Topics Covered in the lecture:

- **File I/O Basics**: The class covers how to handle input and output using files in Python, moving beyond terminal-based inputs and outputs to file-based operations for persistent data handling.
- **File Operations:** It explains file opening modes (read, write, append), reading (lines, specific bytes), writing, and closing files, emphasising the importance of managing file resources properly.
- **Practical Applications:** Examples include reading and verifying student marks, writing data to new files, and handling text file encodings and whitespace efficiently

Practice Questions

Note: Sample test case files can be downloaded from
https://filesamples.com/formats/txt

Some output/error finding/mcq type questions for practice:

Q1: What will be the output of the following code, and how many newlines will be in the file "data.txt" after execution?

```
f = open("data.txt", "w")
f.write("Line 1\n")
f.write("Line 2")
f.close()
```

Q2: Given the following content in a file "data.txt":

Good Morning, Everybody!
Lovely day in Argentina!

What will be the output of the following code?

```
f = open("data.txt", "r")
print(f.read(12))
f.seek(0)
print(f.readline())
f.close()
```

Q3: You have a 1GB text file "largefile.txt", and you need to create a new file "filtered.txt" that contains only the lines which have more than 3 words. How would you efficiently accomplish this without loading the entire file into memory?

Q4: Consider the following code snippet. What will happen if the file "data.txt" has less than 5 lines?

```
with open("data.txt", "r") as f:
    for _ in range(5):
        print(f.readline().strip())
```

Q5: What will be the result of running the following code snippet?

```
f = open("example.txt", "w")
f.write("Writing to a file.")
f.close()
f.write("Attempting to write again.")  # Trying to write after closing
```

Q6: What will be the output of the following code?

Assume that numbers.txt contains the following:

5
Hello
74

```
with open("numbers.txt", "r") as f:
    for line in f:
```

```
        number = int(line)  # Assuming each line should contain
an integer
```

Write code for the given problems:

Q1: Write a Python program to extract words from various text files and store the frequency of unique words in a dictionary.

Q2: Write a Python program that reads a text file character by character and performs the following tasks:

- For each unique character (excluding whitespace characters), store the character along with its ASCII value in a tuple.
- Use a set to keep track of all unique characters encountered in the file.
- At the end of the program, display the set of unique characters and a list of tuples containing characters and their corresponding ASCII values, sorted by ASCII value.

Q3: You are tasked with creating a text analysis tool that reads from a text file named input.txt and performs various operations to analyze the text content. Your program should include the following features:

1. Read the Entire File: Use f.read() to read the entire content of the file into a string. Count the total number of characters in the file (excluding whitespace characters) and display the count.

2. Count Unique Words:

- Read the file line by line using f.readline() and split each line into words.
- Store each unique word in a set (ignoring case) and keep a count of how many times each word appears in a dictionary.
- Display the total number of unique words.

3. Create a List of Tuples:

- Create a list of tuples, where each tuple contains a word and its corresponding count.
- Sort this list by word frequency (highest to lowest) and alphabetically for words with the same frequency.

4. Analyze Line Lengths:

- Use f.readlines() to read all lines into a list.
- Create a dictionary where the keys are line numbers (starting from 1) and the values are the lengths of each line (excluding whitespace).
- Display the line lengths in a formatted manner.

5. Read N Characters:

- Use f.read(n) to read the first n characters of the file (where n can be set as 100 for this example).
- Display these characters and the total number of characters read.

6. Output Summary:

- Finally, output a summary that includes:
- Total characters (excluding whitespace)
- Total unique words
- List of tuples (word, count) sorted by frequency
- Line lengths in a formatted dictionary
- First n characters of the file

Note: For simplicity, assume that there are no special characters in the file

Q4. Write a Python program to merge the contents of two files "file1.txt" and "file2.txt" into a new file "merged.txt", alternating lines from each file. If one file has more lines, the remaining lines should be appended at the end.

file1.txt

```
Writing to a file.
Testing the code
How are you?
```

file2.txt

```
What are you doing?
The code works well.
I am good.
```

merged.txt

```
Writing to a file.
What are you doing?
Testing the code
The code works well.
How are you?
I am good.
```

Approach:

Step 1: Open file1.txt and file2.txt in read mode and open merged.txt in the write mode

Step 2: Get the list of lines of file1.txt and file2.txt

Step 3: Find the maximum number of lines present in the files. (max(file1_lines, file2_lines))

Step 4: Use looping (0 to max_lines-1) and if statements to write alternating lines from each file. If all lines of one file has been written to merged, then the remaining lines of the other (larger) file should be appended.

Extra Questions:

Note: Sample test case files can be downloaded from
https://filesamples.com/formats/bin

Q: Suppose you are working with a binary file "data.bin" that contains a sequence of 32-bit integers. How would you read these integers from the file, sum them, and write the result into a new file "sum.txt"?

Note: You can read “.bin” files using “open("file_path", "rb")”