Practical 7

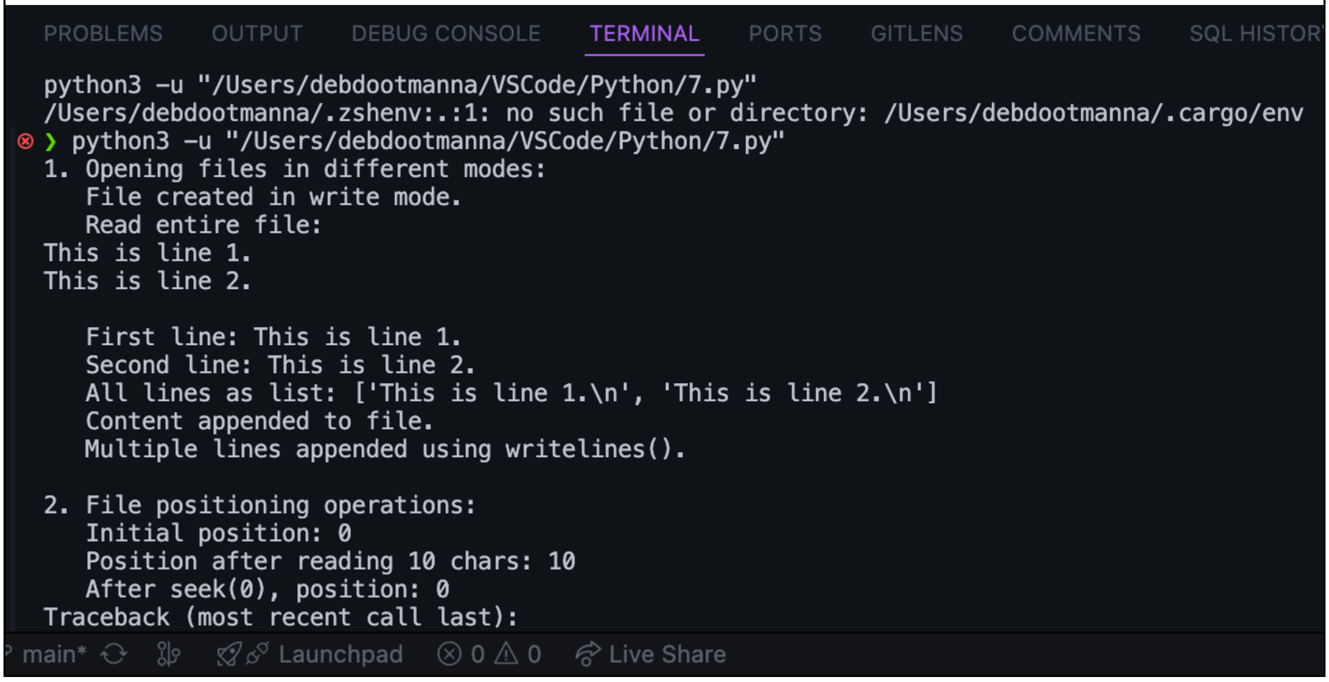
7 File Operation

Aim: Open a file in read, write, or append mode using the open function.
Read the entire content of a file using read(), readline(), readlines().
Write data to a file using write() and writelines()
Append data to the end of a file using write() in append mode.
Perform the different file operations (close, tell, seek, os.path.exists, os.remove, os.rename, os.path)

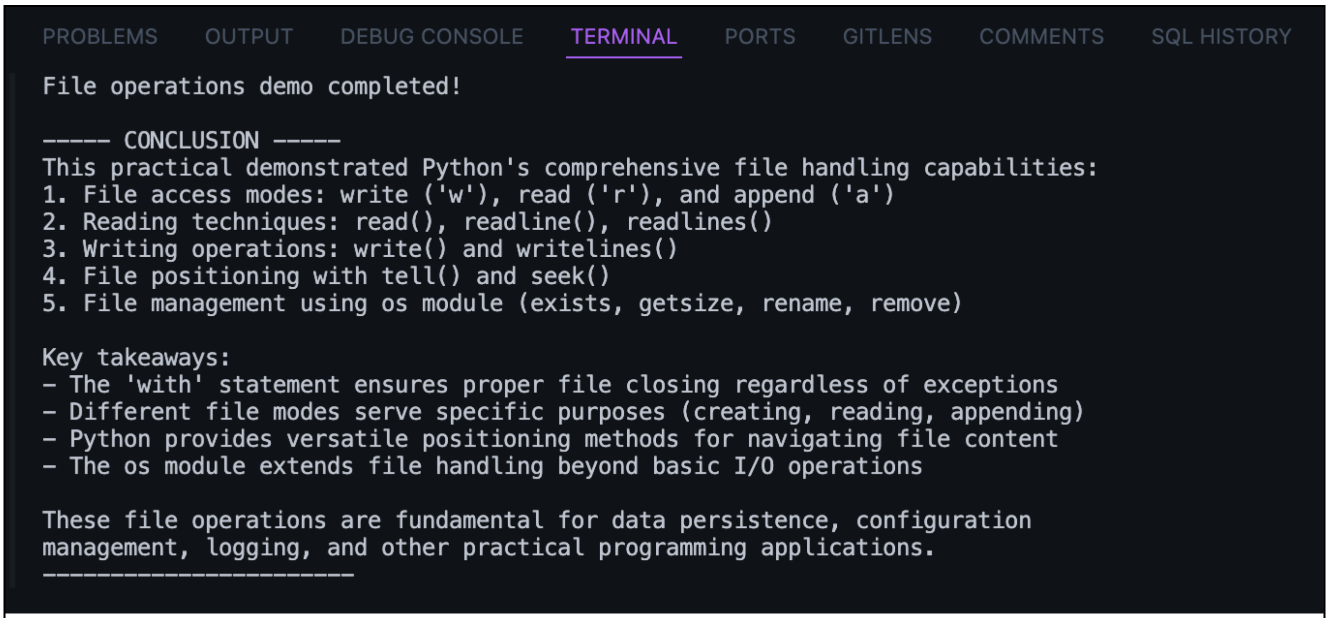
Code:

```python
import os

def file_operations_demo():
    # 1. Opening files in different modes
    print("1. Opening files in different modes:")

    # Write mode ('w') - creates new file or overwrites existing
    with open('example.txt', 'w') as file:
        file.write("This is line 1.\n")
        file.write("This is line 2.\n")
    print("   File created in write mode.")

    # Read mode ('r') - default mode
    with open('example.txt', 'r') as file:
        # Read entire file content
        content = file.read()
        print(f"   Read entire file:\n{content}")

    # Reading line by line
    with open('example.txt', 'r') as file:
        # Read first line
        first_line = file.readline()
        print(f"   First line: {first_line.strip()}")

        # Read next line
        second_line = file.readline()
        print(f"   Second line: {second_line.strip()}")

    # Reading all lines into a list
    with open('example.txt', 'r') as file:
        lines = file.readlines()
        print(f"   All lines as list: {lines}")

    # Append mode ('a') - adds to end of file
    with open('example.txt', 'a') as file:
        file.write("This is line 3 (appended).\n")
    print("   Content appended to file.")

    # Writing multiple lines at once
    with open('example.txt', 'a') as file:
        lines_to_add = ["Line 4 from list.\n", "Line 5 from list.\n"]
        file.writelines(lines_to_add)
    print("   Multiple lines appended using writelines().")

    # 2. File positioning operations
    print("\n2. File positioning operations:")
    with open('example.txt', 'r') as file:
        # Get current position
        position = file.tell()
        print(f"   Initial position: {position}")

        # Read some content
        file.read(10)
        position = file.tell()
        print(f"   Position after reading 10 chars: {position}")

        # Seek to specific position
        file.seek(0)  # Go back to beginning
        print(f"   After seek(0), position: {file.tell()}")

        # Seek from current position
        file.seek(5, 1)  # Move 5 chars forward from current position
        print(f"   After seek(5, 1), position: {file.tell()}")

    # 3. File management operations
    print("\n3. File management operations:")

    # Check if file exists
    file_exists = os.path.exists('example.txt')
    print(f"   Does example.txt exist? {file_exists}")

    # Get file information
    file_size = os.path.getsize('example.txt')
    print(f"   Size of example.txt: {file_size} bytes")

    # Rename a file
    os.rename('example.txt', 'renamed_example.txt')
    print("   File renamed to 'renamed_example.txt'")
    # Create a copy to demonstrate removal
    with open('to_be_deleted.txt', 'w') as file:
        file.write("This file will be deleted.")

    # Remove a file
    os.remove('to_be_deleted.txt')
    print("   File 'to_be_deleted.txt' removed")

    # Show current directory path
    current_dir = os.path.abspath(os.path.dirname(__file__))
    print(f"   Current directory: {current_dir}")

    print("\nFile operations demo completed!")

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

Output Screenshot:



```
python3 -u "/Users/debdootmanna/VSCode/Python/7.py"
/Users/debdootmanna/.zshenv:.:1: no such file or directory: /Users/debdootmanna/.cargo/env
⊗ ❯ python3 -u "/Users/debdootmanna/VSCode/Python/7.py"
1. Opening files in different modes:
   File created in write mode.
   Read entire file:
This is line 1.
This is line 2.

   First line: This is line 1.
   Second line: This is line 2.
   All lines as list: ['This is line 1.\n', 'This is line 2.\n']
   Content appended to file.
   Multiple lines appended using writelines().

2. File positioning operations:
   Initial position: 0
   Position after reading 10 chars: 10
   After seek(0), position: 0
Traceback (most recent call last):
```



```
File operations demo completed!

----- CONCLUSION -----
This practical demonstrated Python's comprehensive file handling capabilities:
1. File access modes: write ('w'), read ('r'), and append ('a')
2. Reading techniques: read(), readline(), readlines()
3. Writing operations: write() and writelines()
4. File positioning with tell() and seek()
5. File management using os module (exists, getsize, rename, remove)

Key takeaways:
- The 'with' statement ensures proper file closing regardless of exceptions
- Different file modes serve specific purposes (creating, reading, appending)
- Python provides versatile positioning methods for navigating file content
- The os module extends file handling beyond basic I/O operations

These file operations are fundamental for data persistence, configuration
management, logging, and other practical programming applications.
-----------------------
```

Conclusion/Summary:

In this practical assignment, I explored fundamental file handling operations in Python, which are essential skills for any programmer. I learned how to:

Open files in different modes:

Write mode ('w') for creating new files or overwriting existing ones
Read mode ('r') for accessing file contents
Append mode ('a') for adding content to existing files
Implement various reading techniques:

read() for getting entire file content
readline() for reading a single line
readlines() for obtaining all lines as a list
Write content to files using:

write() for adding single strings
writelines() for adding multiple lines from a list
Manipulate file positions with:

tell() to determine current position
seek() to move to specific positions within files
Perform file management operations using the os module:

Checking file existence with os.path.exists()
Getting file size with os.path.getsize()
Renaming files with os.rename()
Removing files with os.remove()
Finding directory paths with os.path functions
I also practiced using the 'with' statement for proper file handling, which ensures files are correctly closed even if exceptions occur during processing.

These operations form the foundation for working with persistent data in Python applications and will be valuable for future programming tasks involving data storage, configuration management, and log processing.

Student Signature & Date	**Marks:**	**Evaluator Signature & Date**