



MAPÚA UNIVERSITY

SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 1: Using Software Tools and Code Versioning System

CPE106L (Software Design Laboratory)

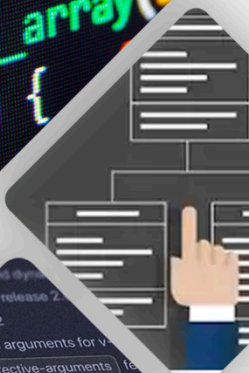
Danikka Villaron

Jessa Abigaile Tongol

Mark Vincent De Villa

Matthew Benedict Nepomuceno

Group No.: 2
Section: **E03**



PreLab

Readings, Insights, and Reflection

Professional Git
9781119285007

De Villa (Chapter 1 - What is Git?)

Brent Laster's Professional Git first chapter - Understanding Git's Concepts - provided information regarding the history of Git, its advantages and disadvantages, and the expected terms and concepts encountered while using Git.

De Villa (Chapter 2 - Key Concepts)

The second chapter - Using Git - enlightened us on the design and function of Git as a source management system and covered the model utilized by Git to create an environment perfect for the implementation and testing phase of various projects.

Fundamentals of Python: First Programs

Danikka Villaron (Chapter 1 - Introduction & Chapter 2 - Software Development, Data Types, and Expressions)

In Chapter 1, I learned the basics of Python programming. It started with explaining what programs and modules are. A module is just a file that contains Python code, which makes it easier to organize programs.

One of the examples was a "Guessing a Number" game. It taught me how to use the `random` module and write a program with loops and conditions. It was fun and helped me understand how to create simple interactive programs.

I also learned about adding comments to code. Comments make the program easier to understand, especially when working with others or revisiting code later.

Also, this chapter introduced important parts of Python, like variables, keywords, and how to name things in the program. I liked how Python automatically figures out the type of a variable, which makes it easier to write code.

Lastly, I discovered how to use the `help()` function in Python. It's really handy when I need to know more about any Python function or module.

InLab

- Objectives
 1. Familiarize oneself with GitHub through the course [Introduction to GitHub](#)
 2. Familiarize oneself with Python with the sample codes in the first chapter of Lambert’s Fundamentals Of Python: First Programs.
- Tools Used
 - o Git Terminal
 - o Visual Studio Code
 - o Virtual Box
 - o OnlineGDB
- Procedure

Part 1: Introduction To Github

The group began by setting up their Virtual Machines with Oracle VM VirtualBox Manager to perform the Introduction To GitHub activity. The first page shown by the site was an introduction to the activity, which provided the scope, estimated time to finish, and instructions to begin the course, as seen in Figures 1.1, 1.2, 1.3, and 1.4.

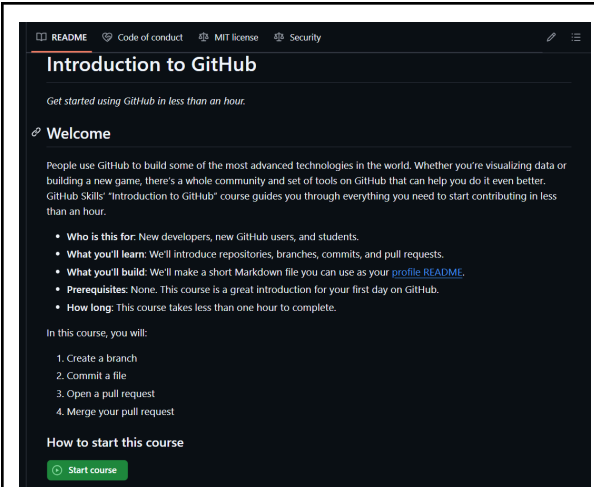


Figure 1.1 Villaron’s first step with GitHub

Figure 1.2 Tongol’s first step with GitHub

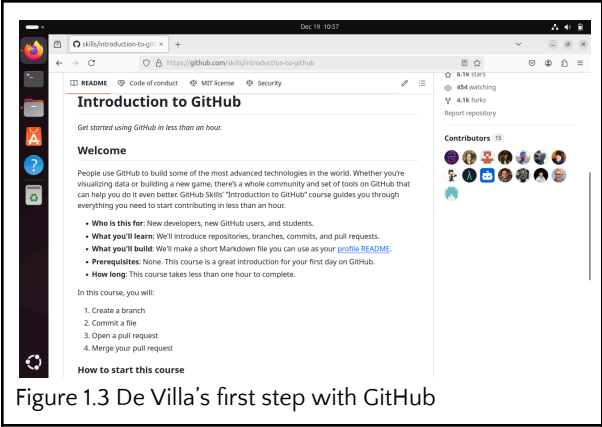


Figure 1.3 De Villa's first step with GitHub

Figure 1.4 Nepomuceno's first step with GitHub

Following the introduction was the first task in the GitHub course – creating a branch – as seen in Figures 2.1, 2.2, 2.3, and 2.4. This course section provided information on what a branch is, its purpose, and steps to create and edit branches.

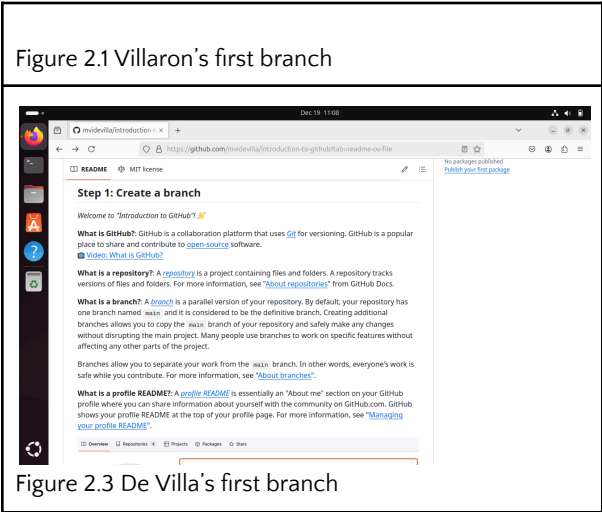


Figure 2.3 De Villa's first branch

Figure 2.2 Tongol's first branch

Figure 2.4 Nepomuceno's first branch

After creating the first branch, the next task provided by the course was producing a commit, which had similarities to creating snapshots according to GitHub; Figures 3.1, 3.2, 3.3, and 3.4 present the group's performance on the given task.

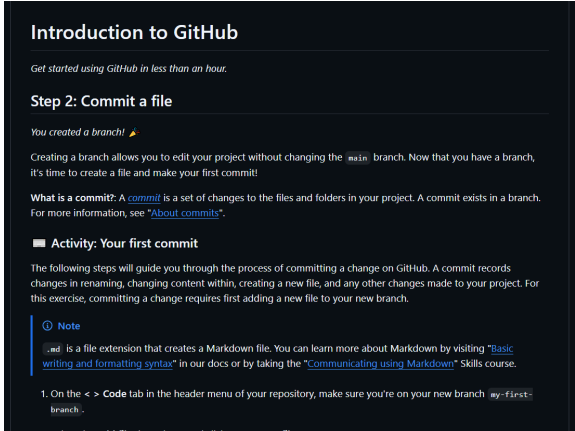


Figure 3.1 Villaron's first commit

Figure 3.2 Tongol's first commit

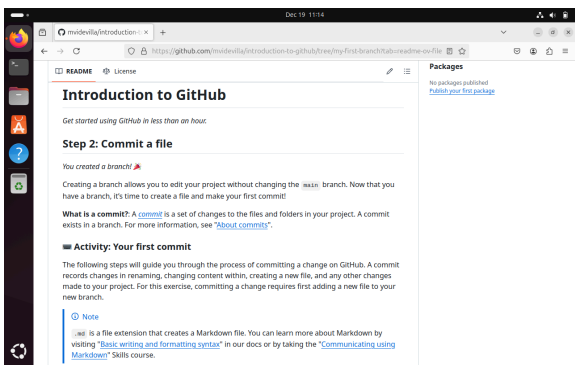


Figure 3.3 De Villa's first commit

Figure 3.4 Nepomuceno's first commit

After the group's commit creation, the following task was to open and create a pull request. A pull request, according to GitHub, enables developers to request to merge their branch with the main project, allowing multiple developers to edit a project simultaneously and select the most suitable change for the main project. Figures 4.1, 4.2, 4.3, and 4.4 present the group members at this stage of the course.

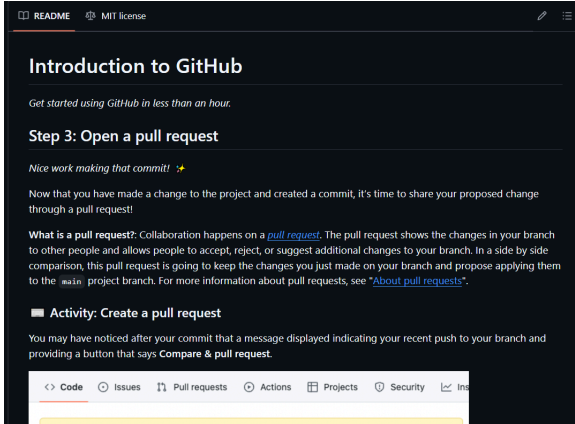


Figure 4.1 Villaron's first pull request

Figure 4.2 Tongol's first pull request

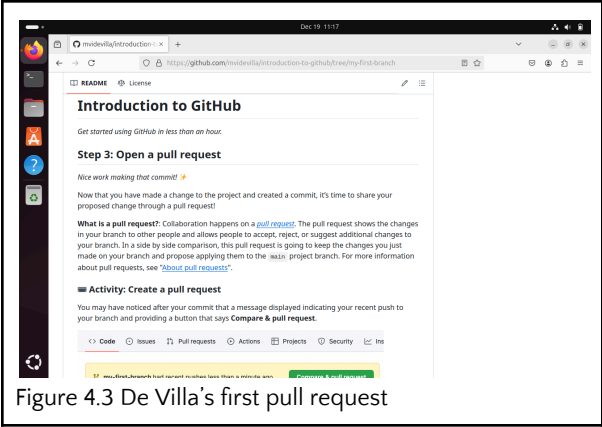


Figure 4.3 De Villa's first pull request

Figure 4.4 Nepomuceno's first pull request

The final task of the introductory course was to manage the pull requests and merge the branch with the main project. The GitHub introductory course explained that merging a main and branch adds the changes made in the branches to the main project. Figures 5.1, 5.2, 5.3, and 5.4 present the group's performance on this task to merge their branch to the main.

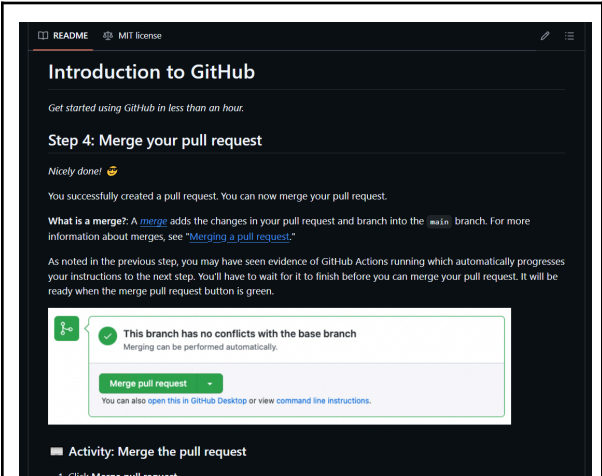


Figure 5.1 Villaron's merge with pull request

Figure 5.2 Tongol's merge with pull request

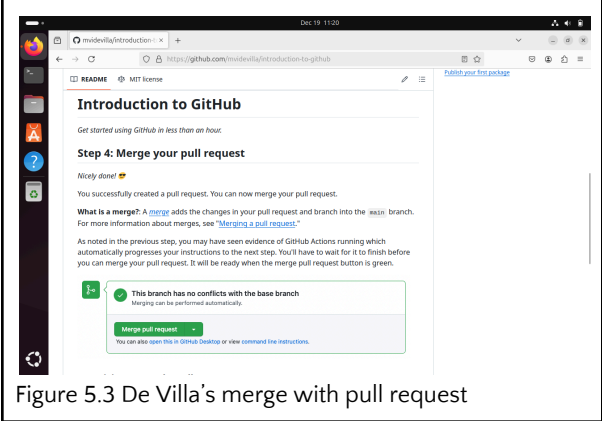
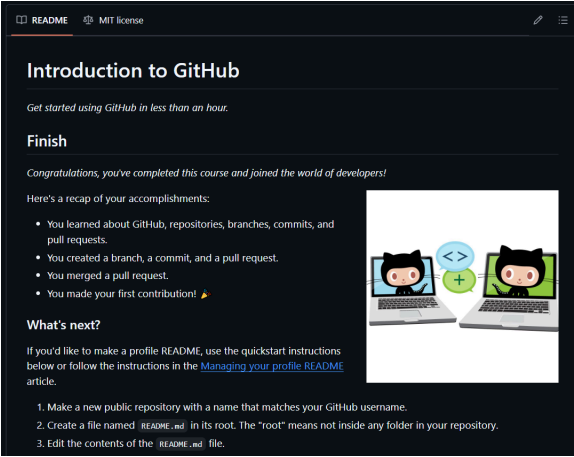
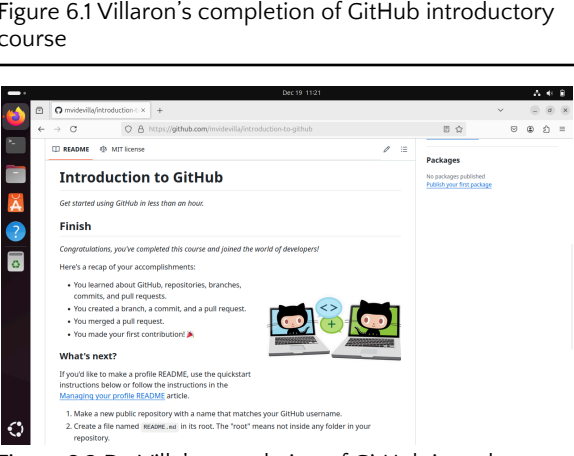
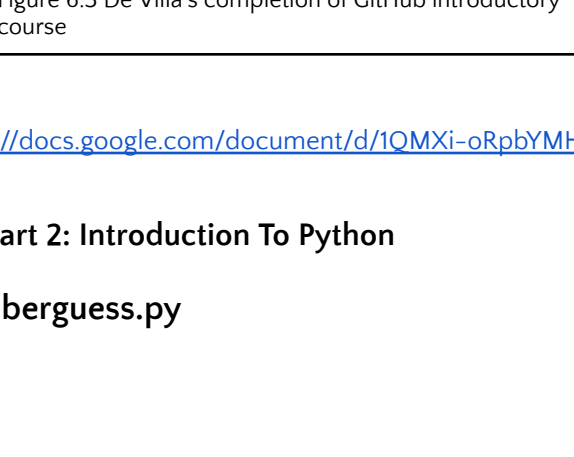



Figure 5.3 De Villa's merge with pull request

Figure 5.4 Nepomuceno's merge with pull request

The final section of the GitHub introductory course presents the group with a summary of the completed tasks and an article that leads the group to instructions for creating a readme file in their project. Figures 6.1, 6.2, 6.3, and 6.4 show proof of accomplishment on the introductory course.

	Figure 6.2 Tongol's completion of GitHub introductory course
	Figure 6.1 Villaron's completion of GitHub introductory course
	Figure 6.3 De Villa's completion of GitHub introductory course
	Figure 6.4 Nepomuceno's completion of GitHub introductory course

https://docs.google.com/document/d/1QMxi-oRpbYMHpMn6Qt4E-sLzOLUhFlItzfu4_rPMV2U/edit?usp=sharing

Part 2: Introduction To Python

numberguess.py

```

numberguess.py > ...
1  #
2  Author: Ken Lambert
3  Plays a game of guess the number with the user.
4  """
5
6  import random
7
8  def main():
9      """Inputs the bounds of the range of numbers,
10     and lets the user guess the computer's number until
11     the guess is correct."""
12     smaller = int(input("Enter the smaller number: "))
13     larger = int(input("Enter the larger number: "))
14     myNumber = random.randint(smaller, larger)
15     count = 0
16     while True:
17         count += 1
18         userNumber = int(input("Enter your guess: "))
19         if userNumber < myNumber:
20             print("Too small")
21         elif userNumber > myNumber:
22             print("Too large")
23         else:
24             print("You've got it in", count, "tries!")
25             break
26
27 if __name__ == "__main__":
28     main()

```

Figure 1.1 Code for numberguess.py

```

nika@nika-VirtualBox:~/Documents
Enter the smaller number: 5
Enter the larger number: 10
Enter your guess: 12
Too large
Enter your guess: 15
Too large
Enter your guess: 15
Too large
Enter your guess: 12
Too large
Enter your guess: 10
You've got it in 5 tries!

```

Figure 1.2 Output of the numberguess.py

Figure 1.1 shows a Python program for a number guessing game. You pick a range of numbers, and the computer randomly chooses one. Then you try to guess the number, and the program lets you know if your guess is too high, too low, or correct. The game keeps going until you guess it right, and it tells you how many tries it took. As for the Figure 1.2, it shows an example of how it works, but the guesses and results will be different each time you input a number, as shown in the sample output.

counter.py


```

counter.py > Counter
~/Documents/Lab 1/counter.py [ect]:
18 def increment(self, amount=1):
19     """Adds amount to the counter."""
20     self._value += amount
21
22 def decrement(self, amount=1):
23     """Subtracts amount from the counter."""
24     self._value -= amount
25
26 # Accessor methods
27 def getValue(self):
28     """Returns the counter's value."""
29     return self._value
30
31 def __str__(self):
32     """Returns the string representation of the counter."""
33     return str(self._value)
34
35 def __eq__(self, other):
36     """Returns True if self equals other, or False otherwise."""
37     if self is other:
38         return True
39     if type(self) != type(other):
40         return False
41     return self._value == other._value
42

```

Figure 2.1 Code for counter.py

Figure 2.2 is a code that defines a Counter class that acts like a simple counter. It has methods to increase, decrease, check the value, and compare with another counter. A Counter object can be used to keep track of a number. It's an easy way to see how classes and methods work.

ourSum.py

```

ourSum.py > ourSum
1 def ourSum(lower, upper, margin = 0):
2     """Returns the sum of the numbers from lower to upper,
3     and outputs a trace of the arguments and return values
4     on each call."""
5     blanks = " " * margin
6     print(blanks, lower, upper)
7     if lower > upper:
8         print(blanks, 0)
9         return 0
10    else:
11        result = lower + ourSum(lower + 1, upper, margin + 4)
12        print(blanks, result)
13    return result

```

Figure 2.2 Code for ourSum.py

The ourSum function adds numbers from lower to upper and shows each step. It works by calling itself (recursion). If lower is greater than upper, it stops and returns 0. Otherwise, it adds lower to the result of calling ourSum with lower + 1. The margin adds spaces to show how deep it is in the process. For example, calling ourSum(1, 3) adds 1 + 2 + 3 and shows the steps as it calculates.

pickling.py

```
pickling.py > ...
1  import pickle
2
3  lyst = [60, "A string object", 1977]
4  fileObj = open("items.dat", "wb")
5  for item in lyst:
6      pickle.dump(item, fileObj)
7  fileObj.close()
8
9  lyst = list()
10 fileObj = open("items.dat", "rb")
11 while True:
12     try:
13         item = pickle.load(fileObj)
14         lyst.append(item)
15     except EOFError:
16         fileObj.close()
17         break
18 print(lyst)
```

Figure 3.1 Code for pickling.py

```
nika@nika-VirtualBox:~/Documents/Lab 1$
[60, 'A string object', 1977]
```

Figure 3.2 Output

Figure 3.1 shows how to use the pickle module to save and load objects to and from a file. It starts with a list of items (lyst) and writes each item into a binary file called items.dat using pickle.dump(). After that, it reads the file back using pickle.load(), appending each item to a new list until the end of the file is reached. If an EOFError occurs (end of file), the loop stops. As for the output, it prints the reconstructed list to show the data has been successfully saved and loaded.

try-except.py

```
1 """
2 Author: Ken Lambert
3 Demonstrates a function that traps number format errors during input.
4 """
5
6 def safeIntegerInput(prompt):
7     """Prompts the user for an integer and returns the
8     integer if it is well-formed. Otherwise, prints an
9     error message and repeats this process."""
10    inputString = input(prompt)
11    try:
12        number = int(inputString)
13        return number
14    except ValueError:
15        print("Error in number format:", inputString)
16        return safeIntegerInput(prompt)
17
18 if __name__ == "__main__":
19     age = safeIntegerInput("Enter your age: ")
20     print("Your age is ", age)
```

Figure 4.1 Code of try-except.py

```
Enter your age: abc
Error in number format: abc
Enter your age: 6i
Error in number format: 6i
Enter your age: 61
Your age is  61

...Program finished with exit code 0
```

Figure 4.2 Output of try-except.py

The try-except.py demonstrates the exception handling of the Python language. Figure 4.1 presents the code where a safeIntegerInput function is defined to determine whether the input is an integer or not, wherein a try block attempts to create a variable called number obtained from the assumed integer inputString variable, and when an error occurs due to inputString not being an integer, an except block runs and displays an error for the user to see as seen in lines 2 and 4 on the output. However, when no errors occur, the code continues and displays "Your age is [integer]" in the console output, as seen in line 6 of Figure 4.2.

PostLab

Programming Problems

1.

- Programming Problem 1:

Statisticians would like to have a set of functions to compute the median and mode of a list of numbers. The median is the number that would appear at the midpoint of a list if it were sorted. The mode is the number that appears most frequently in the list. Define these functions in a module named stats.py. Also include a function named mean, which computes the average of a set of numbers. Each function expects a list of numbers as an argument and returns a single number.

Code: Figure 3.1 stats.py

```
stats.py > ...
1  import statistics
2  numbers_list = [8, 0, 8, 0]
3
4  mean_value = statistics.mean(numbers_list)
5  median_value = statistics.median(numbers_list)
6  mode_value = statistics.mode(numbers_list)
7
8  print("List:", numbers_list)
9  print("Mode:", mode_value)
10 print("Median:", median_value)
11 print("Mean:", mean_value)
```

Output: Figure 3.2 output

```
nika@nika-VirtualBox:
List: [8, 0, 8, 0]
Mode: 8
Median: 4.0
Mean: 4
```

In these figures, I used Python to find the **mean**, **median**, and **mode** of a list of numbers. I gave the list [8, 0, 8, 0] and used the statistics module to do the calculations for me. The **mean** is just the average, so Python adds up the numbers and divides by how many there are. The **median** is the middle value when the numbers are sorted, and the **mode** is the number that shows up the most. Then, I printed all the results so it's easy to see. I used the statistics module to save me writing long formulas.

- Programming Problem 2:

Write a program that allows the user to navigate through the lines of text in a file. The program prompts the user for a filename and inputs the lines of text into a list. The program then enters a loop in which it prints the number of lines in the file and prompts the user for a line number. Actual line numbers range from 1 to the number of lines in the file. If the input is 0, the program quits. Otherwise, the program prints the line associated with that number.

9

Code: Figure 3.2 TEXT.py

```

TEXT.PY > ...
1  file_name = input("Enter file name: ")
2
3  try:
4      with open(file_name, "r") as f:
5          listFile = f.read().splitlines()
6
7      while True:
8          print("\nThe file has", len(listFile), "lines.")
9          selection = int(input("Enter line number (or 0 to quit): "))
10         if selection == 0:
11             print("BaiBai:<")
12             break
13
14         try:
15             selection = int(selection) - 1
16             if 0 <= selection < len(listFile):
17                 print("\n", listFile[selection], "\n")
18             else:
19                 print("Invalid line number. Pweasee try again.")
20         except ValueError:
21             print("Invalid input. Valid # onle.")
22     except FileNotFoundError:
23         print("File not found. Please check the file name and try again.")

```

Output: Figure 3.3 output

```

nika@nika-VirtualBox:~/Documents/Lab 1$
Enter file name: book.txt

The file has 6 lines.
Enter line number (or 0 to quit): 1
My Favorite

The file has 6 lines.
Enter line number (or 0 to quit): 2
book

The file has 6 lines.
Enter line number (or 0 to quit): 3
is

The file has 6 lines.
Enter line number (or 0 to quit): 4
Computer Architecture & Organization

The file has 6 lines.
Enter line number (or 0 to quit): 5
by

The file has 6 lines.
Enter line number (or 0 to quit): 6
John P Hayes

The file has 6 lines.
Enter line number (or 0 to quit): 7
Invalid line number. Pweasee try again.

The file has 6 lines.
Enter line number (or 0 to quit): 0
BaiBai:<

```

The code starts by asking for a file name using `input()` and tries to open the file with `open()`. If the file is found, it reads all the lines using `readlines()` and shows how many lines there are with `len()`. Then, it lets the user enter a line number to see that line, using indexing. If the line number is invalid, it shows an error. The code handles missing files with `try-except` and uses `FileNotFoundError` for errors. To close or quit, I set the quit command to 0, the function `close()` makes sure to close the file when finished. The 'r' is a mode in python that only 'reads' the file. It can be in different modes such as w, a, b, and x. But in this problem, 'r' mode is applied.