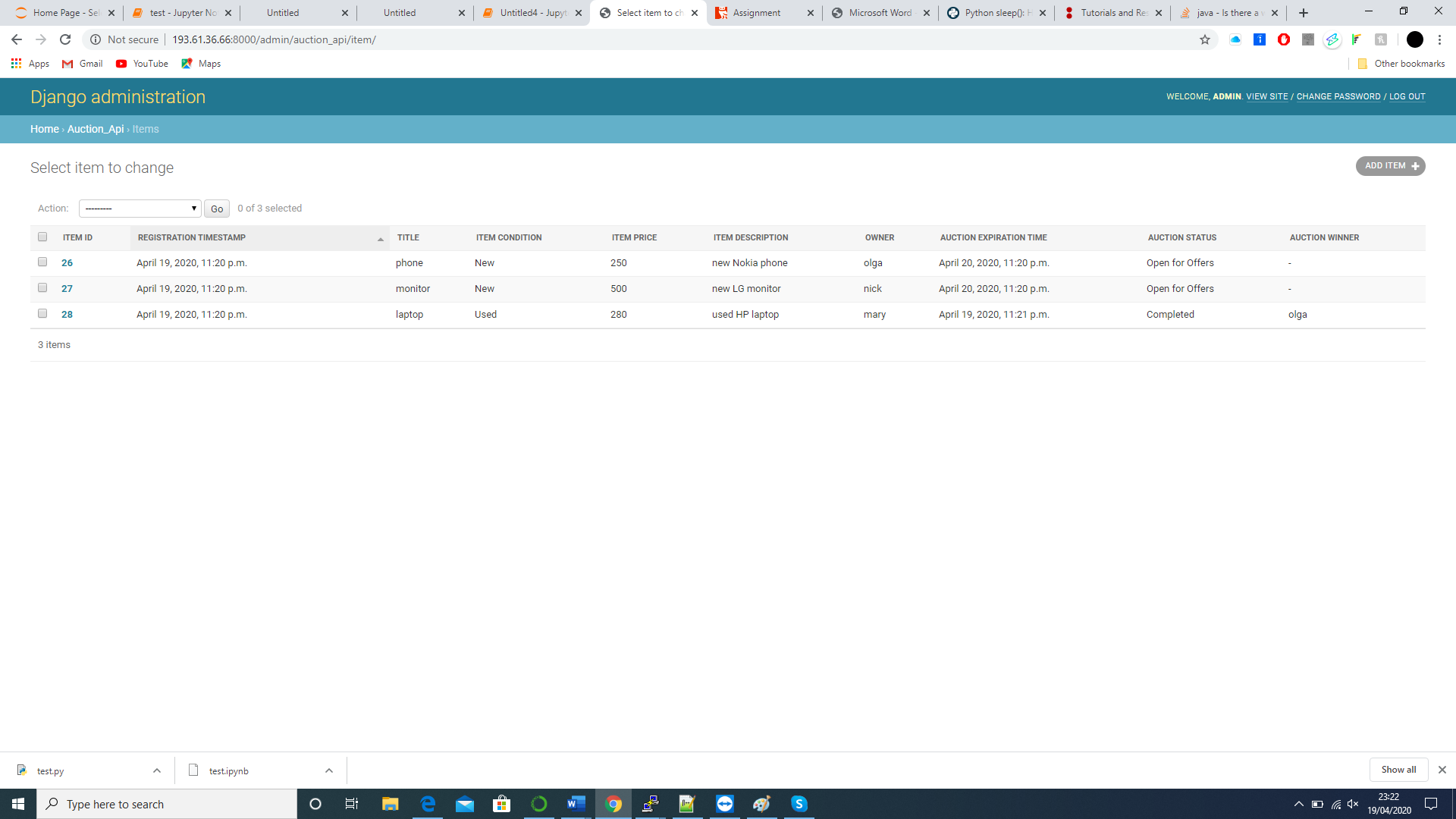


This is the successful output of the test cases.



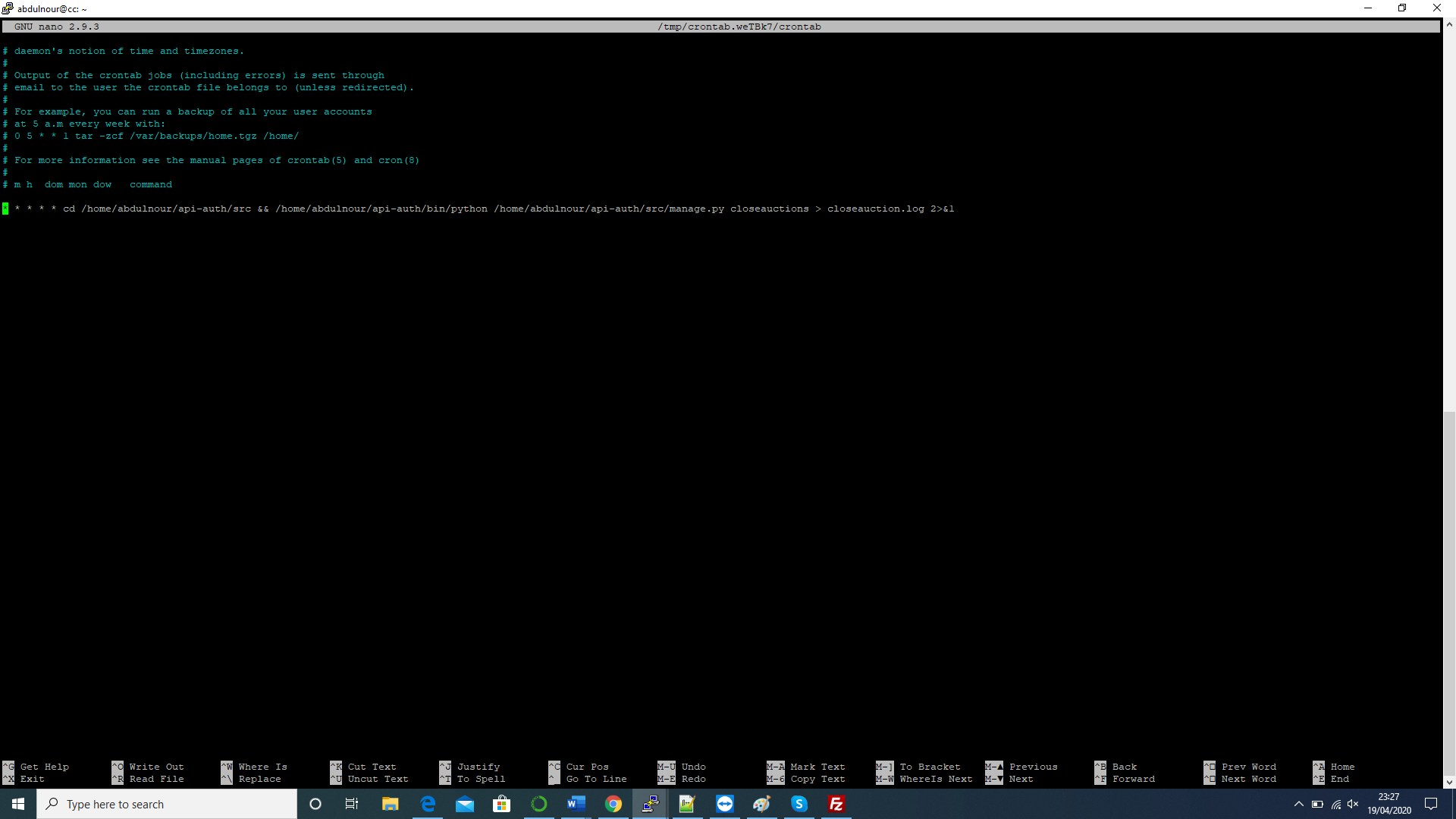
As you can see the 3 users have been registered

This is the items api endpoint. As you can see we have 3 items and 1 auction has been completed which Olga has won.



In my solution to I also use Crontab which is a file which contains the schedule of cron entries to be run at certain specified times.

In order to close the auction after the expired time, a Django management command was added called closeauction. This command will check the database for all the items where the expiry time is up and change the state of those items as well as find a winner of the auction from the bids. If no bid was present the winner will be set to null. The closeauction command will be run every minute, on Linux using crontab, which can be changed depending on how long the auction is.



References:

<https://www.django-rest-framework.org/community/tutorials-and-resources/>

<https://docs.djangoproject.com/en/dev/howto/custom-management-commands/#howto-custom-management-commands>

<https://www.adminschoice.com/crontab-quick-reference>

Cloud Computing Concepts Lab 4.