file handling

```
file =open("/content/drive/MyDrive/lithin m/LITHIN HM.txt","r")
content = file.read()
print(content)
file.close()
```

```
h vbhjghgvhjhjghjlithin ME
```

```
file =open("/content/drive/MyDrive/lithin m/LITHIN HM.txt","w")
file.write("hello,world!\n")
file.close()
```

APPENDING TO A FILE

```
file =open("/content/drive/MyDrive/lithin m/LITHIN HM.txt","a")
file.write("hello,world!\n")
file.close()
```

FILE HANDLING MODULES

```
with open("/content/raja.jpg","rb") as file:
  data = file.read()


try:
  number =int(input("enter the number:"))
  print(10/number)
except ZeroDivisionError:
  print("you cannot divide by zero")
except ValueError:
  print("invaild input! please enter a number")
```

```
enter the number:10
1.0
```

```
try:
  file = open ("/content/drive/MyDrive/lithin m/LITHIN HM.txt","r")
except FileNotFoundError:
  print("file not found.")
finally:
  print("execuation complete.")


execuation complete.
```

```
def check_age(age):
  if age<18:
```

```
    raise ValueError("Age must be 18 or oldr")
  return True

try:
  check_age(16)
except ValueError as e:
  print(e)
```

```
Age must be 18 or oldr
```

create and wright the file

```
with open('data.txt', 'w') as file:
    file.write("Hello, World!\n")
```

read from a file

```
with open('data.txt', 'r') as file:
    print("Contents of data.txt:")
    print(file.read())
```

```
Contents of data.txt:
Hello, World!
```

append to the file

```
with open('data.txt', 'a') as file:
    file.write("Welcome to Python programming!\n")
```

count linse in file

```
with open('data.txt', 'r') as file:
    lines = file.readlines()
    print(f"Number of lines in data.txt: {len(lines)}")
```

```
Number of lines in data.txt: 2
```

count words in a file

```
with open('data.txt', 'r') as file:
    text = file.read()
    words = text.split()
    print(f"Number of words in data.txt: {len(words)}")
```

```
Number of words in data.txt: 6
```

Copy File Contents

```
with open('data.txt', 'r') as src_file:
    content = src_file.read()

with open('copy.txt', 'w') as dest_file:
    dest_file.write(content)
print("Contents copied to copy.txt.")

Contents copied to copy.txt.
```

Check if File Exists

```
file_exists = os.path.exists('data.txt')
print(f"Does data.txt exist? {file_exists}")
```

Read File Line by Line

```
with open('data.txt', 'r') as file:
    print("Reading data.txt line by line:")
    for line in file:
        print(line.strip())

Reading data.txt line by line:
Hello, World!
Welcome to Python programming!
```

Search for a Word in a File

```
search_word = "Python"
with open('data.txt', 'r') as file:
    lines_with_word = [line for line in file if search_word in line]
    print(f"Lines containing '{search_word}':")
    for line in lines_with_word:
        print(line.strip())

Lines containing 'Python':
Welcome to Python programming!
```

Write a List to a File

```
numbers = [1, 2, 3, 4, 5]
with open('numbers.txt', 'w') as file:
    for number in numbers:
        file.write(f"{number}\n")
print("List of numbers written to numbers.txt.")

List of numbers written to numbers.txt.
```

Reverse File Contents

```
with open('data.txt', 'r') as file:
    content = file.readlines()

with open('reverse.txt', 'w') as file:
    for line in reversed(content):
        file.write(line)
print("Contents of data.txt written in reverse order to reverse.txt.")
```

```
Contents of data.txt written in reverse order to reverse.txt.
```

File Statistics

```
with open('data.txt', 'r') as file:
    content = file.read()
    num_chars = len(content)
    num_words = len(content.split())
    num_lines = len(content.splitlines())

print(f"File statistics for data.txt:")
print(f"Characters: {num_chars}, Words: {num_words}, Lines:
{num_lines}")
```

```
File statistics for data.txt:
Characters: 45, Words: 6, Lines: 2
```

Merge Two Files

```
with open('data.txt', 'r') as file1, open('numbers.txt', 'r') as
file2, open('merged.txt', 'w') as merged_file:
    merged_file.write(file1.read())
    merged_file.write("\n")
    merged_file.write(file2.read())
print("Contents of data.txt and numbers.txt merged into merged.txt.")
```

```
Contents of data.txt and numbers.txt merged into merged.txt.
```

Count Occurrences of a Word

```
def count_word_occurrences(file_name, word):
    with open(file_name, 'r') as file:
        content = file.read()
    return content.lower().split().count(word.lower())

word_count = count_word_occurrences('data.txt', 'Python')
print(f"Occurrences of the word 'Python': {word_count}")
```

```
Occurrences of the word 'Python': 1
```

1. Remove a Word from a File

```
def remove_word_from_file(file_name, word_to_remove):
    with open(file_name, 'r') as file:
        content = file.read()

    content = content.replace(word_to_remove, '')

    with open(file_name, 'w') as file:
        file.write(content)

remove_word_from_file('data.txt', 'Hello')
print("Removed the word 'Hello' from data.txt.")
```

```
Removed the word 'Hello' from data.txt.
```

1. File Encryption (Caesar Cipher)

```
def encrypt_file(file_name, shift):
    with open(file_name, 'r') as file:
        content = file.read()

    encrypted_content = ''.join([chr((ord(char) + shift) % 256) for
char in content])

    with open('encrypted.txt', 'w') as file:
        file.write(encrypted_content)

encrypt_file('data.txt', 2)
print("Encrypted data.txt and saved to encrypted.txt.")
```

```
Encrypted data.txt and saved to encrypted.txt.
```

File Decryption (Caesar Cipher)

```
def decrypt_file(file_name, shift):
    with open(file_name, 'r') as file:
        encrypted_content = file.read()

    decrypted_content = ''.join([chr((ord(char) - shift) % 256) for
char in encrypted_content])

    with open('decrypted.txt', 'w') as file:
        file.write(decrypted_content)

decrypt_file('encrypted.txt', 2)
print("Decrypted encrypted.txt and saved to decrypted.txt.")
```

```
Decrypted encrypted.txt and saved to decrypted.txt.
```

Remove Blank Lines from a File

```
def remove_blank_lines(file_name):
    with open(file_name, 'r') as file:
        lines = file.readlines()

    non_blank_lines = [line for line in lines if line.strip()]

    with open(file_name, 'w') as file:
        file.writelines(non_blank_lines)

remove_blank_lines('data.txt')
print("Removed blank lines from data.txt.")
```

```
Removed blank lines from data.txt.
```

1. Find Longest Word in a File

```
from collections import Counter

def word_frequency_analysis(file_name):
    with open(file_name, 'r') as file:
        content = file.read()

    words = content.split()
    frequency = Counter(words)

    return frequency

word_frequencies = word_frequency_analysis('data.txt')
print("Word frequency analysis for data.txt:")
for word, freq in word_frequencies.items():
    print(f"{word}: {freq}")
```

```
Word frequency analysis for data.txt:
,: 1
World!: 1
Welcome: 1
to: 1
Python: 1
programming!: 1
```

1. Handle File Not Found Error

```
def handle_file_not_found(file_name):
    try:
        with open(file_name, 'r') as file:
            content = file.read()
        print("File content read successfully.")
    except FileNotFoundError:
        print(f"Error: The file {file_name} does not exist.")

handle_file_not_found('data.txt')  # Test with a non-existing file
```

```
File content read successfully.
```

1. Handle Division by Zero

```
try:
    a = int(input("Enter the numerator: "))
    b = int(input("Enter the denominator: "))
    result = a / b
    print(f"The result is: {result}")
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")

Enter the numerator: 5
Enter the denominator: 5
The result is: 1.0
```

1. Invalid Input Handling

```
try:
    user_input = int(input("Please enter an integer: "))
    print(f"You entered: {user_input}")
except ValueError:
    print("Error: Invalid input. Please enter a valid integer.")

Please enter an integer: 2
You entered: 2
```

1. Handle KeyError

```
my_dict = {"name": "Alice", "age": 30}
try:
    key = input("Enter a key to access in the dictionary: ")
    value = my_dict[key]
    print(f"Value for {key}: {value}")
except KeyError:
    print("Error: The key does not exist in the dictionary.")

Enter a key to access in the dictionary: 5
Error: The key does not exist in the dictionary.
```

1. File Read Permission

```
try:
    with open("data.txt", "r") as file:
        content = file.read()
        print(content)
except PermissionError:
    print("Error: You do not have permission to read the file.")
```

```
, World!
Welcome to Python programming!
```

Catch Multiple Exceptions

```
try:
    with open("data.txt", "r") as file:
        content = file.read()
        print(content)
except FileNotFoundError:
    print("Error: The file was not found.")
except PermissionError:
    print("Error: You do not have permission to access the file.")

, World!
Welcome to Python programming!
```

1. Custom Exception (NegativeNumberError)

```
class NegativeNumberError(Exception):
    def __init__(self, message="Negative number is not allowed"):
        self.message = message
        super().__init__(self.message)

try:
    number = int(input("Enter a number: "))
    if number < 0:
        raise NegativeNumberError()
    print(f"You entered: {number}")
except NegativeNumberError as e:
    print(f"Error: {e}")

Enter a number: 2
You entered: 2
```

1. Handle IndexError

```
my_list = [1, 2, 3, 4, 5]
try:
    index = int(input("Enter the index you want to access: "))
    print(f"The value at index {index} is: {my_list[index]}")
except IndexError:
    print("Error: Index is out of range.")

Enter the index you want to access: 4
The value at index 4 is: 5
```

1. Nested Exception Handling

```
try:
    try:
        a = int(input("Enter a number: "))
        b = int(input("Enter another number: "))
        result = a / b
        print(f"Result: {result}")
    except ZeroDivisionError:
        print("Error: Division by zero.")
except ValueError:
    print("Error: Invalid input. Please enter a valid number.")

Enter a number: 6
Enter another number: 3
Result: 2.0
```

1. Resource Cleanup with finally

```
try:
    file = open("data.txt", "r")
    content = file.read()
    print(content)
except FileNotFoundError:
    print("Error: File not found.")
finally:
    file.close()
    print("File is now closed.")

, World!
Welcome to Python programming!

File is now closed.
```