HERIOT-WATT UNIVERSITY DUBAI
SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

# F21SC - Industrial Programming 2023-24

# Coursework 2 - Data Analysis of a Document Tracker

| Student ID Number | Student Name | Programme |
|---|---|---|
| H00441124 | Ilia Svetlichnyi | MS Data Science |
| H00451036 | Daniil Alekseev | MS Software Engineering |

## 1. Introduction

The main purpose of this report is to give a clear picture of how we designed, developed, and implemented our data-intensive application using Python 3. The focus was to create an application that is not only effective in handling and analyzing large volumes of data but also user-friendly. We aimed to include important features that would make the application practical and convenient for users, such as data visualization tools, option to choose a certain question and input any document_id which the user wants.

## 2. Requirements Checklist

All the required features have been implemented successfully with all needed functionality. Here is the list of them:

- **Views by Country/Continent:** Analyzes and displays histograms showing from which countries and continents documents are viewed**.**
- **Views by Browser:** Identifies popular browsers by counting occurrences in visitor data and displays the results in a histogram.
- **Reader Profiles:** Lists the top 10 readers and calculates the amount of time spent reading overall to determine who is the most passionate reader.
- **"Also Likes" Functionality:** Suggests related documents based on user reading habits and lists the top 10 related documents.
- **"Also Likes" Graph:** Generates a graph to visually represent the relationships between documents and their readers.
- **Command-line Usage:** Provides a command-line interface for automated testing of functionalities, allowing inputs like user UUID, document UUID, task ID, and file name for task execution.

## 3. Design Considerations

In developing our data analytics application, we carefully considered several key aspects to ensure functionality and user-friendliness:

**Programming Style:**

- **Functional Programming:** Unlike object-oriented programming, we employed a functional programming style throughout the project. This means the entire code is structured around functions, making it simpler and more straightforward, especially for handling data analysis tasks.
- **Separation of UI and Logic:** It was crucial to maintain a clear separation between the user interface and the underlying logic, streamlining updates and maintenance. That's why we created individual files for each task and combine them in the main.py

**Data Structures:**

- **Data Management:** We used appropriate data structures, like lists and dictionaries, for efficient handling and analysis of viewer data, such as country and browser information.

**GUI Design:**

- **Tool Selection:** For graphical representation of data, we integrated libraries like matplotlib for histograms and graphviz for graph creation, ensuring clarity and interactivity in data visualization.

- **Tkinter Library:** For the user interface, we chose the Tkinter library. Its simplicity and effectiveness allowed us to create intuitive and user-friendly graphical interfaces for displaying histograms, graphs, and handling user inputs.

**Advanced Language Constructs:**

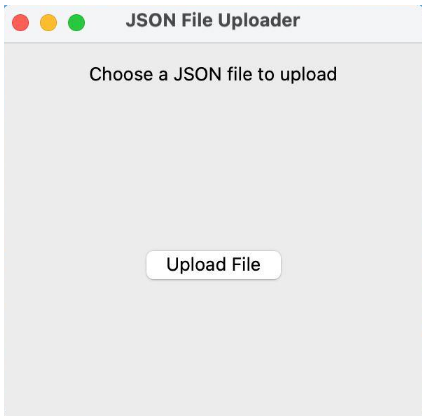
- **Event Handling and Data Processing:** Our program utilizes Python's advanced features for event handling and data processing, allowing it to respond dynamically to user inputs and efficiently process large datasets.

**Performance Choices:**

- **Efficient Data Handling:** To ensure smooth performance, we incorporated strategies like data filtering and memory management, especially in features like "Also Likes" graph and browser views analysis.

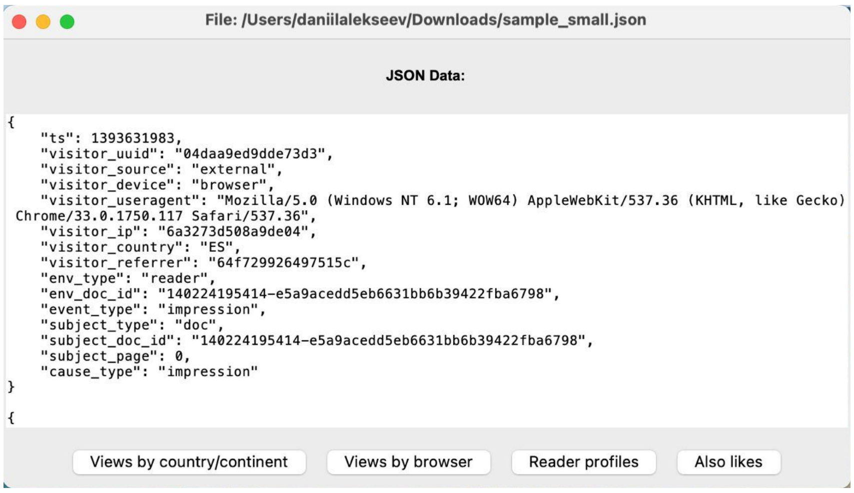
## 4. User Guide

The Tkinter application is designed with user-friendliness in mind, ensuring smooth navigation and usage of its features.
As soon as the application is running the initial application window shows up where you need to choose a JSON file on your computer to proceed.



*The initial application window*

After a specified JSON file is chosen the home screen of the application pops up where in the middle you can see representation of the file and four main buttons for specific tasks.

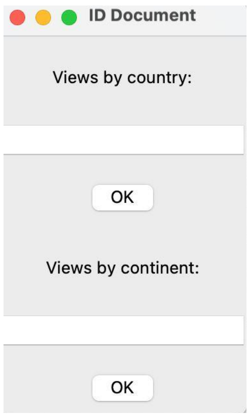

File: /Users/daniilalekseev/Downloads/sample_small.json

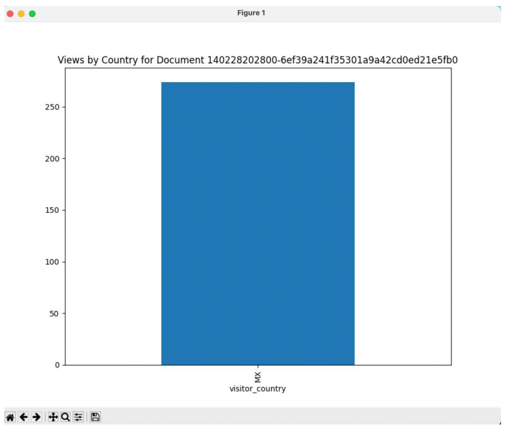
JSON Data:

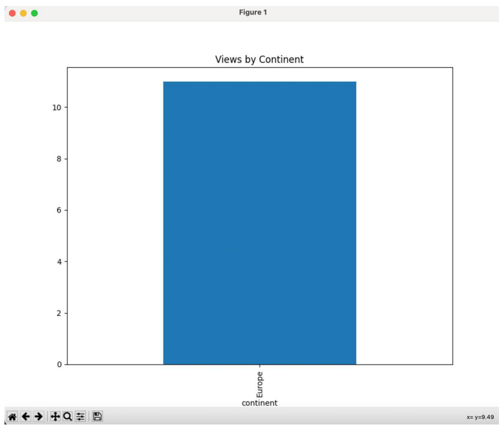
```
{
    "ts": 1393631983,
    "visitor_uuid": "04daa9ed9dde73d3",
    "visitor_source": "external",
    "visitor_device": "browser",
    "visitor_useragent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/33.0.1750.117 Safari/537.36",
    "visitor_ip": "6a3273d508a9de04",
    "visitor_country": "ES",
    "visitor_referrer": "64f729926497515c",
    "env_type": "reader",
    "env_doc_id": "140224195414-e5a9acedd5eb6631bb6b39422fba6798",
    "event_type": "impression",
    "subject_type": "doc",
    "subject_doc_id": "140224195414-e5a9acedd5eb6631bb6b39422fba6798",
    "subject_page": 0,
    "cause_type": "impression"
}

{
```

*The home screen of the application*

**Views by country / continent:** After pressing the corresponding button in the main menu, you will see the part where you can input the chosen document id and by clicking the "OK" button a certain histogram pops up.

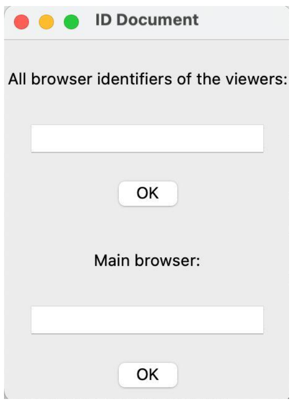


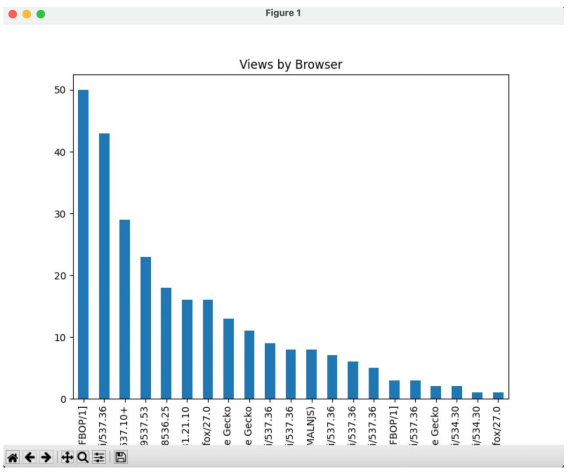


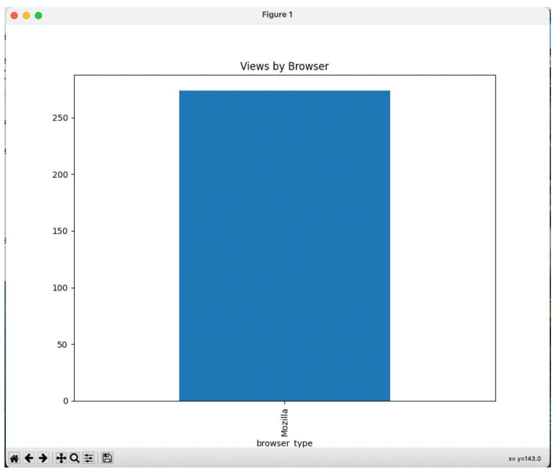


*The functionality of the views by country / continent task*

**Views by Browser:** This function has two options of graphs which you can specify in the corresponding window after pressing the button on the home screen. The first graph shows you the raw source from which the user came from and the second one shows the same information, but with cleaned information about sources.



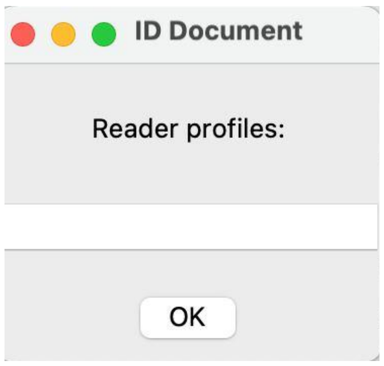


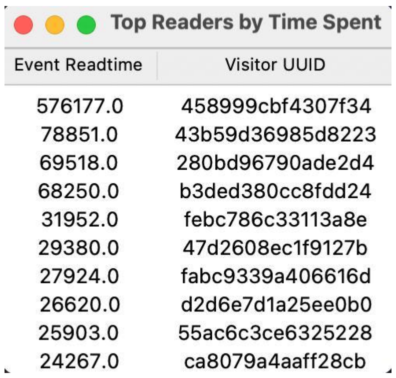


*The functionality of the views by Browser task*

Visual data analysis in Python
[https://www.kaggle.com/code/kashnitsky/topic-2-visual-data-analysis-in-python]

**Reader Profiles:** This function allows you to see the table of top readers for the chosen document_id. To execute it you just need to input the chosen doc_id in the first window and the table will pop up.



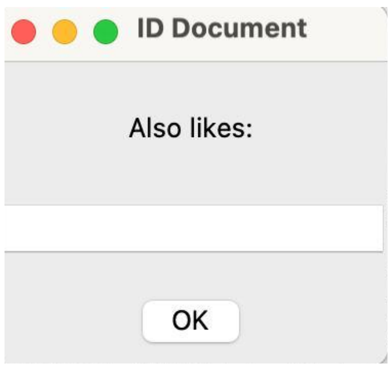


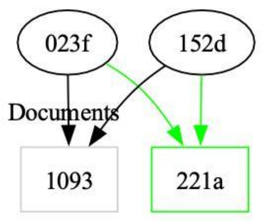
Top Readers by Time Spent

| Event Readtime | Visitor UUID |
|---|---|
| 576177.0 | 458999cbf4307f34 |
| 78851.0 | 43b59d36985d8223 |
| 69518.0 | 280bd96790ade2d4 |
| 68250.0 | b3ded380cc8fdd24 |
| 31952.0 | febc786c33113a8e |
| 29380.0 | 47d2608ec1f9127b |
| 27924.0 | fabc9339a406616d |
| 26620.0 | d2d6e7d1a25ee0b0 |
| 25903.0 | 55ac6c3ce6325228 |
| 24267.0 | ca8079a4aaff28cb |

*The functionality of the reader profiles task*

**"Also Likes":** If you want to see connections between users and documents which they read, you can use this function which generates the graph with all the information. As always you just need to input a specific doc_id and a graph for it in pdf format will pop up.





*The functionality of the "also likes" task*

# 5. Developer Guide

**Application Design Overview:**

In our data analytics application, we've adopted a functional programming style, which means the code is organized around functions rather than objects or classes. This approach simplifies the code and enhances readability, especially beneficial for handling various data analysis tasks. The GUI is designed using the Tkinter library, providing a straightforward and effective way to interact with the user.

**Architecture:**
The application's architecture is modular and organized around specific tasks:

**Main Script (main.py):** This is the primary entry point of the application. It orchestrates the overall functioning and integrates various functionalities.

**Data Analysis Modules:** Separate Python scripts handle specific data analysis tasks:

- **views_by_country.py and views_by_continent.py:** These modules analyze document views by country and continent, respectively, and generate histograms.
- **views_by_browser_clear.py:** Processes browser data to determine popularity.
- **views_by_visitor_useragent.py:** Examines visitor user-agent data to provide insights on source usage.
- **reader_profile.py:** Calculates the total reading time per user to identify the most avid readers.
- **also_like_graph.py:** Generates the "Also Likes" graph visualizing document-reader relationships.

**Command Line Interface (main.py):** The part of the code for Managing the command-line interface that allows automated testing and execution of different tasks is integrated in the main file.

**Code Fragment:**

**Shows the data analysis approach (from views_by_continent.py):**

```
def views_by_continent(data, document_id):
    # Function to get continent from country code using pycountry_convert
    def country_code_to_continent(country_code):
        try:
            continent_code = pc.country_alpha2_to_continent_code(country_code)
            return pc.convert_continent_code_to_continent_name(continent_code)
        except:
            return "Unknown"  # for codes that are not found or invalid

    # Filter data for the specific document_id
    data_for_document = data[data['env_doc_id'] == document_id].copy()

    # Apply the function to your DataFrame
    data_for_document.loc[:, 'continent'] = data_for_document['visitor_country'].apply(
        country_code_to_continent)

    # Group by continent and count views
    views_by_continent = data_for_document.groupby(
        "continent").size().sort_values(ascending=False)

    # Plot the views by continent
    views_by_continent.plot(kind="bar", figsize=(9, 7), title="Views by Continent")
    plt.show()
```
Exploratory Data Analysis with Pandas
[https://www.kaggle.com/code/kashnitsky/topic-1-exploratory-data-analysis-with-pandas]

## 6. Testing

**Manual tests:**
**1. Test Case: Launch Application**
- *Description:* Launch the data analytics application.
- *Result:* The application opens successfully, displaying the initial interface or command prompt without errors.

**2. Test Case: Views by Country Analysis**
- Description: Input a document UUID to analyze views by country.
- *Result:* The application displays a correct histogram of countries from the given data.

**3. Test Case: Views by Continent Analysis**
- *Description: Enter a document UUID to analyze views by continent.*
- *Result:* The application generates a histogram of continents accurately based on the input data.

**4. Test Case: Views by Browser Analysis**
- *Description:* Analyze browser usage from the visitor user-agent data.
- *Result:* A histogram displaying browser popularity is correctly produced.

**5. Test Case: Reader Profiles Analysis**
- *Description:* Identify the top 10 most avid readers.
- *Result:* The application lists the top 10 readers based on total reading time accurately.

**6. Test Case: “Also Likes” Graph Generation**
- *Description:* Input a document UUID to create an “also likes” graph.
- *Result:* A graph illustrating the relationship between documents and readers is accurately generated.

**7. Test Case: Command-Line Interface**
- *Description:* Use the command-line interface to execute different tasks with specified parameters.

   - *Result:* The application responds correctly to the command-line inputs and performs the desired tasks.

**8. Test Case: GUI Functionality**
   - *Description:* Interact with the Tkinter-based GUI for various data analysis tasks.
   - *Result:* The GUI operates smoothly, allowing for efficient input and clear visualization of results.

**9. Test Case: Error Handling and Validation**
   - *Description:* Input invalid or incomplete data for various tasks.
   - *Result:* The application provides appropriate error messages or feedback, ensuring user awareness of incorrect inputs or processing issues.

## 7. Reflections on Programming Language and Implementation

In developing our data analytics application, we deepened our understanding of Python's capabilities. Python proved to be an excellent choice for this project due to its simplicity and powerful data processing abilities. Its extensive libraries, such as Pandas for data manipulation and matplotlib for data visualization, made complex tasks more manageable. The language's straightforward syntax and readability were especially beneficial in implementing functional programming concepts, allowing for clear and concise code. This experience reinforced our view of Python as a versatile tool, particularly suitable for data-intensive applications.

**Useful Language Features and Technologies:**

**Python's Dynamic Typing:** Python's dynamic typing was advantageous for our project. It allowed for more flexibility and faster development, especially useful in a data analytics context where data types can vary.

**Extensive Library Support:** Python's vast ecosystem of libraries, such as Pandas for data manipulation and matplotlib for visualization, was instrumental in our project. These libraries simplified complex tasks, enabling us to focus on the application's functionality.

**Limitations of the Application:**

**Graphical Capabilities:** Using Tkinter for GUI limited the application's visual appeal and interactivity compared to more advanced GUI frameworks.

**Performance Concerns:** Python, being an interpreted language, can have performance limitations, especially when handling extremely large datasets or performing complex data processing tasks.

**Ways to Overcome the Limitations:**

**Enhanced GUI Frameworks:** For more sophisticated user interfaces, considering GUI frameworks like PyQt or Kivy could enhance the application's visual and interactive aspects.

**Performance Optimization:** Utilizing Python’s advanced features, such as asynchronous programming and multiprocessing, could improve the application's performance. Additionally, integrating Python with optimized, lower-level languages like C could be explored for performance-critical components.

In considering this project, it is clear that Python's robust data processing capabilities and adaptability were essential to its development. Although there is always room for development, especially in GUI design and performance, the project was made much easier by Python's vast library and ease of use. Our understanding and appreciation of Python in data-intensive applications have been increased by this experience.

## 8. What did I learn from CW1?

In CW1, we learned the importance of clear and structured coding. It became evident that well-organized code not only simplifies development but also aids in future maintenance and updates. Additionally, we realized the crucial role of testing in the development process. Effective testing helps catch errors and issues early, ensuring a more robust and reliable application. This experience has significantly influenced our approach to coding and testing in subsequent project.

## 9. Conclusions

We are happy with how our data analytics project turned out. The application effectively achieves its goal by using data analysis to provide insightful information. For this kind of application, using Python and its functional programming methodology together with tools like Pandas and matplotlib proved to be useful.

The application's creation process was demanding but satisfying, providing a thorough understanding of data management and visualization methods. Even though we didn't employ an OOP-based methodology, Python's functional programming style was appropriate for our needs, especially when it came to data processing and analysis jobs.

For future improvements, we could look into enhancing the graphical user interface, possibly by exploring more advanced GUI frameworks. Additionally, optimizing the application's performance for handling even larger datasets more efficiently would be a valuable direction.

In summary, this project not only achieved its goals but also provided a great learning experience in Python programming and data analytics. It opens up possibilities for further development and refinement in the future.

Here is the link to the video of running the application:
[https://drive.google.com/drive/folders/1pzUrGcRaXPPxLp2nAvR_YyFEBFzcM49X?usp=sharing](https://drive.google.com/drive/folders/1pzUrGcRaXPPxLp2nAvR_YyFEBFzcM49X?usp=sharing)

## 10. References

1. Tkinter Documentation [https://docs.python.org/3.11/library/tk.html]
2. Python Documentation [https://docs.python.org/3.11/tutorial/index.html]
3. Python GUI Programming With Tkinter [https://realpython.com/python-gui-tkinter/]
4. Exploratory Data Analysis with Pandas [https://www.kaggle.com/code/kashnitsky/topic-1-exploratory-data-analysis-with-pandas]
5. Visual data analysis in Python [https://www.kaggle.com/code/kashnitsky/topic-2-visual-data-analysis-in-python]