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:**  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:** | | |
| **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** |