Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F21SC Industrial Programming
Type of assessment:	**Group**
Coursework Title:	Data Analysis of a Document Tracker
Student Name:	Salman Ansari
Student ID Number:	H00410360

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Anyuses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penaltiesthat I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature: *Salman Ansari*
Date: *08/12/2022*

Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F21SC Industrial Programming
Type of assessment:	**Group**
Coursework Title:	Data Analysis of a Document Tracker
Student Name:	Sasha Fathima Suhel
Student ID Number:	H00410394

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Anyuses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penaltiesthat I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature: *Sasha Fathima Suhel*

Date: *08/12/2022*

INDUSTRIAL PROGRAMMING
F21SC
Coursework 2
Data Analysis of a Document Tracker

Name: Salman Ansari and Sasha Fathima Suhel
HW ID: H00410360 and H00410394

Index

1. Introduction

We have created an application that analyses and displays the tracking data. The implementation, testing and discussion of the application is also discussed in the report. The program should do each task like working with huge datasets with 3 million lines in less than a minute. Data is presented in JavaScript Object Notation (JSON). The program is created using Python 3.10, PyCharm 2022.3 (PY- 223.7571.203), and Window 11 (22H2), and it is intended to work with Linux (Ubuntu 20.04).

The application is made to run from a standalone executable or a Python script when used in a Command Line Interface (CLI). The application's Graphical User Interface (GUI) can be launched from CLI. No inaccurate data is assumed and expected from this data

2. Requirement Checklist

The following are achieved in the application:

1) Python 3 & Ubuntu 20.04
2) View By Browser
3) View By Country
4) View By Continent
5) Reader profiles
6) Additional functionality of “Also likes”
7) Additional graph of “Also likes”
8) GUI Interface
9) Usage of Command Line Interface (CLI)
10) Standalone Executable file

3. Design Considerations

Code Structure

Since each task has its own class, the program is versatile and modular. With this configuration, implementing future changes are simple. Each Package functionality is applied in the Global Package folder. The python file “packages.py” contains all the libraries implemented globally separated by block frames.

These libraries are -

(Block 1) Feed Thread Pool: This package contains a method that returns dataset block from FileIO package and pass them. Each task can input its own function into the same method because the method also accepts a specific function that specifies how to handle data. A dictionary containing the desired query results is the return value.

(Block 2) File IO: It is a file reading method which reads the JSON file. It reads it line by line but returns the output chunk by chunk. All the tasks would use this method to read the files.

(Block 3) Nothing Exception: Salty_Exception class is inherited from Exception class. This exception will be used at task 2 and 3. If there is nothing in the resulting dictionary, we do not have to plot a graph.

(Block 4) Plot Bar chart: Both task 2 and 3 use Matplotlib to plot graph. We added another extra function to plot the histogram to the original task. The loading() is used to show a spinner while analysing the data. The wrap_function() is also added to record the duration for plotting and analysing the data.

(Block 5) Threading: This place is where the chunks are processed. Since there is a Global Interpreter Lock between the interpreter and program thread, this thread pool cannot boost processing speed. Instead, it will start with the same number of threads as the CPU on the computer.

(Block 6) Top N: It is a method that returns the top n items in the given dictionary. The task 2 requires returning the top 10 items from the dictionary.

Every common component of any activity is nearly entirely extracted, making it simple to maintain and add new functionality. Each job is autonomous and will not manage a thread pool or file; instead, it will only focus on the data.

Data Structure

The dictionary is the greatest option for tasks 2, 3, and 4 because the output is a two-dimensional figure, but it may not be the best option for tasks 5 and 6 depending on how many different types of documents are often read by a reader on average. For tasks 2, 3, and 4, there

is only one value for each key, hence the time complexity of creating and reading is O(1). After computing the hash key, we can then immediately retrieve the one value by comparing the hash key. One user will, however, relate to one or more documents in tasks 5 and 6. Dictionary is an excellent option if the reader's typical range of document types is limited. Additionally, since we don't want the same reader to read the same article several times, the Set is ideal for the task since it can avoid duplicate values, and we can mix Sets with a dictionary because the value of each dictionary entry is a set.

Multi-Threading

To prevent the GUI from "freezing" while a job is being executed, we have decided to construct the program utilizing many threads. Due to Matplotlib's lack of multithreading capability, we can only use multithreading for tasks 4, 5, and 6 when using the GUI, which allows users to begin tasks 4, 5, and 6 simultaneously without having to wait for one process to complete before beginning another.

Data Input Batch

Since the incoming JSON datasets may be rather huge (600k – 3M lines). Ten thousand lines of input are read and processed at once in a batch. As a result, less RAM is required to store the dataset.

Error and exception Handling

Compared to CW1, there are more errors and exception handling because of the sophisticated inputs. Producing a graph for tasks 2 and 3 takes time, thus we want to inform the user and forego plotting the chart if there are no results in the result dictionary. There are two basic outcomes for job number five: either the user id provided is right, or there is no user id provided or the user id provided is erroneous.

Help Function

Both the CLI and the GUI have help options available in the software. When using the command line, entering "-h" or "—help" will show the terminal's input choices. When tapped, a dedicated button on the GUI displays the search instructions in the display window.

```
def help():
    print("###########################################################################################\n")
    print("options:")
    print("\t-u  --userID: input the userID that need to query (Optional)")
    print("\t-d  --docID: input the documentID that need to query")
    print("\t-t  --taskID: input one of the task that want to run, [2,3,4,5,6,7]")
    print("\t-f  --filename: input the JSON file that contains the data")
    print("\t-h  --help: ask for help")
    print("\t-q  --quit: exit immediately\n")
    print("###########################################################################################")
```

Figure 1 Help options for GUI

4. User Guide

Command Line Usage

The application is made to function under Ubuntu 20.04. The Ubuntu system must have Python 3 and all required libraries installed before launching the program. The instructions to execute the application in Ubuntu's CLI, or terminal, are listed below:

1. Go to the project folder where the python file is located. In this instance for the file directory.
2. Run the Python script using the input options in the format given in the picture below once you are in the folder directory:

```
salman@salman-VirtualBox:~/Documents/IP$ python3 cw2.py -f sample_3m_lines.json
-d 100528230144-02f68abad20e46449b72482dce6a06a4 -u 9ab3093f8cef3be4 -t 7
```

Figure 2 Command Line Input

The name of our primary Python script file is "main.py". The following input choices are available:

-f file_name: is the name of the.json file containing the dataset. Put the file's name in lieu of "file name" (including .json file extension).

-d doc_uuid: is the document ID number to be looked up. Replace "doc uuid" with the ID of the desired document.

-u user_uuid: The user ID number that will be looked up is. Change "user uuid" to the relevant user's ID number.

-t task_id: is the task ID for the upcoming action. Each assignment specified by the curriculum requirements has a unique number. Substitute the desired task's ID number for "task id". Table 1 lists the task IDs and their purposes.

3. A JSON filename is required for each job. A document ID is needed for tasks 2, 5, and 6. A user ID may be needed for tasks 5d and 6. Graph-generating tasks launch a new

window with the specified graph in it. Text-outputting tasks will be displayed in the terminal.

4. As seen in figure 3, adding "-help" to the end of a Python script's name activates the help function.
5. Tasks performed by each task is given below:

Task ID	Task Performed
2(a)	Show the number of instances and the country/countries' histograms for a specific document.
2(b)	Show the number of occurrences and a histogram of the continents for the provided document.
3(a)	Show a histogram of all viewer browser identifiers.
3(b)	Show a histogram of the viewer's primary browser names.
4	Produces a list of the top 10 readers based on the overall user reading time.
5	Shows the documents that a given document "also likes." The "also likes" feature indicates which other documents the current document's reader has read.
6	Show the "also likes" function's histogram.
7	Opens a GUI window that may be used to carry out the tasks. When the GUI is opened, Task 6 is also completed in accordance with the coursework specification.

GUI Usage

As stated under CLI use, enter "-t 7" to start the GUI in the command line interface.

This figure illustrates the GUI's three input boxes for the user ID, document ID, and file path. The buttons that carry out activities 2a through 6 are located underneath the input areas. The output textbox occupies the majority of the GUI's center. At the bottom of the window are buttons for assistance and escape. The process will be completed by pressing any search buttons after entering the file path and IDs.

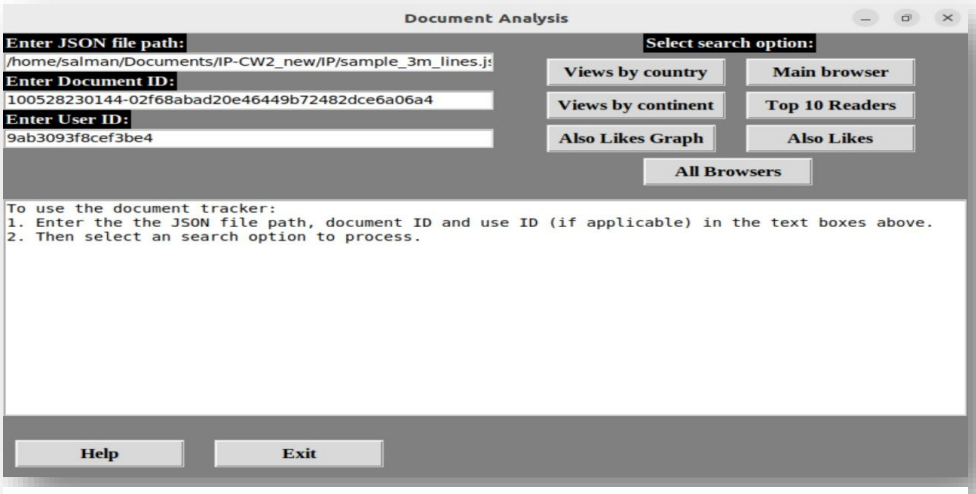

Figure 3 GUI interface

5. Developer Guide

Libraries

We have used many different libraries in this program like:

- *JSON* library is used for processing the JSON datasets
- *Pandas* library is used to read from the dictionary the country and continent.
- *Matplotlib* library is used in this application to plot chart.
- *Tkinter* is used for the GUI development of the application.
- *Graphviz* package is to visually see the .dot graph and automatically convert into pdf.

6. Main File

This file mainly provide function to achieve CLI interactions.

```
opts, args = getopt.getopt(sys.argv[1:], 'u:d:t:f:hq',
                           ['userID=', 'docID=', 'taskID=', 'filename=', 'help', 'quit'])
```

Code 1 Input Titles

FileIO

Blocks() is the only method in the Global Package file, and its size can be adjusted to fit the available memory. Only the Global Package's json_threadpooling() method uses this approach. Yield allows us to save a point without having to keep a separate variable to track the number of lines. We simply need to read the data from the storage device and load it into memory,

therefore the speed limitation is how quickly the storage can read, which is why we don't need to employ multithreading in this situation.

Feed Thread Pool

Each task will immediately contact the one method in the Global Package file, json_threadpooling(). Since we simply give a reference rather than a value into this method, the filter method parameter, which determines how to process data, does not return anything but rather shares a single result dictionary. Since there is no file exception verification when using the GUI, we should double-check the file name and type when using this approach.

Threadpooling

The threadpool execute() method is also contained in the Global Package file. Using this method, we will divide the dataset into many smaller tasks, with the only distinction being that each task will be processed by a different thread. So, to generate a lengthy tuple in args, list comprehension is required. We must first unzip those tuples in the map higher-order function so that the process method can identify them. It depends on the number of CPU cores whether to divide by 8 in this case. We can assume that the size must be less than 10k and that it is small enough to be a single task for the thread pool if the data size cannot be divided by 8 precisely.

TopN

Topn() is a competitive programming method. We can assume it only has a linear time complexity of O(NlogM), where N is the size of the data and M is the number of top items, if the number of top items is significantly less than the size of the data. In our situation, N is approximately one million, but M is only 10, so we can quickly retrieve the top 10 items.

Views by Country/Continent

Since the JSON dataset only contains nation codes. We utilized a dictionary that contains data on all nations and their country codes. This dictionary is used to hold the count of each country code after the document ID and event type have been matched during a search for a certain document ID. While we may simply express the result dictionary in job 2a, replacing the nation code with the word "continent" requires a different approach in task 2b. All.csv, a third-party resource that includes each country code and its continent, was used in this instance. We then extract that data

into a new dictionary, where the continent serves as the value and the country code as the key. Finally, it is simple for us to replace the nation.

```
table = pd.read_csv(os.path.abspath(r"Task2/all.csv"))
country_to_continent = table.set_index('alpha-2')['region'].to_dict()
results = feed_json_threadpool.feed_json_into_Threadpool(0, docID, filename, process_method)
new_results = {}
# replace the country to continent
for x in results:
    new_results[country_to_continent.get(x)] = results.get(x)
del results
return new_results
```

Code 3 Takes histogram title from all.csv file

Top 10 Reader Profiles

By total reading time, this tool displays the top ten users. Top readers are noted and listed along with how much time they spent reading. In this task, we can categorize the "page read time" event type by user ID and obtain the top 10 readers by using the TopN package.

Views by Browser

The only difference between tasks 3 and 2 is how the user-agent is handled. Many browsers will impersonate other browsers for historical reasons to obtain servers' higher quality code. One user-agent string will have a variety of browser names due to this. However, by using a different priority filter, we can still obtain the true browser name. For instance, Chrome frequently impersonates Safari, whereas Safari never does the same. In this situation, we can first see if the user-agent contains a "Chrome" substring, and only then, a "Safari" substring.

```
elif Safari.search(string):
    answer = "Safari"
elif Firefox.search(string):
    answer = "Firefox"
elif IE.search(string):
    answer = "IE"
```

Code 4 Browsers

Also Likes

To perform this function, follow these three steps. The first step is to establish the relationship between the user and the document, which is then kept in a dictionary. While we are processing the data, the second step is to note which user read the given document ID in a set. Finally, just use the set to filter the result dictionary. As a result, regardless of whether a user ID is provided, only readers who have read the provided document will proceed.

We once more use the topN function because task 5d calls for the top 10 also-liked documents as the default sorting algorithm. However, we must first determine how many of each document there are; we have talked about two scenarios in design consideration.

Also Likes Graph

The parent node is printed first, followed by each of its child nodes, precisely like in order traversal of a n-ary tree.

GUI

The GUI is a straightforward window with rows and columns. The GUI is made up of input text fields, labels, buttons, and an output textbox. Each command associated with a GUI button is related to a particular function.

This generates a button with a click action that launches the task 2a processing function for views by nation.

The file path, document ID, and user ID are read into the input text fields by each task processing function and passed to the appropriate task classes for processing. The results of processing the request are output in the output text box.

7. TESTING

Errors and Exceptions Testing Table

Tests	Outcome
General	
Wrong file name	Output "Cannot find file", filename, "because it does not exist."
Wrong file format	Output "Wrong file type:", filename, "not a json file."
File access denied	Output "Cannot open the file ", filename
No input file name	Output "No file input!"
No input task id	Output "No given task ID!"
Input wrong task id	Output "Unknown task ID", opt_value
Task 2	
Input wrong docID or not found given docID	Output "Nothing found in this file!", filename
Task 3	
Not found any reader	Output "Nothing found in this file!", filename
Task 4	
Not found any reader	Output "Not found any readers in the file" + filename
Task 5	
No input user ID or user ID does *not correspond* to the given	Output "Given user ID is not in the also likes user list, and will be ignored"

document ID	
Not found given document ID	Output "There is no user read this document " + docID + " in the file " + filename
Only one reader for given document ID	Output "There is only one user read this document!"
Task 6	
No input user ID or user ID does not corre*spond* to the given document ID	Output "Given user ID is not in the also likes user list, and will be ignored"
Not found given document ID	Output "There is no user read this document " + docID + " in the file " + filename
Only one reader for given document ID	Output "There is only one user read this document!"
Task 7	
No input file name	Output "No filename entered"
No input document ID	Output "No doc ID entered"
Wrong file name	Output "Cannot find file", filename, "because it does not exist."
Wrong file format	Output "Wrong file type:", filename, "not a json file."

Performance Testing Table

File Size	*Average Execution Run-time (s)*
100k lines	*0.10*
400klines	*14.2*
600k lines	*20.3*
3 million lines	*35.6*

The below date shows the time taken to render and process every histogram for "view by country", "view by continent" etc. This function takes place in Global packages. When the button from the GUI is clicked, it calls the block for "plot barchart" and computes the following for individual buttons.

```
Analyzing data duration(s): 71.17949962615967
Plotting data duration(s): 0.6421010494232178
Start analyzing data...\
Analyzing data duration(s): 57.71460771560669
Plotting data duration(s): 18.75513243675232
Start analyzing data...|
Analyzing data duration(s): 39.63824009895325
Plotting data duration(s): 10.775172233581543
Start analyzing data...|
Analyzing data duration(s): 34.03207540512085
Plotting data duration(s): 11.822808265686035
Start analyzing data...-
Analyzing data duration(s): 40.34984040260315
Plotting data duration(s): 20.216696977615356
Start analyzing data...
```

Code 5 Time duration

8. Programming Language and Implementation

Python's syntax is straightforward and very readable. It is made to be simple to learn. End statements and curly braces are replaced with indentation in their place. Variables can be defined by Python without having to be expressly declared by the user. The fact that Python "interpreted" rather than "complied" is another trait.

We discovered that the decorator is quite helpful for sophisticated language features; one function can have one or more decorators, and they are performed in order from bottom to top. Python's performance in comparison to "complied" languages is one of its drawbacks. Comparing C# from coursework 1 to other prominent languages like C, C++, and Java, C# is highly comparable to those languages, making it more familiar to programmers coming from such languages. Python is more distinctive and first requires a bit more getting used to.

9. Discussion from CW1

The majority of CW1's critical criticism is related to report authoring and code quality.

Coding

This program's goal was to have well-organized, legible code. To improve organization and navigation, each task is divided into packages. For the jobs, appropriate data structures were selected. There were less repeatable codes compared to CW1.

Report

The report is kept clear and succinct. To facilitate understanding, each part is well labeled with figures and tables. Testing has improved and now includes more failures and exceptions. This time, the developer manual is more comprehensive. The topic about language reflection and application now includes more detail.

9. Conclusion

In conclusion, we are pleased with the program's overall implementation. The program passed our testing without any bugs and all requirements were satisfied.

By implementing "real" multithreading, improvements can be made to the program's performance. This can be done using a different language or a separate multiprocessing-capable machine.

References

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser_detection_using_the_user_agent
- https://stackoverflow.com/questions/9847580/how-to-detect-safari-chrome-ie-firefox-and-opera-browser
- https://docs.python.org/3/library/tkinter.html