

Backend Engineer – Assignment

Sent by Juan Camilo Muñoz Arenas

1. How the code works: the code starts importing the library used to process and split the .csv file for easier access when dealing with large files. I chose pandas library because is a powerful data analysis library in Python that offers a wide range of functions and features for manipulating, analyzing, and visualizing large datasets. When dealing with large databases, Pandas provides several benefits such as the ability to work with big data in manageable chunks, handle missing and inconsistent data, perform complex data manipulations efficiently, and produce informative visualizations to help gain insights into the data. Then, I chose a chunk number (number of values taken by each partition of the .csv file), in this case just for the example file that I took (100k values with 3 columns) I chose a chunksize of 10k values.

The next step is simply reading the csv file in the established chunk sizes for easier access in case the file is bigger than the memory, for this case the example file is called 'File1.csv'.

A DataFrame called 'total_plays_df' is initiated with the column names being the ones stated by the assignment ('Song', 'Date', 'Total number of Plays for Date'), this data frame is the one that is going to be filled with the processed values.

I start the chunks for loop; we need a for loop because we deal with smaller parts for the complete file, thus is necessary to keep adding up the processed values to the data frame.

Inside the loop, I process the dataframe as established by the assignment, I start by changing the date that was an object type, to datetime (although is not necessary for this assignment, it is a good practice to do it as to avoid any problems after when dealing with time in a dataset). The main task is the line 13, when I use the pandas function groupby that associates the values by user given conditions, in this case I associate it by Song and Date, and then summing the values (in this case it is the column named 'Number of Plays'). Afterwards I reset the index to have the Song and Date again as values and renaming the Number of Plays column to get the assigned name. I do it again for the Dataframe 'total_plays_chunk' in this case being done from the dataframe previously processed and appending those values to the final DataFrame called 'total_plays_df'.

At the end we just do the group by function again and reset index in the final Dataframe called 'total_plays_df' and we call the function to_csv to create a new file (in this case named 'answer.csv').

The complexity of this code depends on the size of the input file and the number of unique songs and dates in the file.

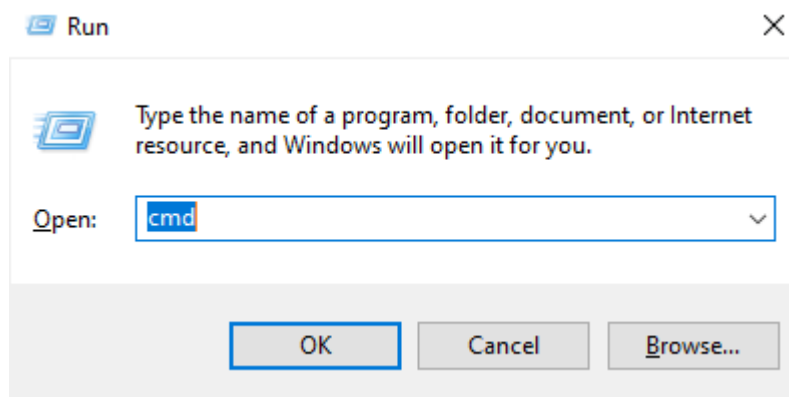
The outer loop of the code iterates over the CSV file in chunks of size chunk_size, which would take $O(n)$ time where n is the number of rows in the file.

Within the loop, the `groupby()` method is called twice, which could potentially take $O(n \log n)$ time where n is the number of unique song and date combinations in the current chunk. However, the time complexity of the `groupby()` method in Pandas can be affected by many factors such as the size of the DataFrame, the number of groups, and the algorithm used to perform the grouping.

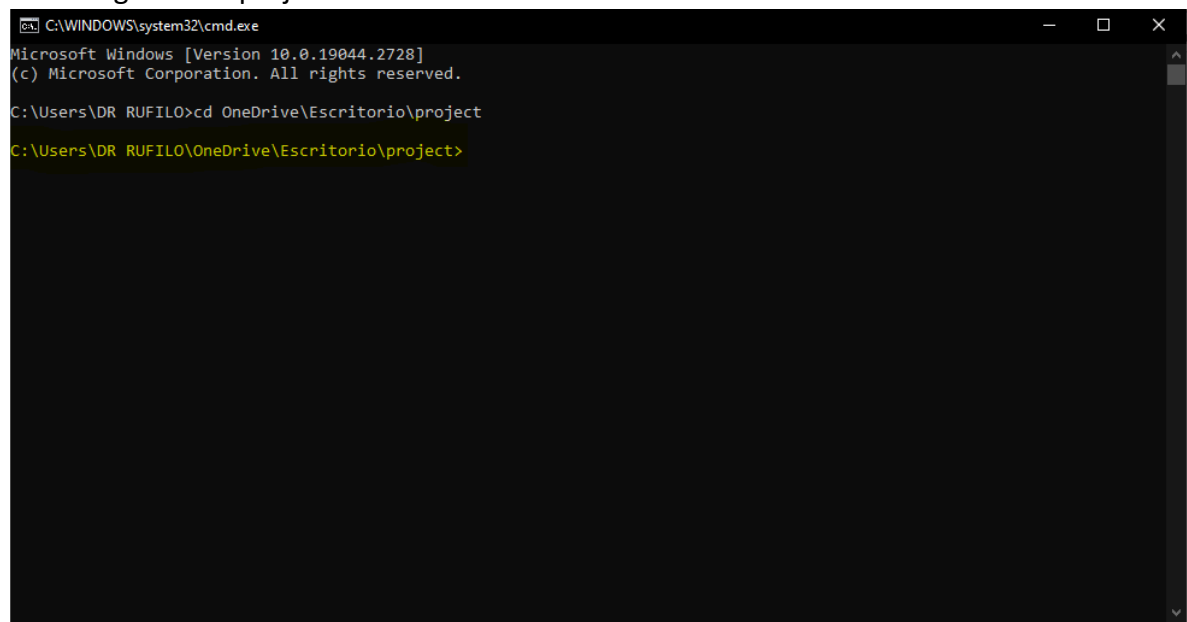
Overall, the time complexity of the code would be somewhere between $O(n)$ and $O(n \log n)$ depending on the size of the input file and the number of unique song and date combinations in the file.

(https://pandas.pydata.org/docs/user_guide/gotchas.html#performance-considerations)

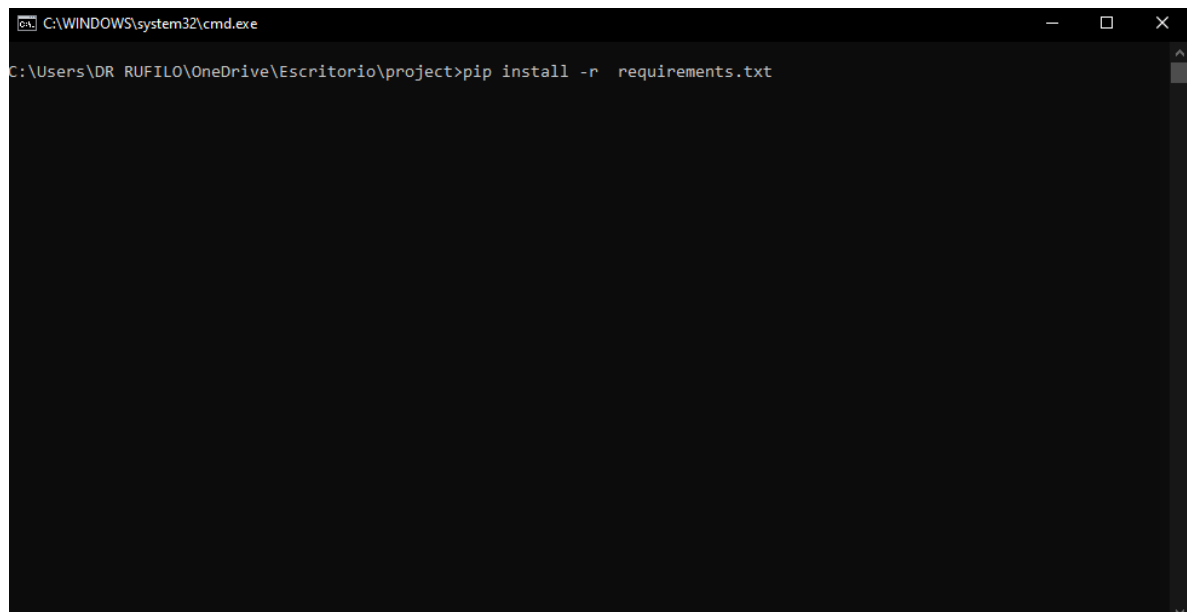
2. Now I will write the configurations and instructions necessary to start the project:
 - We are going to start by starting a terminal window (windows+R-cmd for windows OS).



- We will go to the project folder location.

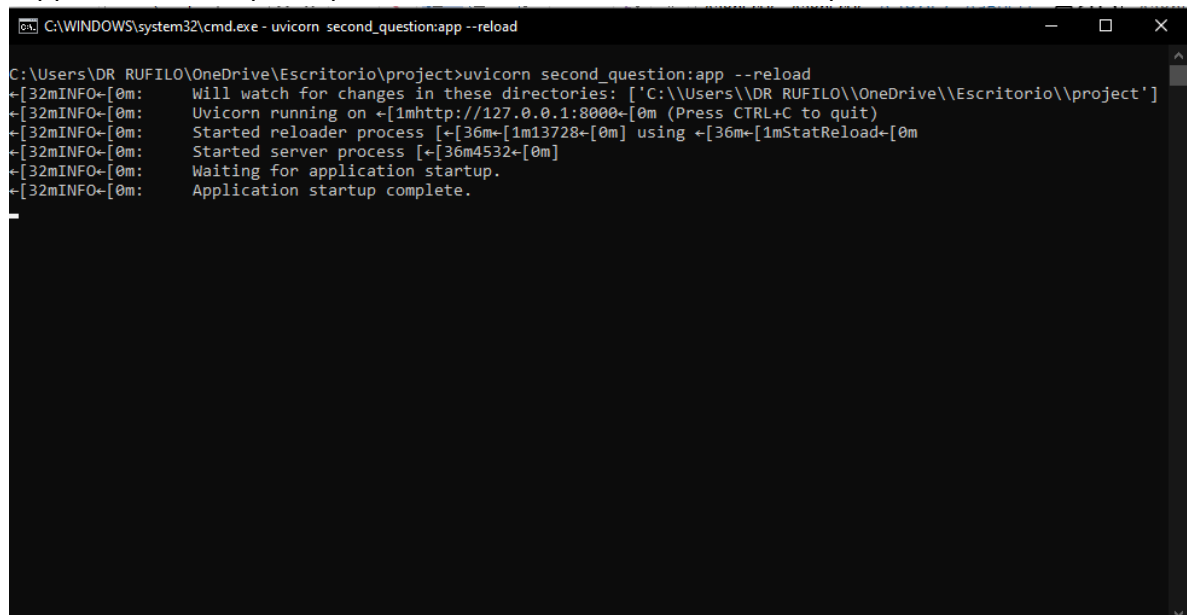


- We will run the command 'pip install -r requirements.txt' that will install two basic libraries for the right usage of the project, in this case it will be FastAPI and Pandas first.



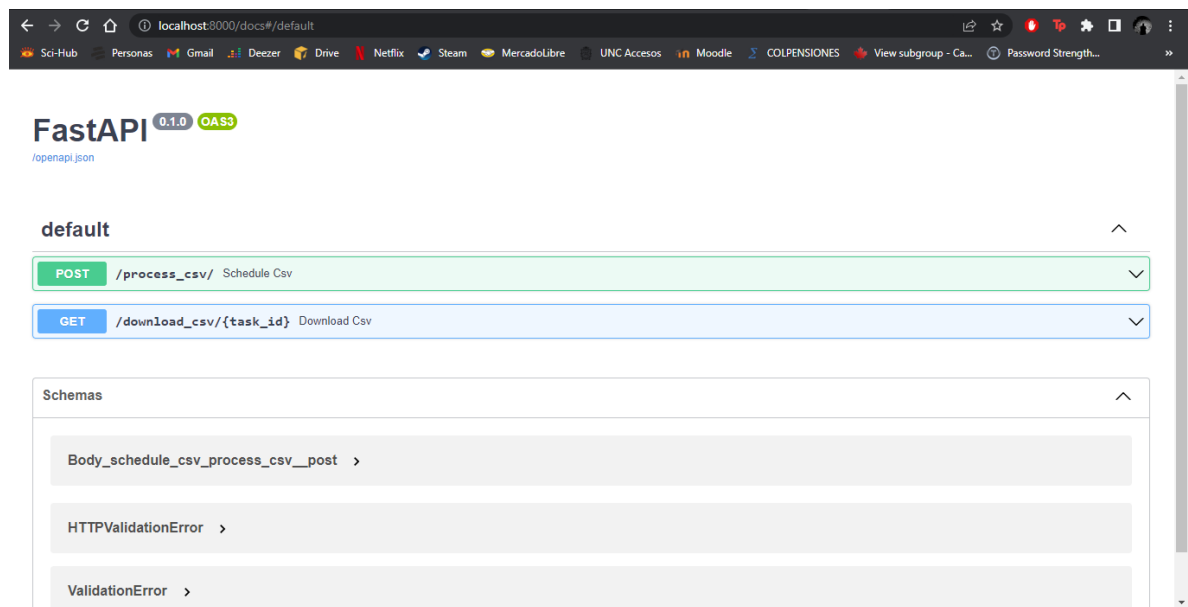
```
C:\WINDOWS\system32\cmd.exe
C:\Users\DR_RUFILO\OneDrive\Escritorio\project>pip install -r requirements.txt
```

- Install any other necessary library stated in the beginning of the file 'second_question.py'.
- In the cmd window we will run the command 'uvicorn second_question:app --reload' so we can start the server we will use and we will see the prompt 'Application startup complete' and we will know the server is up.

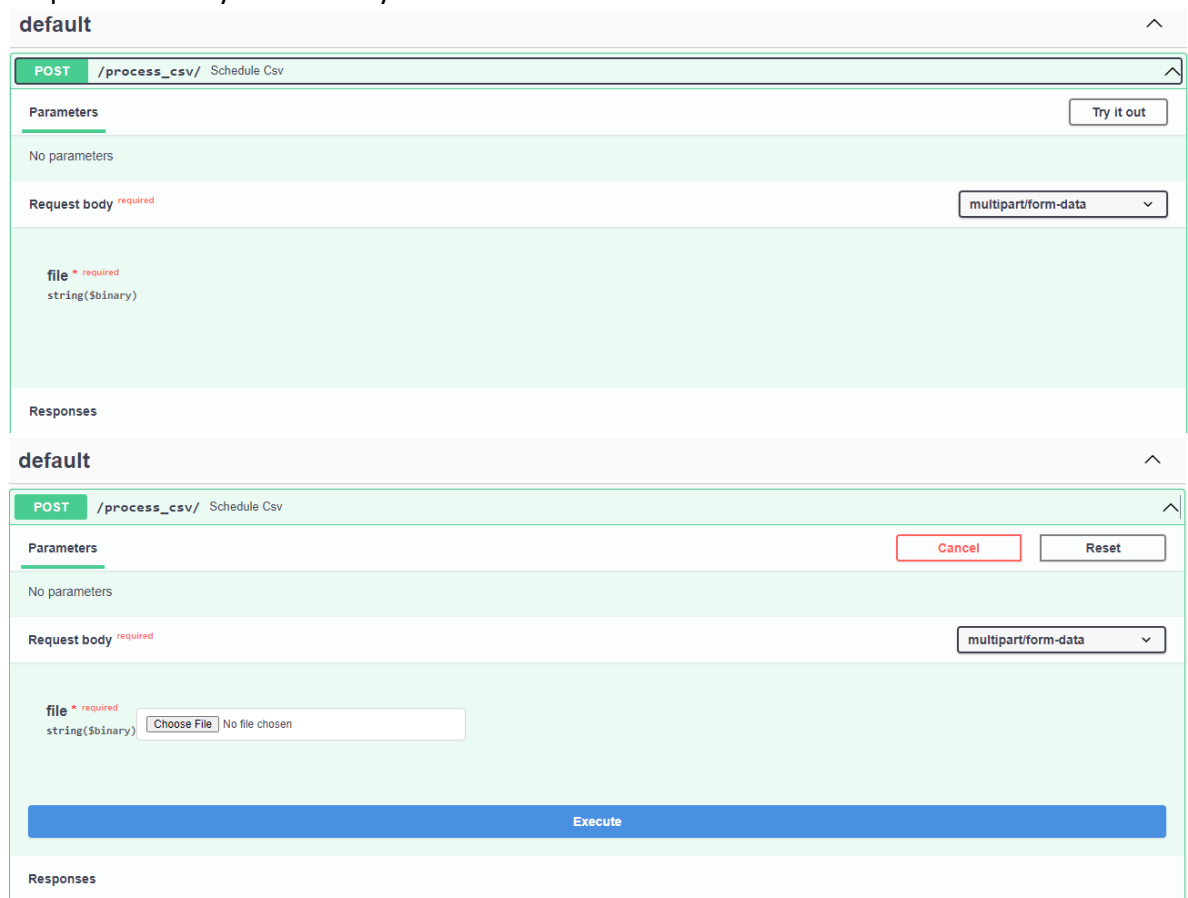


```
C:\WINDOWS\system32\cmd.exe - uvicorn second_question:app --reload
C:\Users\DR_RUFILO\OneDrive\Escritorio\project>uvicorn second_question:app --reload
+ [32mINFO+ [0m: Will watch for changes in these directories: ['C:\\Users\\DR_RUFILO\\OneDrive\\Escritorio\\project']
+ [32mINFO+ [0m: Uvicorn running on +[1mhttp://127.0.0.1:8000+ [0m (Press CTRL+C to quit)
+ [32mINFO+ [0m: Started reloader process [+ [36m+ [1m13728+ [0m] using +[36m+ [1mStatReload+ [0m]
+ [32mINFO+ [0m: Started server process [+ [36m4532+ [0m]
+ [32mINFO+ [0m: Waiting for application startup.
+ [32mINFO+ [0m: Application startup complete.
```

- We will go the next link in our browser: <http://localhost:8000/docs#/default> and we will get a window like this:



- We will click in the POST section, prompting this window and click in the 'Try it out' button so we get to display the button to upload the file we want to be processed asynchronously.



- Now click in the button named 'Choose file' and upload the example file sent in the folder (named File1.csv or myFile2.csv) and then click on the button named 'execute' to upload the file and get the ID.

The image shows a Windows File Explorer window and a REST client interface. The File Explorer is open to the 'project' folder on the Desktop, showing files like File1, first_question, first_question_copy, myFile2, requirements, and second_question. The REST client is configured for a POST request to /process_csv/ with a file upload parameter.

File Explorer:

- Path: This PC > Desktop > project
- Files and folders:

Name	Status	Date modified	Type	Size
__pycache__		4/2/2023 10:06 PM	File folder	
File1		4/1/2023 4:19 PM	Archivo de valores...	2,732 KB
first_question		4/2/2023 9:13 PM	Python Source File	1 KB
first_question_copy		4/2/2023 9:35 PM	Python Source File	2 KB
myFile2		4/1/2023 9:04 PM	Archivo de valores...	2,522 KB
requirements		4/2/2023 8:56 PM	Text Document	1 KB
second_question		4/2/2023 8:52 PM	Python Source File	3 KB

REST Client:

- Method: POST
- URL: /process_csv/
- Parameters: No parameters
- Request body: multipart/form-data
- File upload: file (required), string(\$binary). Button: Choose File. Status: No file chosen.
- Buttons: Execute, Clear
- Responses: Curl

- If you scroll down you will see that the server states code 200 and in the details you get to see a green task ID, which is the one we will use for the next step.

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/process_csv/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@file1.csv;type=text/csv'
```

Request URL

http://localhost:8000/process_csv/

Server response

Code	Details
200	<p>Response body</p> <pre>{ "task_id": "529aea20-6961-4fca-a92e-ad660b302ccb" }</pre> <p>Response headers</p> <pre>content-length: 50 content-type: application/json date: Mon, 03 Apr 2023 03:14:26 GMT server: uvicorn</pre>

- Now copy the task_id without the quote marks. And now you can scroll down until you see the section with a blue button named 'GET' and display that window and again click in the 'Try it out' Button so you can enter the task_id to download the processed file.

GET /download_csv/{task_id} Download Csv

Parameters

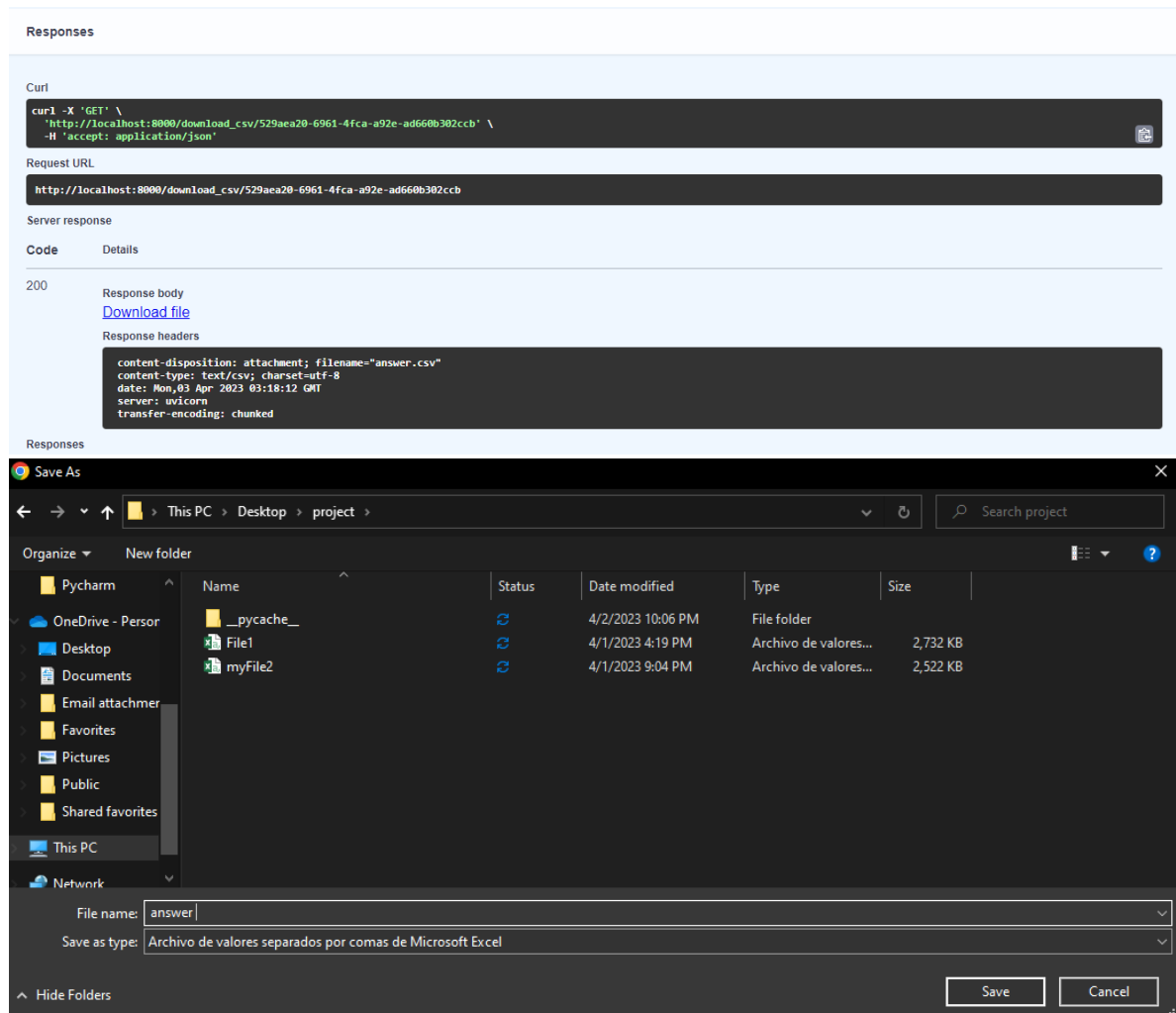
Try it out

Name	Description
task_id * required string (path)	task_id

Responses

Code	Description	Links
200	Successful Response	No links

- Now enter the task_id previously obtained and if you scroll down you will see that there is a blue text stating 'Download file', you can press it and download the processed file.



- And SUCCESS! You got an asynchronously processed .csv file

Important Notes:

- ❖ Remember to keep the cmd window running in the background so as to not close the created server.
- ❖ In the `first_question.py` I used a file already created, and also the output file is already named for easier visualization, but in the `second_question.py` file you can change the name of both .csv files by following the instructions given.