PostLab

Programming Problems

1. **stats.py**

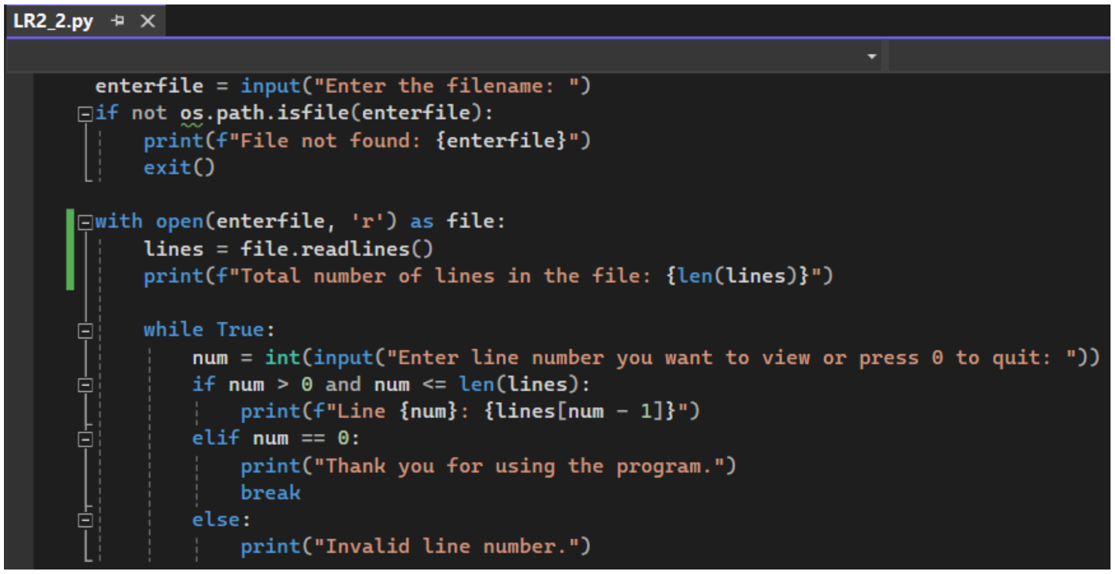
   A group of statisticians at a local college has asked you to create a set of functions that compute the median and mode of a set of numbers, as defined in the below sample programs: ·

   - mode.py
   - median.py

   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 should expect a list of numbers as an argument and return a single number. Each function should return 0 if the list is empty. Include a main function that tests the three statistical functions with a given list.

2. LR2_2.py (Torres)

Write a program that allows the user to navigate the lines of text in a file. The program should prompt the user for a filename and input the lines of text into a list. The program then enters a loop that 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.

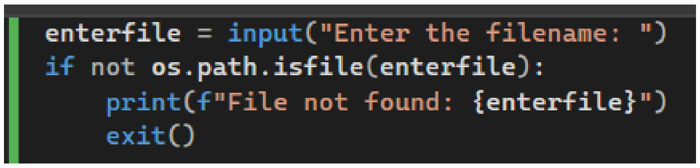


```python
enterfile = input("Enter the filename: ")
if not os.path.isfile(enterfile):
    print(f"File not found: {enterfile}")
    exit()

with open(enterfile, 'r') as file:
    lines = file.readlines()
    print(f"Total number of lines in the file: {len(lines)}")

    while True:
        num = int(input("Enter line number you want to view or press 0 to quit: "))
        if num > 0 and num <= len(lines):
            print(f"Line {num}: {lines[num - 1]}")
        elif num == 0:
            print("Thank you for using the program.")
            break
        else:
            print("Invalid line number.")
```

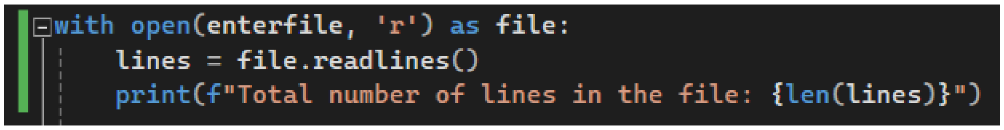
Figure 1. Complete breakdown of code



```python
enterfile = input("Enter the filename: ")
if not os.path.isfile(enterfile):
    print(f"File not found: {enterfile}")
    exit()
```

Figure 2. File Input

The line `enterfile = input("Enter the filename: ")` prompts the user to enter a filename, and `os.path.isfile()` function checks if the file exists at the specified path. If the file doesn't exist, the program prints an error message with the filename and terminates using the `exit()` code. This code block is crucial in ensuring that the program can read from the file entered by the user. By checking if the file exists before continuing, we can ensure that the program will run smoothly.

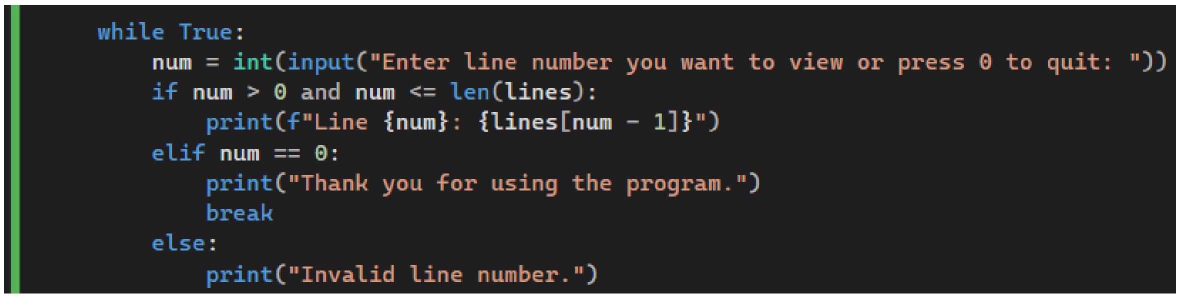


```python
with open(enterfile, 'r') as file:
    lines = file.readlines()
    print(f"Total number of lines in the file: {len(lines)}")
```

Figure 3. Line count

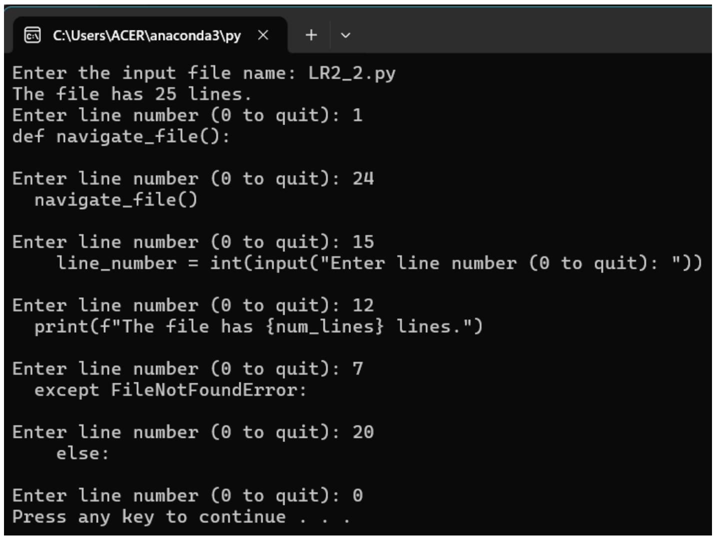
This code reads a file specified by the 'enterfile' variable and calculates the number of lines in the file. It opens the file in read mode, reads all the lines using the 'readlines()' method, calculates the list length using the 'len()' function, and displays the total number of lines. The time and space complexity of the code are both O(n), where n is the number of lines in the file.

Figure 4. Print User chosen Line



```
    while True:
        num = int(input("Enter line number you want to view or press 0 to quit: "))
        if num > 0 and num <= len(lines):
            print(f"Line {num}: {lines[num - 1]}")
        elif num == 0:
            print("Thank you for using the program.")
            break
        else:
            print("Invalid line number.")
```

This code allows the user to view specific lines of a file by entering a line number. The program will prompt the user until 0 is entered to exit the loop. The line's contents are displayed with its number if the input is valid. If the input is invalid, "Invalid line number" is displayed. The input must be a number, or else an error will occur.



```
Enter the input file name: LR2_2.py
The file has 25 lines.
Enter line number (0 to quit): 1
def navigate_file():

Enter line number (0 to quit): 24
  navigate_file()

Enter line number (0 to quit): 15
    line_number = int(input("Enter line number (0 to quit): "))

Enter line number (0 to quit): 12
  print(f"The file has {num_lines} lines.")

Enter line number (0 to quit): 7
  except FileNotFoundError:

Enter line number (0 to quit): 20
    else:

Enter line number (0 to quit): 0
Press any key to continue . . .
```

Figure 5. Output of the program

After executing the Python file in the Anaconda prompt, Figure 5 indicates a successful debugging process without errors.

3. **Generator_modified.py**

Modify the sentence-generator program of Case Study 5.3:

- METIS book: 9781337671019, page 150.0020
- Python source code: generator.py

so that it inputs its vocabulary from a set of text files at startup. The filenames are nouns.txt, verbs. txt, articles.txt, and prepositions.txt. (Hint: Define a single new function, getWords. This function should expect a filename as an argument. The function should open an input file with this name, define a temporary list, read words from the file, and add them to the list. The function should then convert the list to a tuple and return this tuple. Call the function with an actual filename to initialize each of the four variables for the vocabulary.)