



MAPÚA UNIVERSITY

SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 1:

Using Software Tools and Code Versioning System

CPE106L (Software Design Laboratory)

Brian Matthew E. Clemente

Maria Angelica Acantilado

Chalzea Fransen C. Dytianquin

Vance David G. Samia

Group No.: **4**
Section: **FOPI01**



PreLab

Readings, Insights, and Reflection

Professional Git (Brent Laster, 2016)

Chapter 1: What is Git, Git Ecosystem, Git-Hosting Sites

In the group's exploration of *Professional Git* by Brent Laster (2016), we examined the core principles of Git, its ecosystem, and popular Git-hosting platforms. Git is a powerful distributed version control system designed to streamline source code management, enabling developers to track changes, collaborate efficiently, and maintain multiple project versions with ease. Unlike traditional centralized version control systems, Git provides a decentralized approach that allows every contributor to have a complete copy of the repository, ensuring greater flexibility and reliability in software development.

The Git ecosystem is vast, comprising various tools and platforms designed to enhance version control workflows. Popular Git-hosting services such as GitHub, GitLab, and Bitbucket provide cloud-based repositories where teams can collaborate seamlessly. These platforms support essential features like branching, merging, pull requests, and issue tracking, which help streamline development and ensure code quality. Additionally, tools like SourceTree and various Git extensions improve usability by offering graphical interfaces and automation features, making version control more accessible for developers of all experience levels.

A crucial part of our study involved understanding different Git workflows, particularly the centralized and distributed models. In a centralized workflow, all contributors push their changes to a single repository, which serves as the authoritative source of the codebase. The distributed model, on the other hand, allows developers to maintain local copies of the repository, make changes independently, and synchronize their work through merging and pull requests. By learning these workflows, we have developed a strong foundation in best practices for using Git effectively within our team repository. Applying these principles ensures that our code contributions remain well-organized, reducing errors and improving overall efficiency in our development process.

Fundamentals of Python: First Programs (Kenneth Lambert, 2018)

Chapter 1: Basic Python Programming

Our reading of *Fundamentals of Python: First Programs* by Kenneth Lambert (2018) provided an introduction to the core principles of Python programming. We explored Python's syntax, data types, and operators, gaining an understanding of how the language handles variables and expressions. One of the key aspects we discussed was Python's emphasis on readability and simplicity, particularly its use of indentation to define code blocks instead of traditional braces. Additionally, we examined Python's dynamic typing, which allows variables to change types without explicit declarations, making the language more flexible and user-friendly.

Another important focus was Python's built-in data structures, including lists, tuples, and dictionaries. These structures enable efficient data storage and manipulation, playing a crucial role in problem-solving and algorithm development. We also reviewed control structures like loops and conditional statements, which help in automating repetitive tasks and making decisions within a program. Overall, the group recognizes both Python's advantages and limitations through our introduction to Python highlighted its versatility and ease of use, reinforcing why it is widely used in various fields, from web development to data science.

Git and Visual Studio Code

As part of our learning experience, we delved into three essential tools in software development and data science: Git, Anaconda, and Visual Studio Code (VS Code). Our hands-on exploration provided us with practical knowledge of version control, coding environments, and efficient collaboration, all of which are crucial skills in modern programming workflows. We practiced creating repositories, managing branches, and committing changes using GitHub, reinforcing our understanding of best practices in version control and teamwork.

Furthermore, the confines of Anaconda served as a structured environment for running Python scripts and managing dependencies. We explored its user-friendly interface, which simplifies launching Python and executing scripts within a controlled environment. While we have not yet extensively explored package installations, we gained a foundational understanding of how Anaconda facilitates Python development by providing a consistent workspace that prevents compatibility issues with different Python versions and libraries. This makes it an ideal choice for data science and machine learning projects that require extensive package management and environment configurations.

Likewise, Visual Studio Code (VS Code) proved to be a powerful and versatile code editor that supports multiple programming languages and extensions. We set up VS Code for Python development by configuring essential extensions, utilizing the integrated terminal, and leveraging debugging tools. Additionally, we practiced using VS Code alongside Git and GitHub, managing repositories through GitHub Desktop, and pushing and pulling changes to ensure smooth collaboration and efficient source control. The underscored repositories of the built-in Git integration in VS Code further streamlined our workflow, making it easier to track changes and resolve conflicts directly within the editor.

By exploring these tools, we gained valuable hands-on experience with essential software development practices. Our understanding of version control strengthened as we learned how to manage repositories and work collaboratively. Anaconda provided us with an organized approach to Python development, while VS Code enhanced our productivity through its versatile features. This learning experience not only expanded our technical skills but also deepened our understanding of best practices in software development and teamwork. As we continue to develop our expertise, these tools will play an integral role in our programming journey, helping us build more efficient and scalable projects.

Summary

Python's integrated data structures, such as dictionaries, tuples, and lists, play a vital role in data organization and processing. These structures enhance algorithm development and problem-solving by providing efficient ways to store, retrieve, and manipulate information. Their versatility allows developers to handle complex data efficiently, making them essential in various fields such as data analysis, machine learning, and web development. Additionally, control structures like loops and conditional statements enable automation, streamline repetitive tasks, and facilitate logical decision-making within programs. Through hands-on exercises and coding examples, we explored how these elements contribute to Python's flexibility and adaptability in software development, reinforcing our understanding of core programming concepts. This practical approach not only strengthened our technical proficiency but also allowed us to appreciate Python's readability and efficiency in writing clean and maintainable code.

Beyond the technical aspects, the group also discussed Python's widespread applications in modern computing, including web development, data science, artificial intelligence, and automation. Python's extensive library support and active developer community make it an indispensable tool in various industries, offering solutions for complex problems with minimal effort. The foundational principles covered in our studies have provided a strong base for delving into more advanced topics such as object-oriented programming, data visualization, and machine learning. Understanding these fundamentals allows us to apply Python to real-world challenges, showcasing its usability as both a beginner-friendly and powerful programming language. As we continue our learning, mastering these basics will support our ability to develop scalable and efficient software solutions, enabling us to build innovative projects and contribute effectively to professional development environments.

PostLab

Programming Problems

In this laboratory activity, we explored essential tools for software development and data science, specifically Git, Anaconda, and Visual Studio Code. The practical application of these tools allowed us to gain a deeper understanding of version control, coding environments, and collaborative workflows. The integration of the tools underscored during the pre-lab portion of the exercise not only strengthened our technical skills but also reinforced best practices in software development, version control, and project collaboration.

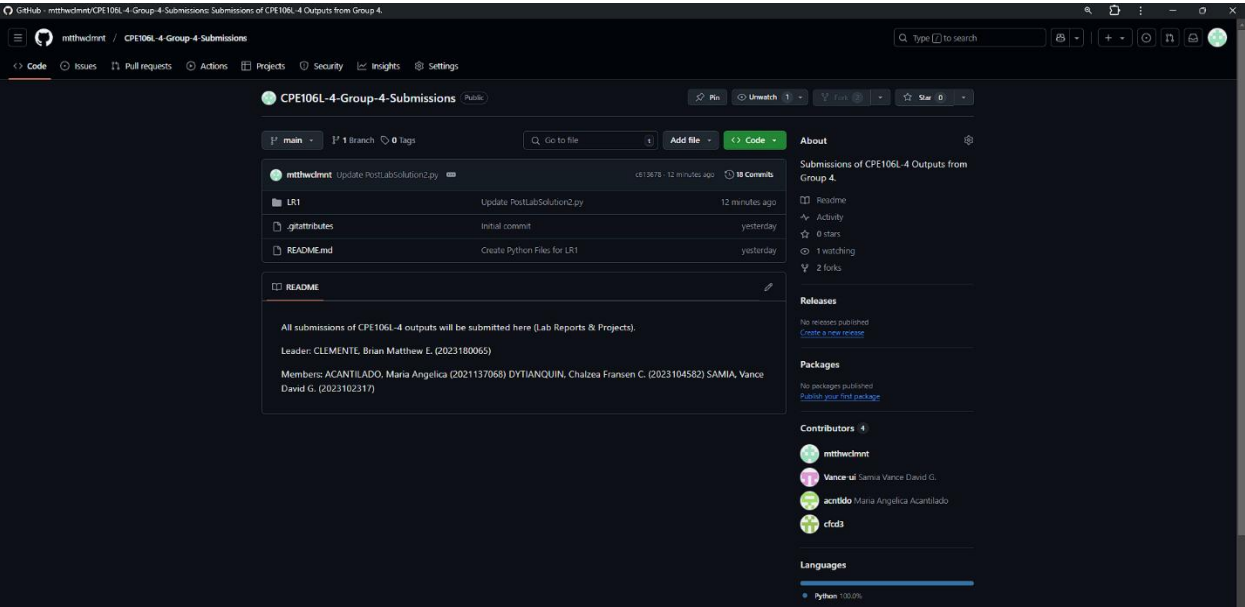


Figure 1.1. Group 4 GitHub Repository Preview

Figure 1.1 illustrates our group’s GitHub repository, showcasing the practical application of version control principles learned throughout the activity. The repository serves as a central hub for managing code, tracking modifications, and maintaining an organized workflow among team members. Through the creation of our main branch, committing changes, and pushing updates, we applied key Git functionalities that are essential for efficient software development. This structured approach to repository management ensured that all contributions were properly documented and reduced the risk of conflicts when working on collaborative projects, such as this one. The experience we gained from using GitHub enhanced our ability to work in professional development environments where version control is a critical component of effective software engineering practices.

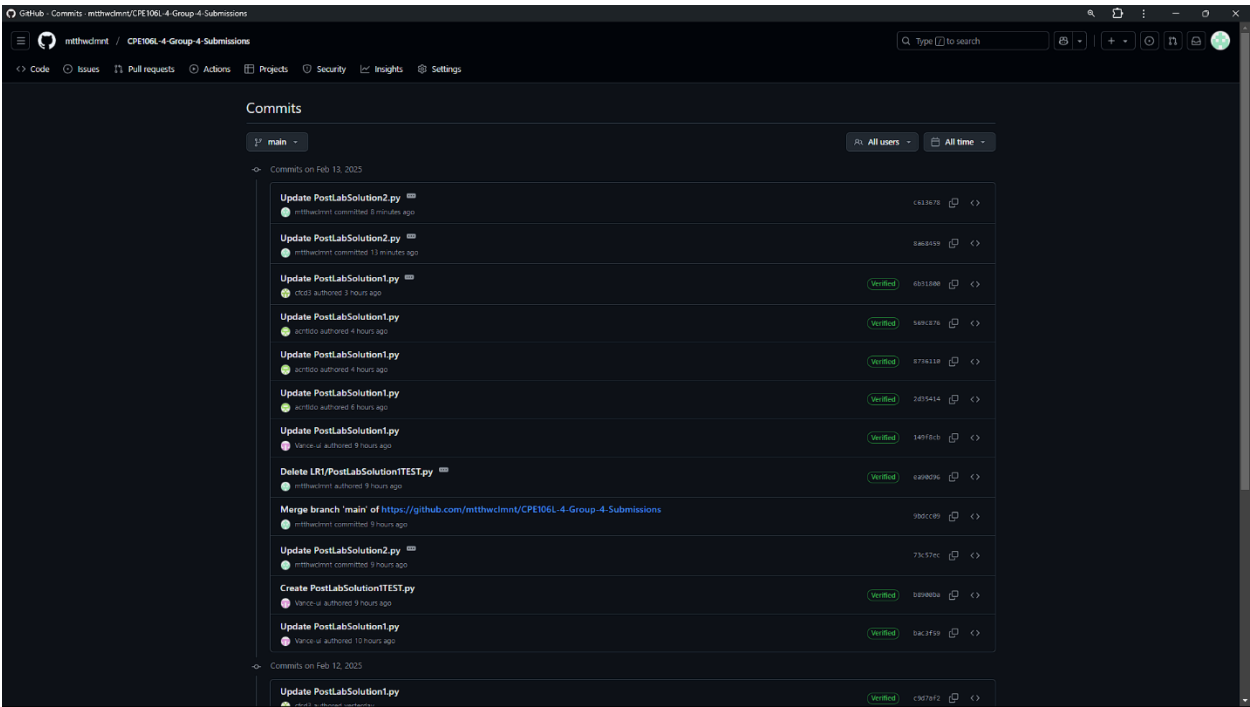


Figure 1.2. Lab Report 1 GitHub Repository Commits History Log (Screenshot 1)

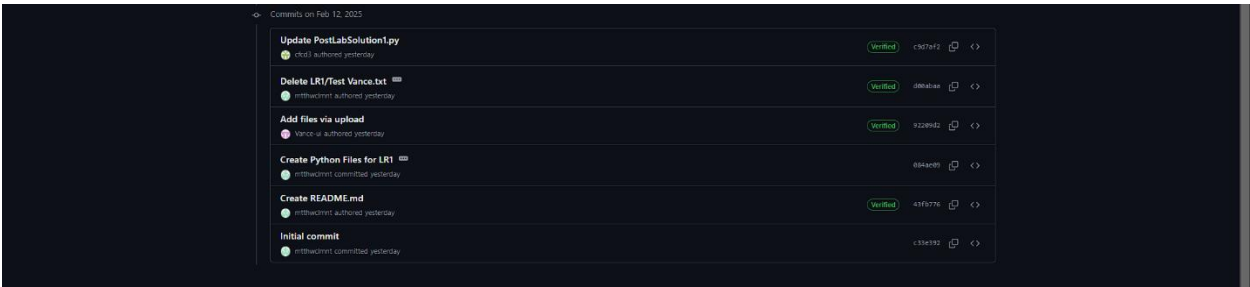


Figure 1.3. Lab Report 1 GitHub Repository Commits History Log (Screenshot 2)

Particularly, Figures 1.2 and 1.3 display the commit history logs of the GitHub repository, providing a detailed record of the modifications and contributions made throughout the laboratory activity. Each entry in the commit history represents a specific change to the repository, including file updates, additions, and branch merges, accompanied by corresponding commit messages. This systematic documentation ensures transparency, facilitates version tracking, and enables team members to revert to previous versions when necessary. Maintaining an organized commit history is essential in collaborative software development, as it allows for efficient debugging, accountability, and streamlined workflow management. By adhering to best practices in version control, this activity reinforced our understanding of Git's significance in professional development environments, where structured and well-documented repositories are fundamental to effective project coordination and long-term maintainability.

Programming Problem #1

The implementation of statistical functions such as mean, median, and mode is crucial in data analysis, providing essential measures of central tendency. For the first programming exercise, we focused on designing and implementing these functions within a dedicated Python module named stats.py. The objective was to create efficient and reusable functions that can process numerical data and return meaningful statistical insights. The mean function computes the arithmetic average by summing all elements in a list and dividing the result by the total number of values, offering a general measure of central tendency. The median function identifies the middle value in an ordered dataset, making it particularly useful for distributions with skewed values or outliers. The mode function determines the most frequently occurring number in a dataset, which is valuable in identifying patterns and trends.

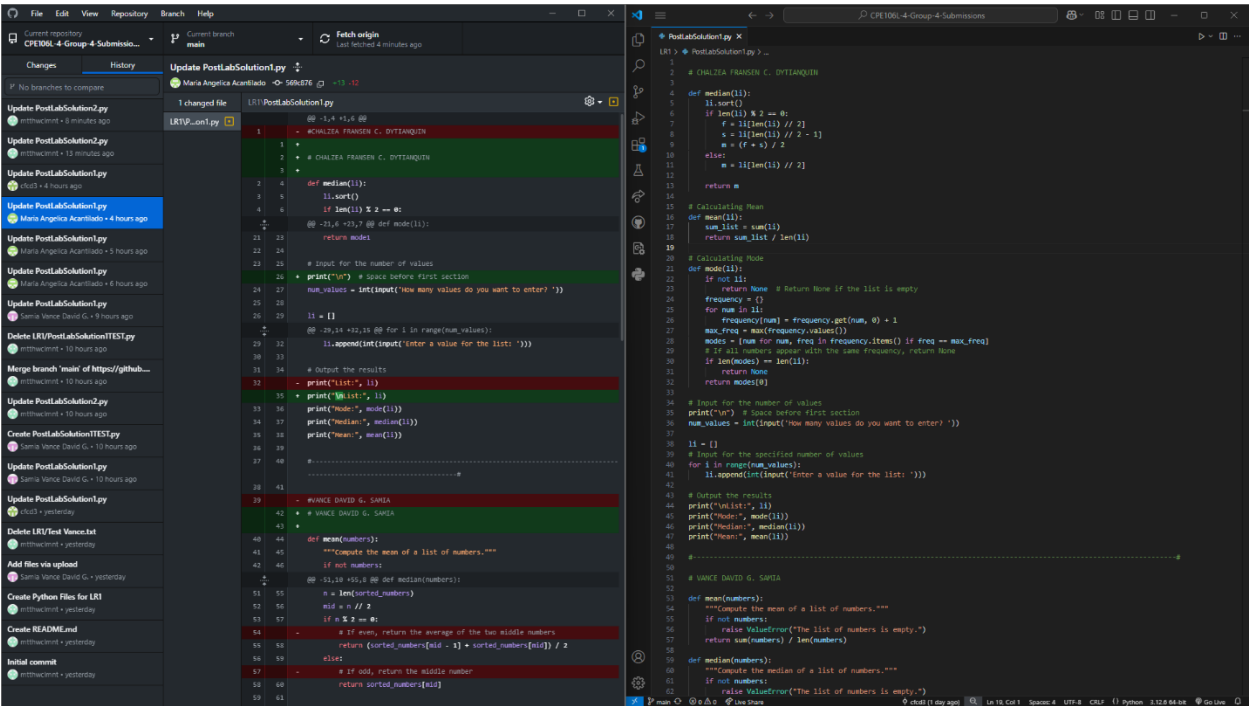


Figure 2.1 Screenshot of GitHub Desktop and VS Code open during Post Lab Solution 1 Commit and Push to Origin

Figure 2.1 provides a visual representation of the workflow followed in developing, testing, and managing the module. The left section of the screenshot displays GitHub Desktop, capturing the staged changes before committing and pushing them to the repository, ensuring version control best practices. On the right, Visual Studio Code is showcased as the primary coding environment, where the team members wrote, debugged, and refined their respective implementations. The collaborative nature of the development process was emphasized, with three group members contributing to their individual implementations of the functions. Each member provided a unique approach to calculating the statistical measures, allowing for comparison and refinement of the final solution, with these contribution sectioned out and labelled using comments within the

Python script. These contributions were systematically merged into a single optimized version, ensuring accuracy and efficiency.

```
(base) C:\Users\Hp>cd Documents\GitHub\CPE106L-4-Group-4-Submissions\LR1

(base) C:\Users\Hp\Documents\GitHub\CPE106L-4-Group-4-Submissions\LR1>python PostLabSolution1.py

How many values do you want to enter? 5
Enter a value for the list: 1
Enter a value for the list: 6
Enter a value for the list: 3
Enter a value for the list: 4
Enter a value for the list: 3

List: [1, 6, 3, 4, 3]
Mode: 3
Median: 3
Mean: 3.4

Enter the size of the list: 5
Enter value 1: 1
Enter value 2: 6
Enter value 3: 3
Enter value 4: 4
Enter value 5: 3

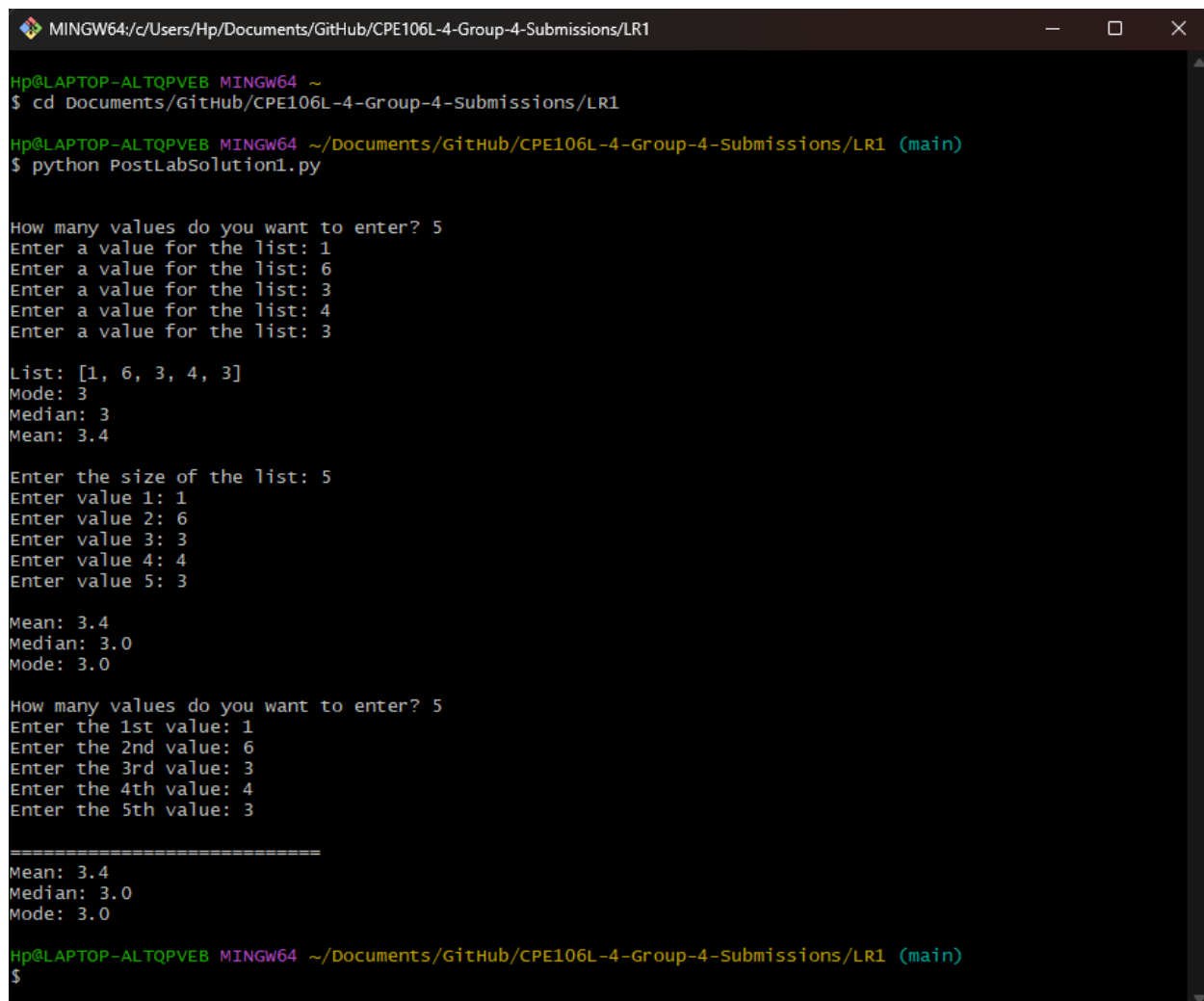
Mean: 3.4
Median: 3.0
Mode: 3.0

How many values do you want to enter? 5
Enter the 1st value: 1
Enter the 2nd value: 6
Enter the 3rd value: 3
Enter the 4th value: 4
Enter the 5th value: 3

=====
Mean: 3.4
Median: 3.0
Mode: 3.0
```

Figure 2.2 Post Lab Solution 1 Python Script and Sample Text File executed in Anaconda Prompt

The execution of the aforementioned module resulted in three distinct output layouts, reflecting the variations in implementation by the three contributing members. Each version of the program adhered to the core requirements of computing the mean, median, and mode but differed in terms of user interaction, formatting, and input handling. *Figure 2.2* showcases one such output, where the program prompts the user to input values and then displays the computed statistical measures in a structured format; it incorporates the consolidated codes of the three members that committed their respective repositories to the module. The differences among the implementations stem from variations in coding style, print formatting, and user prompts, demonstrating the flexibility of programming in achieving the same functional goals through diverse approaches.



```
MINGW64:/c/Users/Hp/Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1
Hp@LAPTOP-ALTQPVEB MINGW64 ~
$ cd Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1
Hp@LAPTOP-ALTQPVEB MINGW64 ~/Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1 (main)
$ python PostLabSolution1.py

How many values do you want to enter? 5
Enter a value for the list: 1
Enter a value for the list: 6
Enter a value for the list: 3
Enter a value for the list: 4
Enter a value for the list: 3

List: [1, 6, 3, 4, 3]
Mode: 3
Median: 3
Mean: 3.4

Enter the size of the list: 5
Enter value 1: 1
Enter value 2: 6
Enter value 3: 3
Enter value 4: 4
Enter value 5: 3

Mean: 3.4
Median: 3.0
Mode: 3.0

How many values do you want to enter? 5
Enter the 1st value: 1
Enter the 2nd value: 6
Enter the 3rd value: 3
Enter the 4th value: 4
Enter the 5th value: 3

=====
Mean: 3.4
Median: 3.0
Mode: 3.0

Hp@LAPTOP-ALTQPVEB MINGW64 ~/Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1 (main)
$
```

Figure 2.3 Post Lab Solution 1 Python Script and Sample Text File executed in GitBash

Lastly, *Figure 2.3* further illustrates the variations in program execution, showcasing another layout of the stats.py script, this time executed in GitBash. While the core computations of mean, median, and mode remain consistent, the structure of user prompts, input handling, and output formatting differ from the previous implementations. Notably, this version displays additional list processing steps before presenting the final statistical measures.

Programming Problem #2

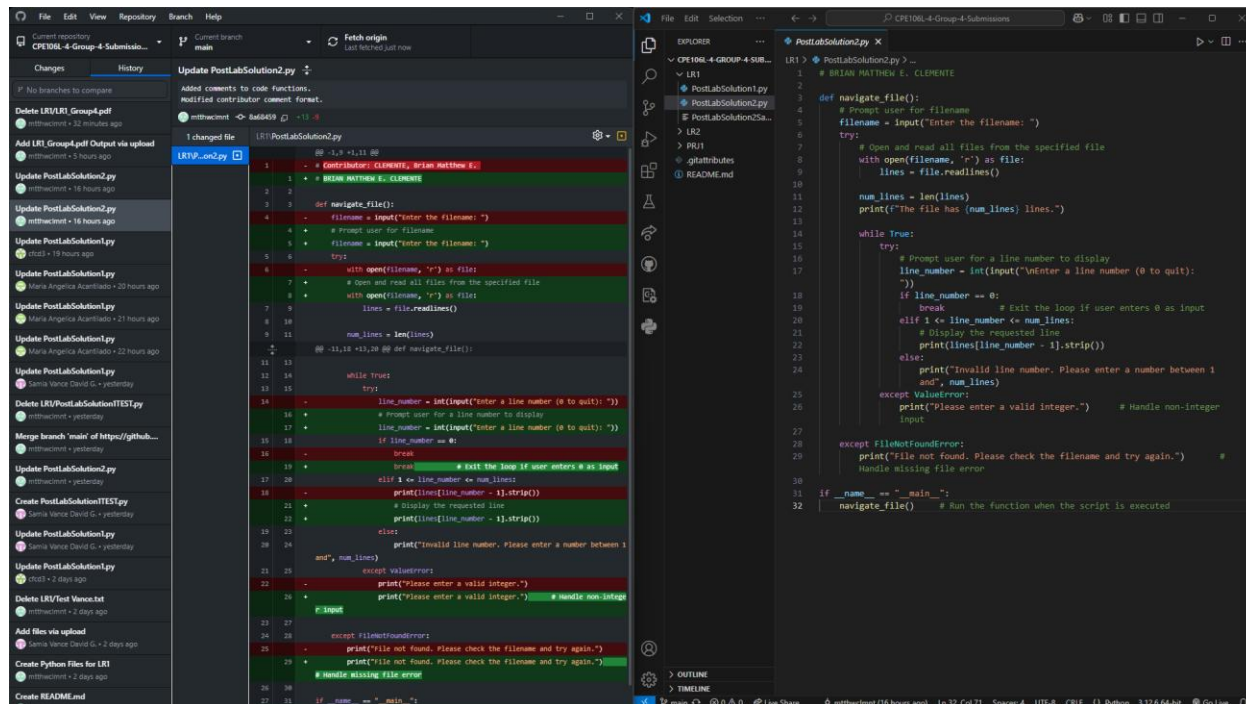
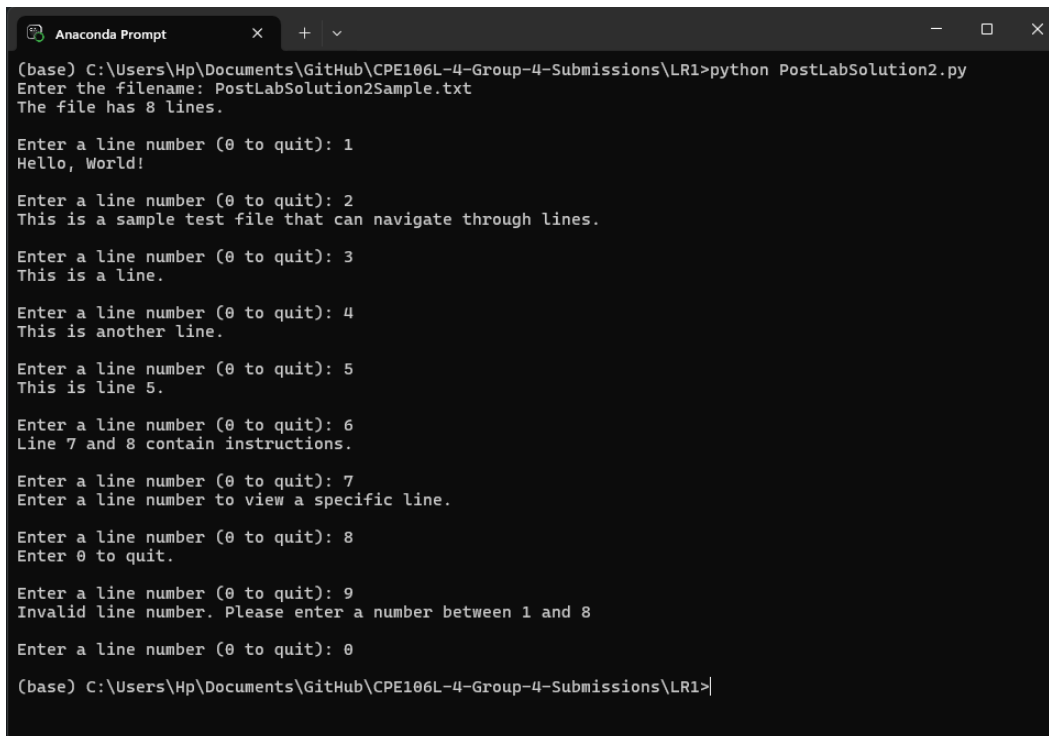


Figure 3.1 Screenshot of GitHub Desktop and VS Code open during Post Lab Solution 2 Commit and Push to Origin

Figure 3.1 displays a screenshot capturing the use of GitHub Desktop and Visual Studio Code during the commit and push process for Post Lab Solution 2. The contributor of this script has been labelled with comments, as practiced in Post Lab Solution 1. In the image, GitHub Desktop shows the tracking of changes and the execution of a commit and push to the remote repository (origin). Meanwhile, VS Code is open, displaying the modified code involved in the update. This figure illustrates the workflow of integrating version control with GitHub while actively coding in VS Code.



```

Anaconda Prompt
(base) C:\Users\Hp\Documents\GitHub\CPE106L-4-Group-4-Submissions\LR1>python PostLabSolution2.py
Enter the filename: PostLabSolution2Sample.txt
The file has 8 lines.

Enter a line number (0 to quit): 1
Hello, World!

Enter a line number (0 to quit): 2
This is a sample test file that can navigate through lines.

Enter a line number (0 to quit): 3
This is a line.

Enter a line number (0 to quit): 4
This is another line.

Enter a line number (0 to quit): 5
This is line 5.

Enter a line number (0 to quit): 6
Line 7 and 8 contain instructions.

Enter a line number (0 to quit): 7
Enter a line number to view a specific line.

Enter a line number (0 to quit): 8
Enter 0 to quit.

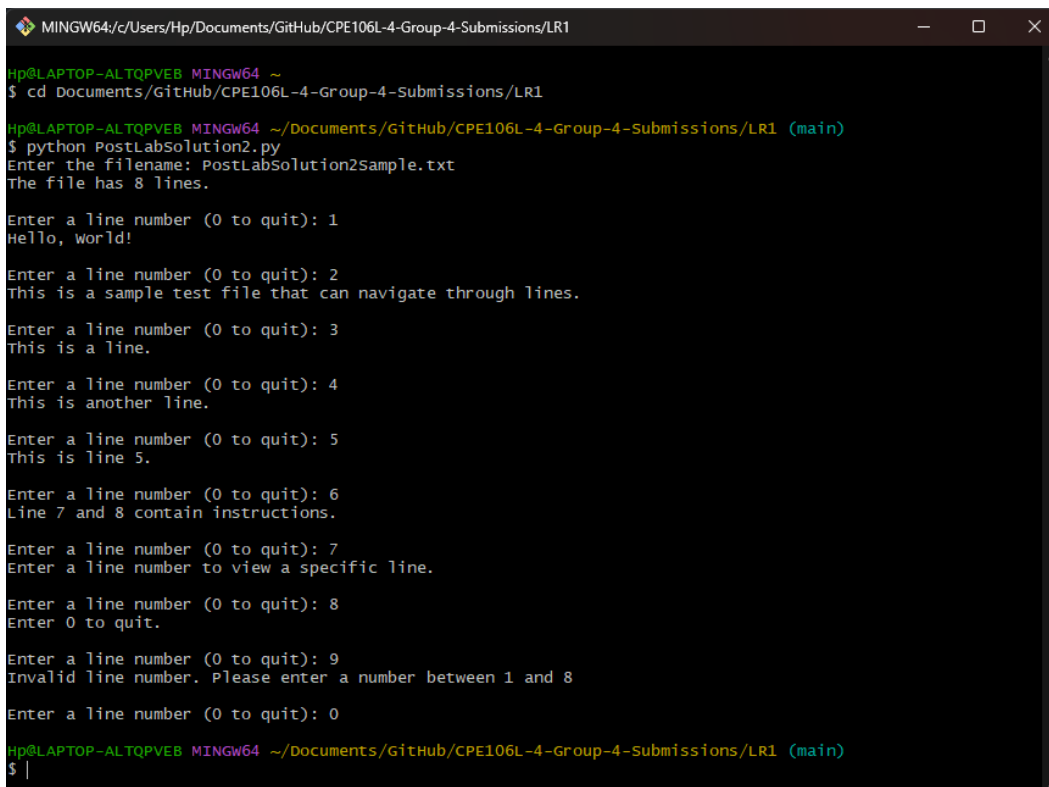
Enter a line number (0 to quit): 9
Invalid line number. Please enter a number between 1 and 8

Enter a line number (0 to quit): 0

(base) C:\Users\Hp\Documents\GitHub\CPE106L-4-Group-4-Submissions\LR1>

```

Figure 3.2 Post Lab Solution 2 Python Script and Sample Text File executed in Anaconda Prompt



```

MINGW64/c/Users/Hp/Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1
Hp@LAPTOP-ALTQPVEB MINGW64 ~
$ cd Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1
Hp@LAPTOP-ALTQPVEB MINGW64 ~/documents/GitHub/CPE106L-4-Group-4-Submissions/LR1 (main)
$ python PostLabSolution2.py
Enter the filename: PostLabSolution2Sample.txt
The file has 8 lines.

Enter a line number (0 to quit): 1
Hello, world!

Enter a line number (0 to quit): 2
This is a sample test file that can navigate through lines.

Enter a line number (0 to quit): 3
This is a line.

Enter a line number (0 to quit): 4
This is another line.

Enter a line number (0 to quit): 5
This is line 5.

Enter a line number (0 to quit): 6
Line 7 and 8 contain instructions.

Enter a line number (0 to quit): 7
Enter a line number to view a specific line.

Enter a line number (0 to quit): 8
Enter 0 to quit.

Enter a line number (0 to quit): 9
Invalid line number. Please enter a number between 1 and 8

Enter a line number (0 to quit): 0

Hp@LAPTOP-ALTQPVEB MINGW64 ~/Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1 (main)
$ |

```

Figure 3.3 Post Lab Solution 2 Python Script and Sample Text File executed in GitBash

Figures 3.2 and 3.3 present the execution of the Post Lab Solution 2 Python script along with a sample text file using both the Anaconda Prompt and GitBash, respectively. The figures showcase how the script interacts with the text file, demonstrating its functionality within the Anaconda and GitBash environments, highlighting the script's compatibility with cross-platform execution and command-line interactions.