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 explored Git, Anaconda, and Visual Studio Code, gaining hands-on knowledge of their functionalities in programming and data science. Particularly, in GitHub, we practiced creating branches by navigating through the Code tab, selecting the current branch, and naming a new one before confirming its creation. After switching to the new branch, we committed files by specifying filenames, adding content, and saving changes, reinforcing our understanding of version control. Additionally, we familiarized ourselves with key GitHub concepts such as repository management, collaboration, and tracking changes, which are essential for efficient teamwork in software development.

With Anaconda, we focused on using it as an environment to run Python scripts. We learned how to launch Python through Anaconda's interface and execute scripts, which provided a structured way to manage and run our code. Although we have not yet explored package installations, we gained an understanding of how Anaconda simplifies Python development by providing a dedicated workspace.

We also worked with Visual Studio Code (VS Code), a versatile code editor that supports multiple programming languages and extensions. We set up VS Code for Python development, configured essential extensions, and navigated features like the integrated terminal and debugging tools. Through the utilization of VS Code alongside Git, specifically GitHub Desktop, we practiced pushing and pulling changes from GitHub, reinforcing our understanding of source control and collaborative development.

By exploring Git, Anaconda, and Visual Studio Code, we gained valuable hands-on experience in version control, coding environments, and collaborative development. Learning how to manage repositories, execute Python scripts, and integrate tools like GitHub and VS Code reinforced our ability to work efficiently in professional programming settings. This process not only enhanced our technical skills but also deepened our understanding of best practices in software development and teamwork.

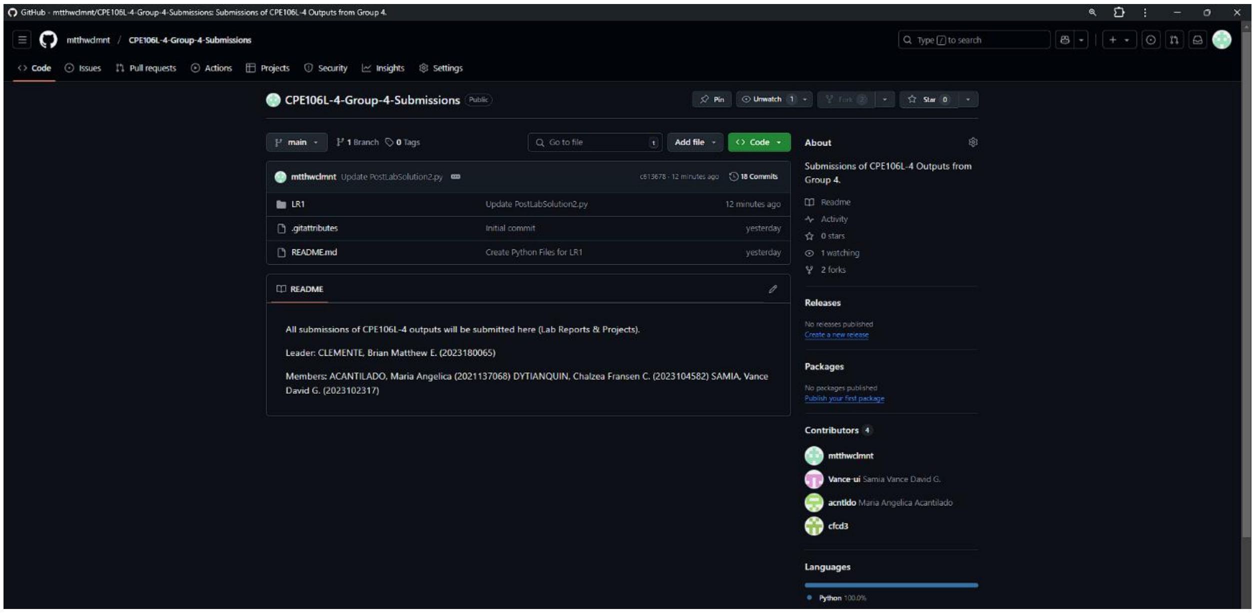
Summary

Python's integrated data structures, including dictionaries, tuples, and lists, were the subject of another important study. These structures are useful tools for developing algorithms and solving problems since they are crucial to the effective storage and processing of data. We also looked at control structures, such as statements with conditions and loops, which are essential for automating repeated operations and enabling a program to make logical judgments. We were better able to comprehend how these structures add to Python's adaptability in software development by dealing with demonstrations and exercises.

Python's use in contemporary computer applications, including developing websites, data science, and automation, was also discussed in our study. This chapter laid a solid basis for future research into more complex subjects by reiterating the core ideas of Python programming. Knowing these fundamentals has given us the ability to use Python in practical problem-solving situations, proving its worth as a language for programming that is both user-friendly and strong.

PostLab

Programming Problems



CPE106L-4-Group-4-Submissions

mtthwclmnt Update PostLabSolution2.py · c613678 · 12 minutes ago · 18 Commits

LR1	Update PostLabSolution2.py	12 minutes ago
.gitattributes	Initial commit	yesterday
README.md	Create Python Files for LR1	yesterday

All submissions of CPE106L-4 outputs will be submitted here (Lab Reports & Projects).

Leader: CLEMENTE, Brian Matthew E. (2023180065)

Members: ACANTILADO, Maria Angelica (2021137068) DYTIANQUIN, Chalzea Fransen C. (2023104582) SAMIA, Vance David G. (2023102317)

About

Submissions of CPE106L-4 Outputs from Group 4.

Readme
Activity
0 stars
1 watching
2 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

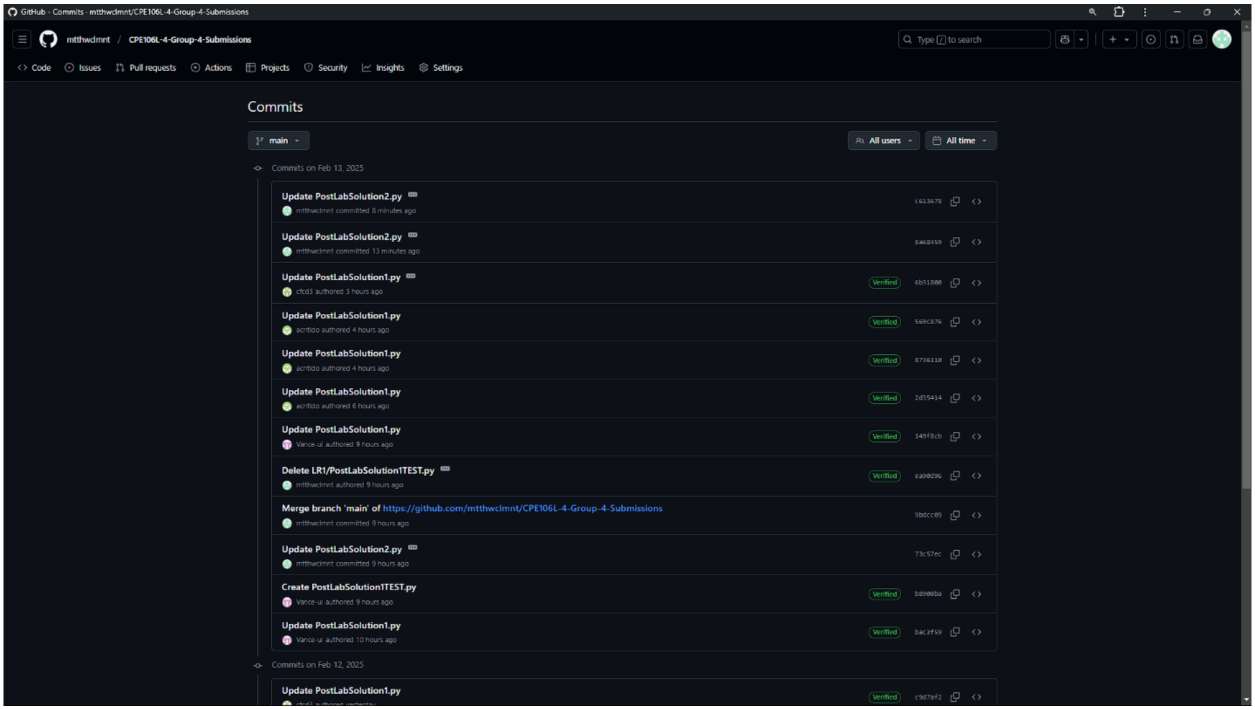
Contributors 4

mtthwclmnt
Vance-ui Samia Vance David G.
acntldo Maria Angelica Acantilado
cfcd3

Languages

Python 100.0%

Figure 1.a Group 4 GitHub Repository Preview


GitHub · Commits · mtthwclmnt/CPE106L-4-Group-4-Submissions
mtthwclmnt / CPE106L-4-Group-4-Submissions
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
main
All users
All time

Commits

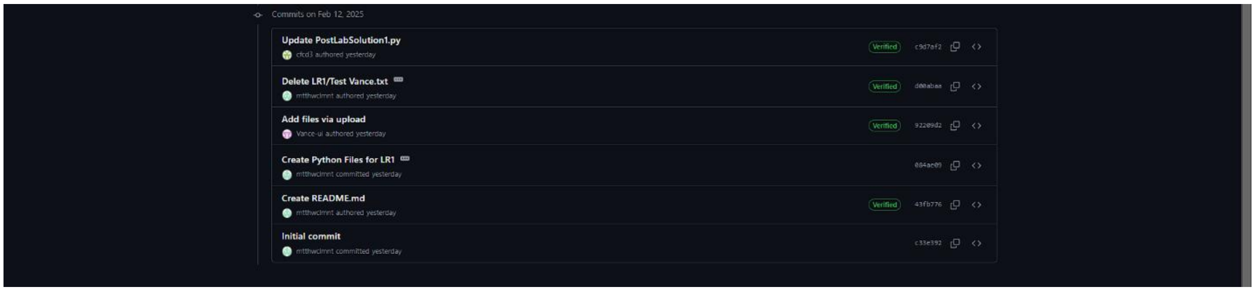
Commits on Feb 13, 2025

Update PostLabSolution2.py	mtthwclmnt committed 8 minutes ago		c613678
Update PostLabSolution2.py	mtthwclmnt committed 13 minutes ago		8a68459
Update PostLabSolution1.py	cfcd3 authored 3 hours ago	Verified	6b31800
Update PostLabSolution1.py	acntldo authored 4 hours ago	Verified	569c876
Update PostLabSolution1.py	acntldo authored 4 hours ago	Verified	8736110
Update PostLabSolution1.py	acntldo authored 6 hours ago	Verified	2d35414
Update PostLabSolution1.py	Vance-ui authored 9 hours ago	Verified	149f8cb
Delete LR1/PostLabSolution1TEST.py	mtthwclmnt authored 9 hours ago	Verified	ea90d96
Merge branch 'main' of https://github.com/mtthwclmnt/CPE106L-4-Group-4-Submissions	mtthwclmnt committed 9 hours ago		9bdcc09
Update PostLabSolution2.py	mtthwclmnt committed 9 hours ago		73c57ec
Create PostLabSolution1TEST.py	Vance-ui authored 9 hours ago	Verified	b8900ba
Update PostLabSolution1.py	Vance-ui authored 10 hours ago	Verified	bac3f59

Commits on Feb 12, 2025

Update PostLabSolution1.py	cfcd3 authored yesterday	Verified	c9d7af2

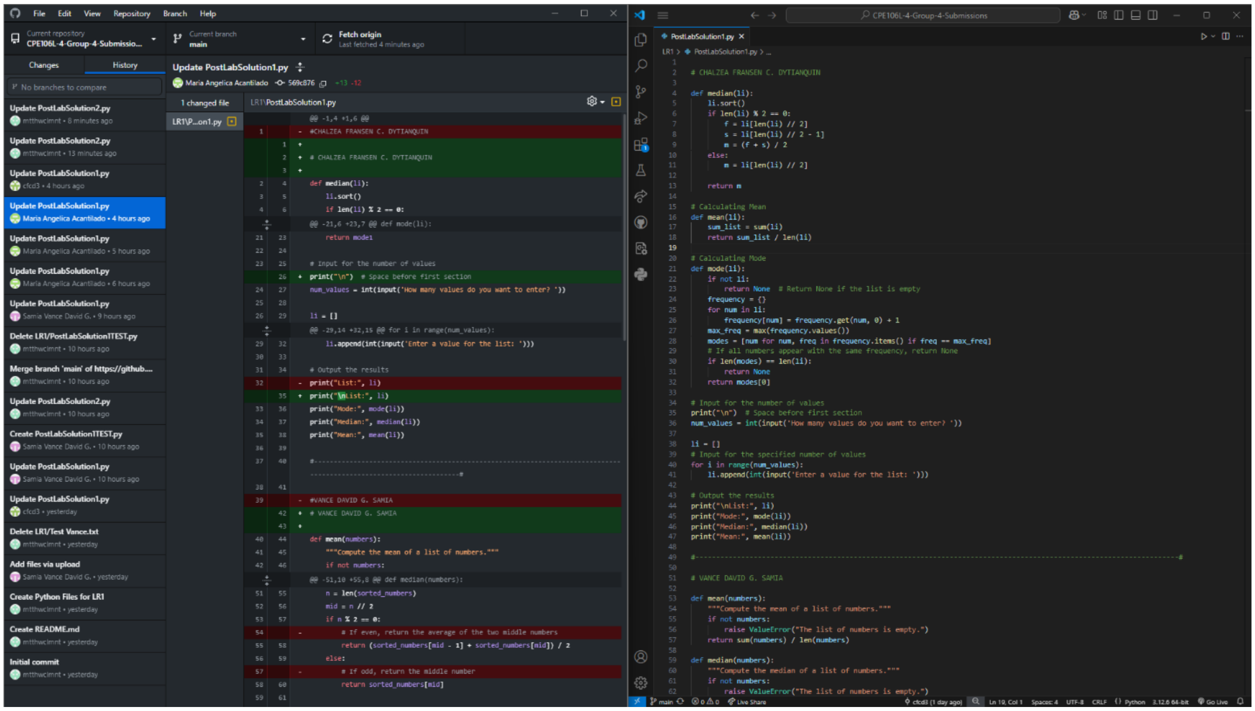
Figure 1.b Lab Report 1 GitHub Repository Commits History Log (Screenshot 1)



```
Commits on Feb 12, 2025

Update PostLabSolution1.py                                   Verified  c9d7af2
cfcd3 authored yesterday

Delete LR1/Test Vance.txt                                    Verified  d00abae
mtthwclmnt authored yesterday

Add files via upload                                         Verified  92209d2
Vance-ui authored yesterday

Create Python Files for LR1                                            084ae09
mtthwclmnt committed yesterday

Create README.md                                             Verified  43fb776
mtthwclmnt authored yesterday

Initial commit                                                         c33e392
mtthwclmnt committed yesterday
```

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

Programming Problem #1



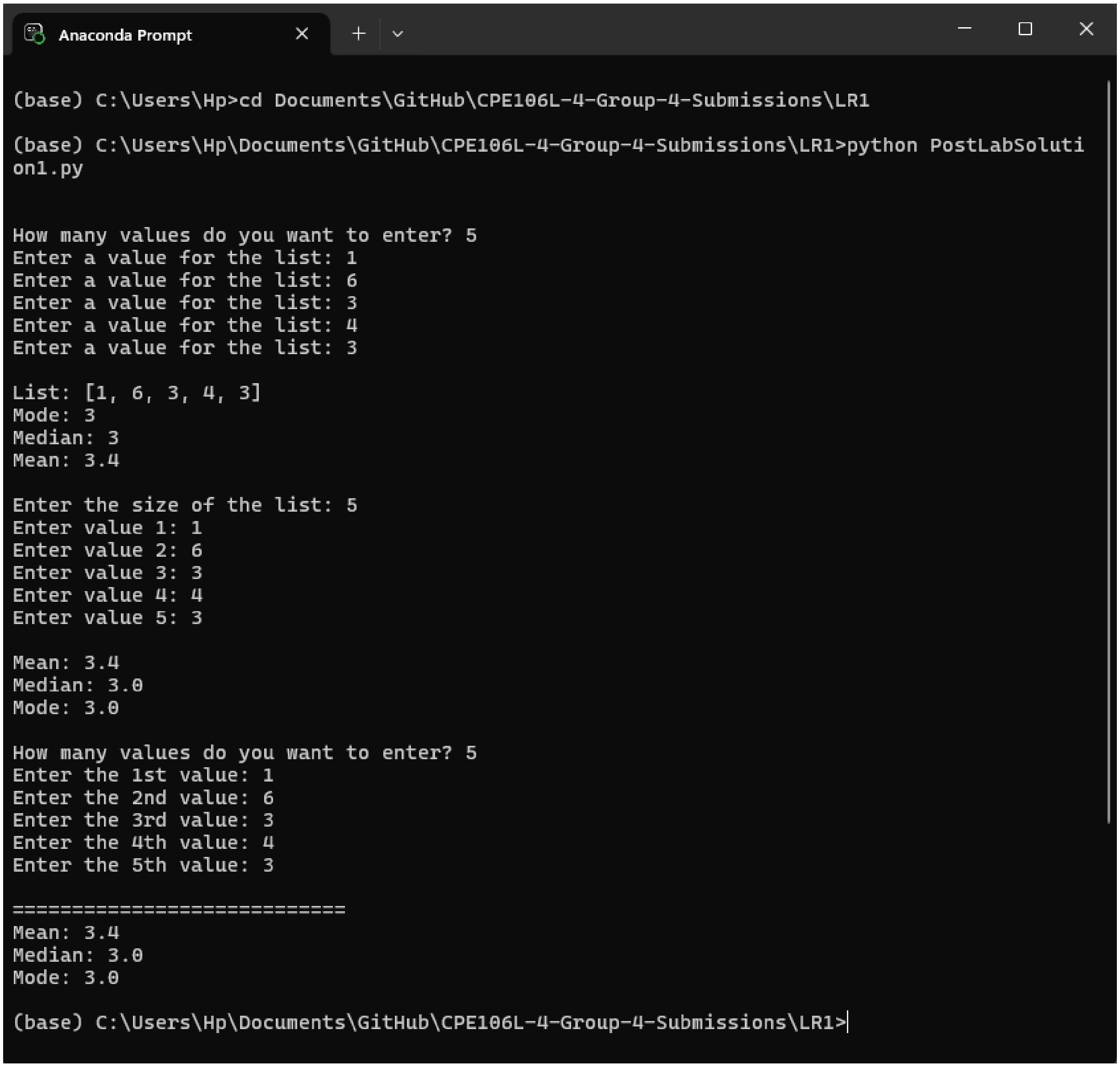
```
Update PostLabSolution2.py
mtthwclmnt • 8 minutes ago
Update PostLabSolution2.py
mtthwclmnt • 13 minutes ago
Update PostLabSolution1.py
cfcd3 • 4 hours ago
Update PostLabSolution1.py
Maria Angelica Acantilado • 4 hours ago
Update PostLabSolution1.py
Maria Angelica Acantilado • 5 hours ago
Update PostLabSolution1.py
Maria Angelica Acantilado • 6 hours ago
Update PostLabSolution1.py
Samia Vance David G. • 9 hours ago
Delete LR1/PostLabSolution1TEST.py
mtthwclmnt • 10 hours ago
Merge branch 'main' of https://github....
mtthwclmnt • 10 hours ago
Update PostLabSolution2.py
mtthwclmnt • 10 hours ago
Create PostLabSolution1TEST.py
Samia Vance David G. • 10 hours ago
Update PostLabSolution1.py
Samia Vance David G. • 10 hours ago
Update PostLabSolution1.py
cfcd3 • yesterday
Delete LR1/Test Vance.txt
mtthwclmnt • yesterday
Add files via upload
Samia Vance David G. • yesterday
Create Python Files for LR1
mtthwclmnt • yesterday
Create README.md
mtthwclmnt • yesterday
Initial commit
mtthwclmnt • yesterday
```

```
Update PostLabSolution1.py
Maria Angelica Acantilado  569c876  +13 -12
1 changed file
LR1\P...on1.py
LR1\PostLabSolution1.py

@@ -1,4 +1,6 @@
- #CHALZEA FRANSEN C. DYTIANQUIN
+
+ # CHALZEA FRANSEN C. DYTIANQUIN
+
  def median(li):
      li.sort()
      if len(li) % 2 == 0:
@@ -21,6 +23,7 @@ def mode(li):
      return mode1

  # Input for the number of values
+ print("\n")  # Space before first section
  num_values = int(input('How many values do you want to enter? '))

  li = []
@@ -29,14 +32,15 @@ for i in range(num_values):
      li.append(int(input('Enter a value for the list: ')))

  # Output the results
- print("List:", li)
+ print("\nList:", li)
  print("Mode:", mode(li))
  print("Median:", median(li))
  print("Mean:", mean(li))

  #----------------------------------------------------------------------------------------------------#

- #VANCE DAVID G. SAMIA
+ # VANCE DAVID G. SAMIA
+
  def mean(numbers):
      """Compute the mean of a list of numbers."""
      if not numbers:
@@ -51,10 +55,8 @@ def median(numbers):
      n = len(sorted_numbers)
      mid = n // 2
      if n % 2 == 0:
-         # If even, return the average of the two middle numbers
          return (sorted_numbers[mid - 1] + sorted_numbers[mid]) / 2
      else:
-         # If odd, return the middle number
          return sorted_numbers[mid]

```

CPE106L-4-Group-4-Submissions
PostLabSolution1.py
LR1 > PostLabSolution1.py > ...

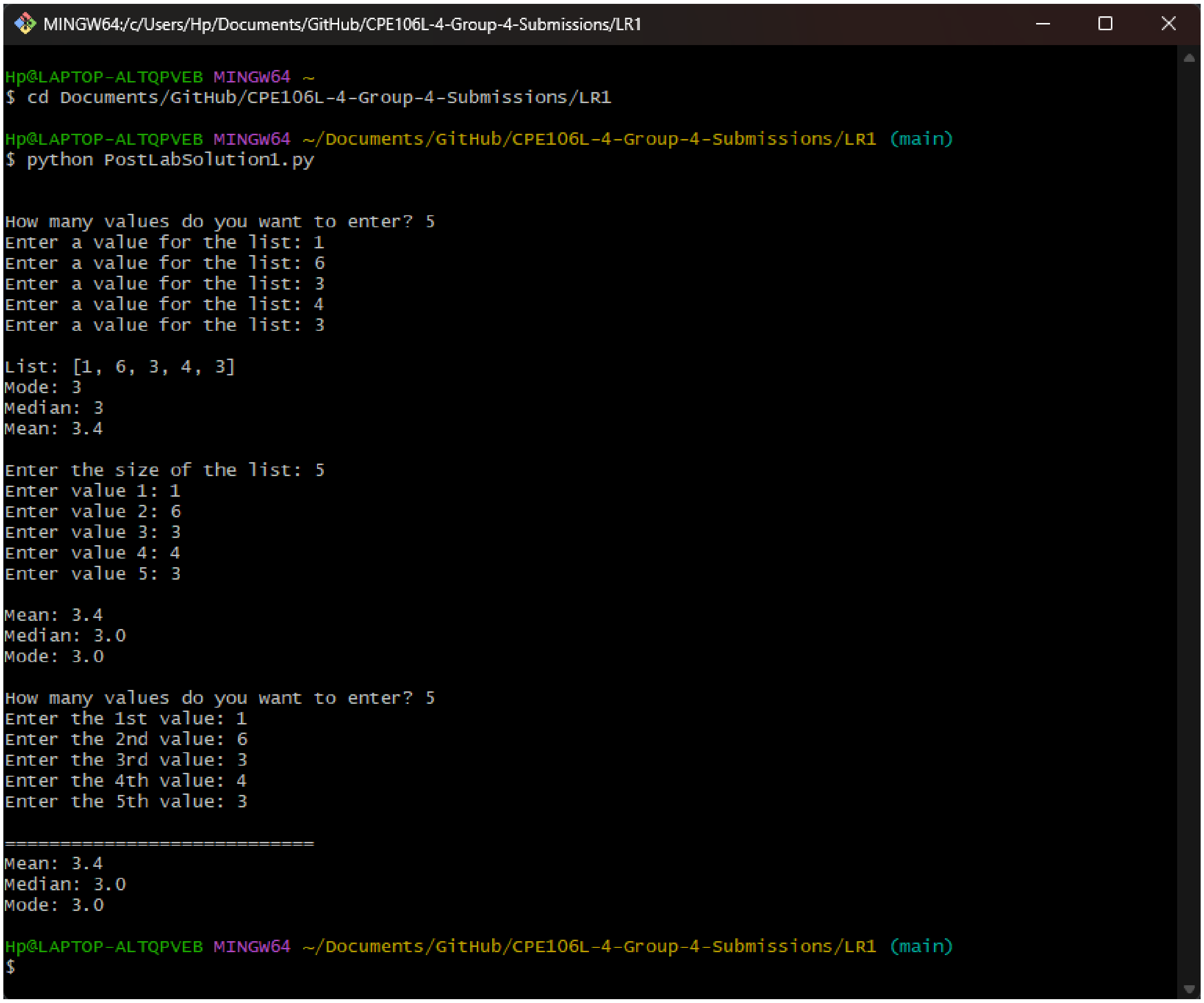
```python

# CHALZEA FRANSEN C. DYTIANQUIN

def median(li):
    li.sort()
    if len(li) % 2 == 0:
        f = li[len(li) // 2]
        s = li[len(li) // 2 - 1]
        m = (f + s) / 2
    else:
        m = li[len(li) // 2]

    return m

# Calculating Mean
def mean(li):
    sum_list = sum(li)
    return sum_list / len(li)

# Calculating Mode
def mode(li):
    if not li:
        return None  # Return None if the list is empty
    frequency = {}
    for num in li:
        frequency[num] = frequency.get(num, 0) + 1
    max_freq = max(frequency.values())
    modes = [num for num, freq in frequency.items() if freq == max_freq]
    # If all numbers appear with the same frequency, return None
    if len(modes) == len(li):
        return None
    return modes[0]

# Input for the number of values
print("\n")  # Space before first section
num_values = int(input('How many values do you want to enter? '))

li = []
# Input for the specified number of values
for i in range(num_values):
    li.append(int(input('Enter a value for the list: ')))

# Output the results
print("\nList:", li)
print("Mode:", mode(li))
print("Median:", median(li))
print("Mean:", mean(li))

#----------------------------------------------------------------------------------------------------#

# VANCE DAVID G. SAMIA

def mean(numbers):
    """Compute the mean of a list of numbers."""
    if not numbers:
        raise ValueError("The list of numbers is empty.")
    return sum(numbers) / len(numbers)

def median(numbers):
    """Compute the median of a list of numbers."""
    if not numbers:
        raise ValueError("The list of numbers is empty.")
```

Figure 2.a Screenshot of GitHub Desktop and VS Code open during Post Lab Solution 1 Commit and Push to Origin



```
(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 PostLabSoluti
on1.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

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

Figure 2.b Post Lab Solution 1 Python Script and Sample Text File executed in Anaconda Prompt

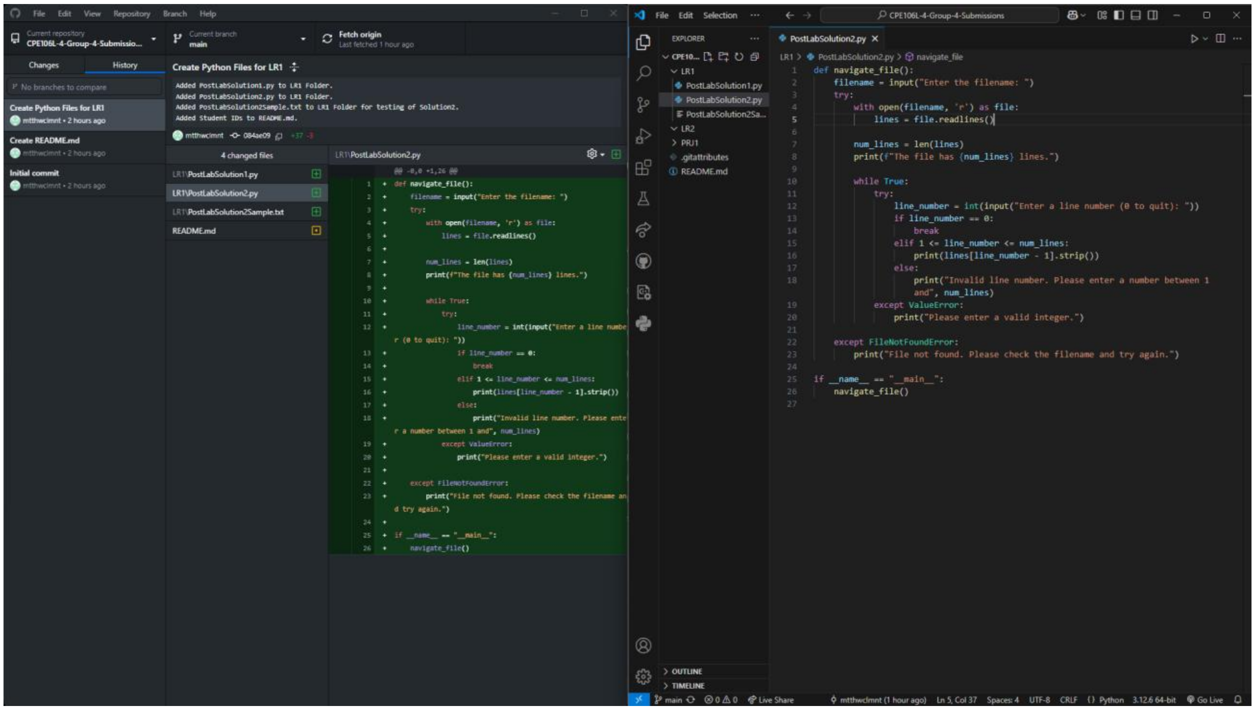


MINGW64:/c/Users/Hp/Documents/GitHub/CPE106L-4-Group-4-Submissions/LR1

```
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 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.c Post Lab Solution 1 Python Script and Sample Text File executed in GitBash

Programming Problem #2



Create Python Files for LR1

```
Added PostLabSolution1.py to LR1 Folder.
Added PostLabSolution2.py to LR1 Folder.
Added PostLabSolution2Sample.txt to LR1 Folder for testing of Solution2.
Added Student IDs to README.md.
```

mtthwclmnt 084ae09 +37 -3

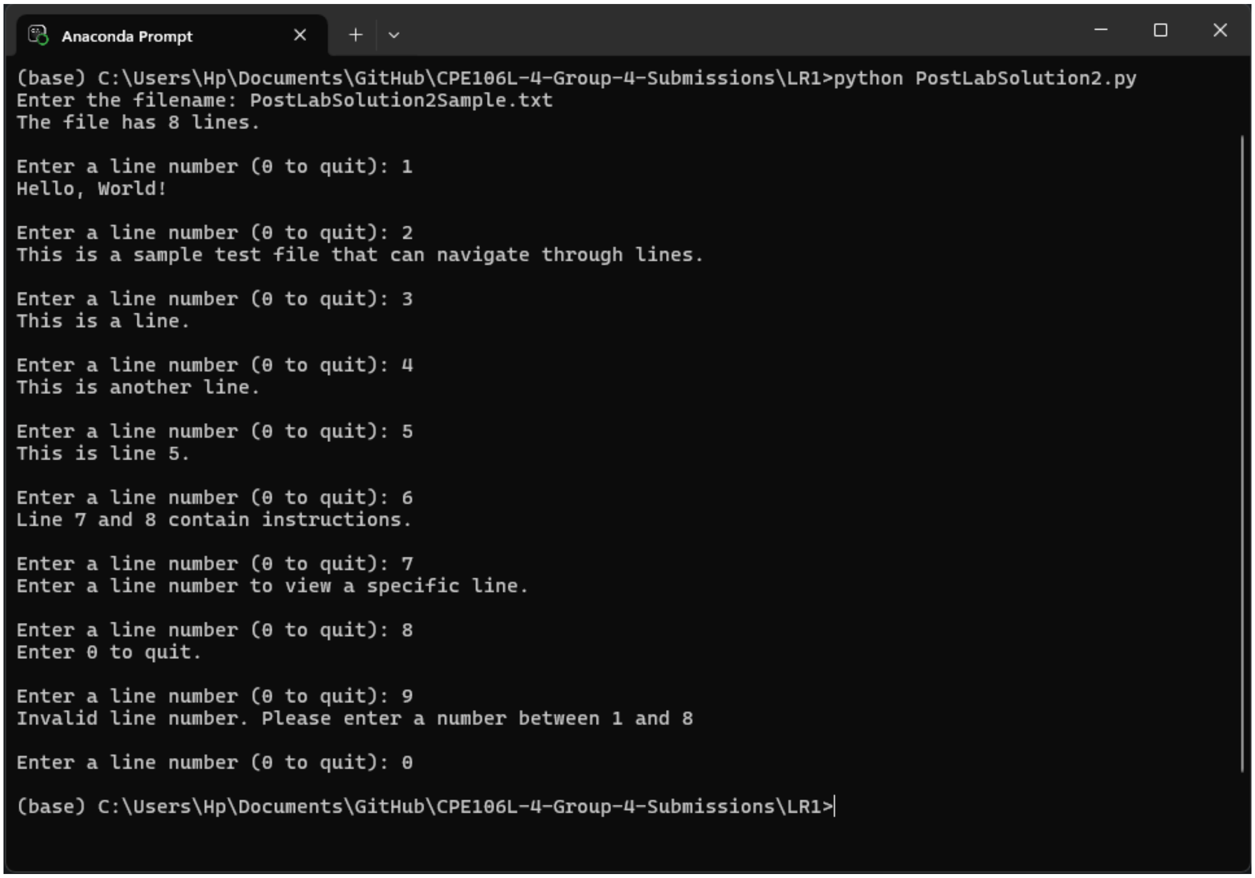
4 changed files

LR1\PostLabSolution1.py
LR1\PostLabSolution2.py
LR1\PostLabSolution2Sample.txt
README.md

LR1 > PostLabSolution2.py > navigate_file

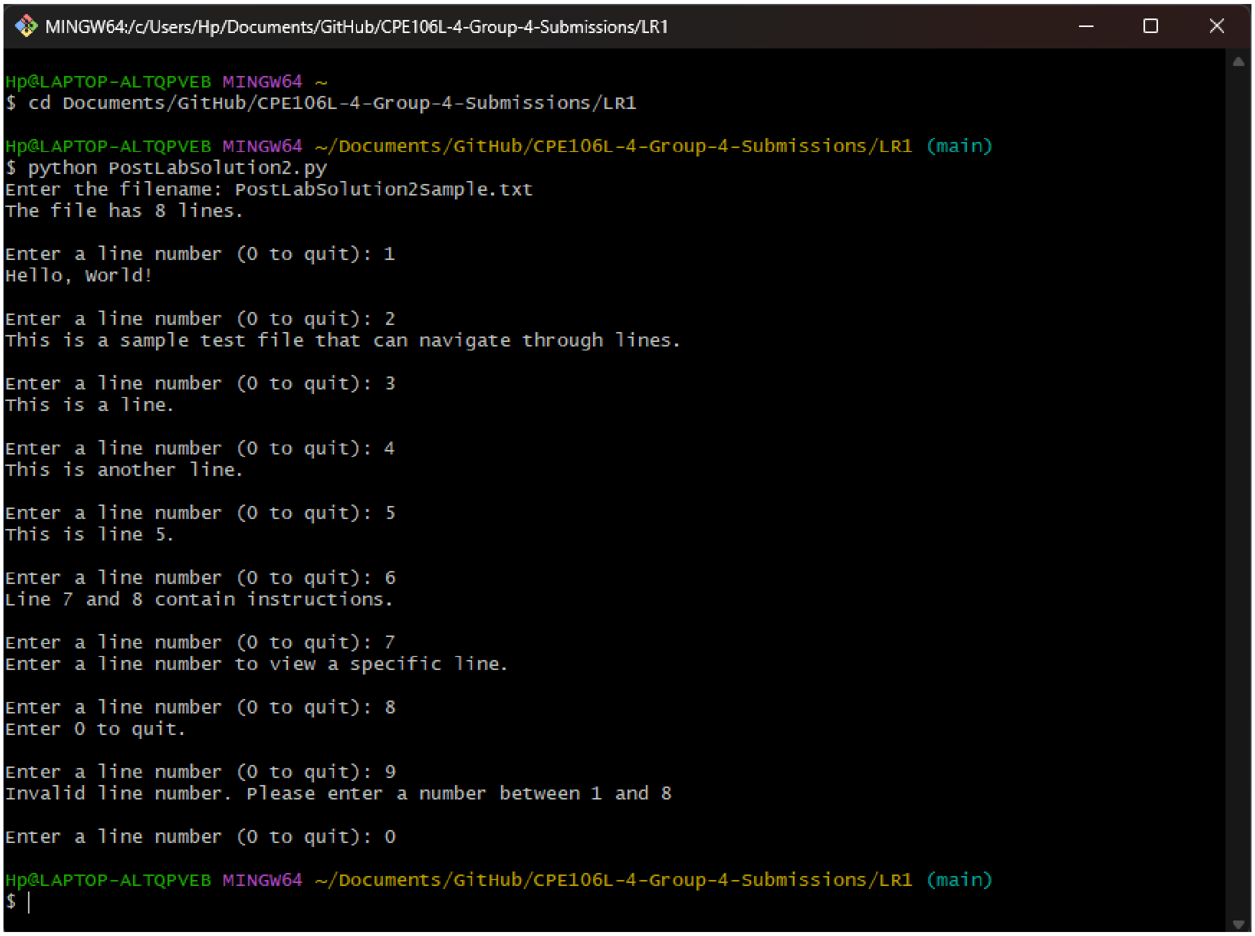
```python
def navigate_file():
    filename = input("Enter the filename: ")
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()

        num_lines = len(lines)
        print(f"The file has {num_lines} lines.")

        while True:
            try:
                line_number = int(input("Enter a line number (0 to quit): "))
                if line_number == 0:
                    break
                elif 1 <= line_number <= num_lines:
                    print(lines[line_number - 1].strip())
                else:
                    print("Invalid line number. Please enter a number between 1 and", num_lines)
            except ValueError:
                print("Please enter a valid integer.")

    except FileNotFoundError:
        print("File not found. Please check the filename and try again.")

if __name__ == "__main__":
    navigate_file()
```

Figure 3.a Screenshot of GitHub Desktop and VS Code open during Post Lab Solution 2 Commit and Push to Origin



```
(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.b 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.c Post Lab Solution 2 Python Script and Sample Text File executed in GitBash