Task-6 Implement various text file operation

Aim:

To write a python program implement various text file operations.

Problem 6.1:

You need to write the sentence "Error objects are thrown when runtime errors occurs. The Error object can also be used as a base object for user-defined ~~exce~~ exceptions" into a text file named log.txt. Implement a function that performs this task.

Algorithm:

1. Write to a file:
   - Define writefile(filename) functions:
     i. Open a file named "log.txt" in write mode.
     ii. Write the following text to the file:
     iii. "Error objects are thrown when runtime errors occur. The error object can also be used as a base object for user-defined exceptions".
     iv. Close the file.
2. Read from a file:
   - Define readfile(filename) function:
     i. Open the file specified by filename in read mode using a with statement.
     ii. Read the entire content of the file.
     iii. Print the content.
3. Execute the program:
   - Call writefile("write") to write the predefined text to "log.txt".
   - Call readfile("text") to attempt to read from a file named

Output:

Error objects are thrown when runtime errors occur. The Error objects can also be used as a base object for user-defined exceptions.

```
def writefile(filename):
    f=open("log.txt","w")
    f.write("Error objects are thrown runtime errors occurs. The
    Error object can also be used as a base for user-defined
    expections").
    f.close()
def readfile(filename):
    with open(filename, ~~"read"~~ "r") as file:
    ~~Content~~ content = file.read()
    ~~Print~~ print(content)
writefile("write")
readfile("log.txt")
```

"text and print its content.

Program 6.1

```
def writefile(filename):
f=open("log.txt"), "w")
f.write("Error objects are thrown when runtime errors occurs.
The Error object can also be used as a base for user-defined
expections").
f.close()
def readfile(filename):
with open(filename,"r") as file:
Content=file.read()
Print(content)
writefile("write")
readfile("text")
```

Problem 6.2

You have a text file log.txt containing logs of a system. Write a function that counts the number of lines containing the word "ERROR".

Algorithm:

1. Initialize Error Counter:
   - Define the function Count_error_lines(filename):
     - Initialize error_count to 0.

2 Open and Read file:

- open the file specified by filename in read mode using a with statement.

Output:

Number of lines with 'ERROR' is 2.

3. Check Each line for "ERROR":

- Loop through each line in the file:
  - If the line contains the word "ERROR", increment error_count by 1.

4. Return Error count:

- After reading all the lines, return the value of error_count.

5. Execute the program:

- Call count_error_lines("log.txt") to count the number of lines with the word "ERROR" in the file "log.txt".
- Print the result with the message: "Number of lines with 'ERROR': {error_lines}".

Program 6.2:

```
def count_error_lines(filename):
    error_count=0
    with open(filename,"r") as file:
        for line in file:
            if "ERROR" in line:
                error_count+=1
    return error_count

error_lines = count_error_lines("log.txt")
Print(f"Number of lines with 'ERROR': {error_lines}")
```

log.txt

* Error object are thrown when runtime Error occur.

The Error object can also be used as a base object for user-defined exceptions".

Problem 6.31

You need to write a report containing the details (Name, departments) of the employee in list. Write a Python function that writes the report to a file named employee_report.txt.

Algorithm:

1. Create employee Data:
   - Define the function write_employee_report (filename):
     - Create a list employees contains dictionaries, each with "name" and ~~"development"~~ "department" keys for individual employees.
2. Open file for writing:
   - Open the file specified by filename in write mode using statement.
3. Write employee Data to file:
   - Loop through each employee in the employees list.
   - For each employee, format a string as "Name: {employee['name']}, Department: {employee['department']}".
   - Write the formatted string to the file, followed by a newline character (\n).
4. Execute the Program:
   - Call write_employee_report("employee_report.txt") to write the employee data to the file "employee_report.txt".

output:

Name: Alice, Department: HR

Name: Bob, Department: Engineering

Name: Charlie, Department: Finance.

Program 6.3:

```
def write_employee_report(filename):
    employees = [{"name": "Alice", "department": "HR"}, {"name": "Bob",
    "department": "Engineering"}, {"name": "Charlie", "department":
    "Finance"}]
    with open(filename, "w") as file:
        for employee in employees:
            line = f"Name: {employee['name']}, Department: {employee['depar-
            tment']}\n"
            file.write(line)

# Example usage:
write_employee_report("employee_report.txt")
```

VEL TECH	
EX No.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	25
WITH DATE	10/9

Result:

Thus, the Python Program implement various text file operations was successfully excuted and the output is verified.