File Handling and Error Handling in Python

File operations

```
#creating a file
file_object=open("/content/misbah.txt","r")

#Modes:
#reading a file
file_object=open("/content/misbah.txt","r")
print(file_object.read())
```

```
MISBAH HARMAIN
```

```
#reading all lines
file_object=open("/content/drive/MyDrive/misbah.txt","r")
print(file_object.readlines())
```

```
['python is awesome!\n', 'python is easy to learn']
```

```
#writing to a file
file_object=open("/content/misbah.txt","w")
file_object.write("hello world")
```

```
11
```

```
#appending to a file
file_object=open("/content/misbah.txt","a")
file_object.write("hello india")
```

```
11
```

```
#using with statement
with open("/content/misbah.txt","r") as file_object:
  print(file_object.read())
```

```
hello world hello india
```

```
#file handling modes
#binary mode
with open("/content/image.jpg","rb")as file:
  data=file.read()
```

Error Handling

```
#try-expecpt block
try:
  num=int(input("enter a number:"))
  print(10/num)
except ZeroDivisionError:
  print("cannot divide by zero")
```

```
except ValueError:
  print("Invalid input! please enter a valid number")
```

```
enter a number:0
cannot divide by zero
```

```
#finally block
try:
  file=open("/content/misbah.txt","r")
except FileNotFoundError:
  print("file not found.")
finally:
  print("exception complete.")
```

```
file not found.
exception complete.
```

```
#raising exceptions
def check_age(age):
  if age<18:
    raise ValueError("Age must be 18 or older.")
  return true

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

```
Age must be 18 or older.
```

Hands on prctice

```
#reading and writing to a file
with open("/content/misbah.txt","w")as file:
  file.write("python is awesome!\n")

  with open("/content/misbah.txt","r")as file:
    print(file.read())



#appending data to a file
with open("/content/misbah.txt","a")as file:
  file.write("python is easy to learn.\n")
  with open("/content/misbah.txt","r")as file:
    print(file.read())
```

```
python is easy to learn.
```

```
#handling division by zero error
try:
  num1=int(input("enter a number:"))
  num2=int(input("enter another number:"))
  result=num1/num2
  print("result:",result)
except ZeroDivisionError:
  print("cannot divide by zero")
except ValueError:
  print("invalid input! please enter a valid number")
```

```
enter a number:556
enter another number:765
result: 0.726797385620915
```

```
#creating a custom exception
class NegativeNumberError(Exception):
  pass
def check_positive(number):
  if number<=0:
    raise NegativeNumberError("Negative number entered.")

try:
  num=int(input("enter a number:"))
  check_positive(num)
  print("You entered a positive number.")
except NegativeNumberError as e:
  print(e)
```

```
enter a number:5
You entered a positive number.
```

```
#count words in a file
def count_words_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            return len(file.read().split())
    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

file_path = input("Enter the path to the file: ")
word_count = count_words_in_file(file_path)
if word_count:
    print(f"The file contains {word_count} words.")
```

```
Enter the path to the file: misbah
Error: The file 'misbah' was not found.
```

```
#copy file contents
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

words = content.split()

word_count = len(words)

print(f"The number of words in data.txt is: {word_count}")
```

```
The number of words in data.txt is: 8
```

```
#check if file exists
with open('/content/drive/MyDrive/misbah.txt', 'r') as source_file,
open('copy.txt', 'w') as destination_file:
    content = source_file.read()

    destination_file.write(content)

print("Contents of data.txt have been copied to copy.txt")
```

```
Contents of data.txt have been copied to copy.txt
```

```
#check if file exist
import os

if os.path.exists('/content/drive/MyDrive/misbah.txt'):
    print("The file 'data.txt' exists in the current directory.")
else:
    print("The file 'data.txt' does not exist in the current
directory.")
```

```
The file 'data.txt' exists in the current directory.
```

```
#read file line by line
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    # Read and print each line in the file
    for line in file:
        print(line, end='')
```

```
python is awesome!
python is easy to learn
```

```
#search for a word in a file
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    line_number = 1

    for line in file:
        if 'Python' in line:
          print(f"Line {line_number}: {line.strip()}")
        line_number += 1
```

```python
#write a list to a file
# List of numbers
numbers = [1, 2, 3, 4, 5]

# Open the file 'numbers.txt' in write mode
with open('numbers.txt', 'w') as file:
    # Write each number from the list to the file
    for number in numbers:
        file.write(f"{number}\n")

print("Numbers have been written to 'numbers.txt'")
```

```
Numbers have been written to 'numbers.txt'
```

```python
#reverse file contents
# Open 'data.txt' in read mode
with open('/content/drive/MyDrive/misbah.txt', 'r') as source_file:
    # Read all lines from the file
    lines = source_file.readlines()

# Open 'reverse.txt' in write mode
with open('reverse.txt', 'w') as destination_file:
    # Write the lines to the new file in reverse order
    for line in reversed(lines):
        destination_file.write(line)

print("Contents of '/content/drive/MyDrive/misbah.txt' have been
written in reverse order to 'reverse.txt'")
```

```
Contents of '/content/drive/MyDrive/misbah.txt' have been written in
reverse order to 'reverse.txt'
```

```python
#file statistics
line_count = 0
word_count = 0
char_count = 0

# Open '/content/drive/MyDrive/misbah.txt' in read mode
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    # Iterate through each line in the file
    for line in file:
        line_count += 1  # Increment line count
        char_count += len(line)  # Add the number of characters in the
line
        word_count += len(line.split())  # Split the line into words
and count them

# Display the results
print(f"Number of lines: {line_count}")
print(f"Number of words: {word_count}")
print(f"Number of characters: {char_count}")
```

```
Number of lines: 2
Number of words: 8
Number of characters: 42
```

```python
#merge two files
# Open the two files (data.txt and numbers.txt) and read their
contents
with open('/content/drive/MyDrive/misbah.txt', 'r') as file1,
open('numbers.txt', 'r') as file2:
    data_content = file1.read()
    numbers_content = file2.read()

# Open the merged.txt file and write the combined content
with open('merged.txt', 'w') as merged_file:
    merged_file.write(data_content + '\n' + numbers_content)

#count occurance of a word
# Open the data.txt file and read its content
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

# Count how many times 'Python' appears in the content
python_count = content.lower().count('python')

# Print the result
print(f"The word 'Python' appears {python_count} times.")
```

```
The word 'Python' appears 2 times.
```

```python
#remove a word from a file
# Open the file in read mode
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

# Replace the word 'Hello' with an empty string
modified_content = content.replace('Hello', '')

# Open the file in write mode to save the modified content
with open('/content/drive/MyDrive/misbah.txt', 'w') as file:
    file.write(modified_content)

print("The word 'Hello' has been removed from data.txt.")
```

```
The word 'Hello' has been removed from data.txt.
```

```python
#file encryption
# Function to encrypt text using Caesar Cipher
def caesar_cipher(text, shift):
    encrypted_text = ''
    for char in text:
        # Encrypt only alphabetic characters
        if char.isalpha():
            # Shift character within the alphabet
            shift_base = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) - shift_base + shift) % 26
+ shift_base)
            encrypted_text += encrypted_char
        else:
            # Keep non-alphabet characters as they are
            encrypted_text += char
    return encrypted_text

# Open the /content/drive/MyDrive/misbah.txt file and read its content
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

# Encrypt the content with a shift of 2
shift = 2
encrypted_content = caesar_cipher(content, shift)

# Save the encrypted content to encrypted.txt
with open('encrypted.txt', 'w') as encrypted_file:
#file encryption
# Function to encrypt text using Caesar Cipher
 def caesar_cipher(text, shift):
    encrypted_text = ''
    for char in text:
        # Encrypt only alphabetic characters
        if char.isalpha():
            # Shift character within the alphabet
            shift_base = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) - shift_base + shift) % 26
+ shift_base)
            encrypted_text += encrypted_char
        else:
            # Keep non-alphabet characters as they are
            encrypted_text += char
    return encrypted_text

# Open the /content/drive/MyDrive/misbah.txt file and read its content
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

# Encrypt the content with a shift of 2
shift = 2
encrypted_content = caesar_cipher(content, shift)

# Save the encrypted content to encrypted.txt
with open('encrypted.txt', 'w') as encrypted_file:
    encrypted_file.write(encrypted_content)
```

```
print("The contents of data.txt have been encrypted and saved to
encrypted.txt")
```

```
The contents of data.txt have been encrypted and saved to
encrypted.txt
```

```
#file decryption
# Function to decrypt text using Caesar Cipher (shift by -2)
def caesar_cipher_decrypt(text, shift):
    decrypted_text = ''
    for char in text:
        # Decrypt only alphabetic characters
        if char.isalpha():
            # Shift character within the alphabet (reversing the
shift)
            shift_base = 65 if char.isupper() else 97
            decrypted_char = chr((ord(char) - shift_base - shift) % 26
+ shift_base)
            decrypted_text += decrypted_char
        else:
            # Keep non-alphabet characters as they are
            decrypted_text += char
    return decrypted_text

# Open the encrypted.txt file and read its content
with open('encrypted.txt', 'r') as file:
    encrypted_content = file.read()

# Decrypt the content with a shift of -2 (reverse the encryption)
shift = 2
decrypted_content = caesar_cipher_decrypt(encrypted_content, shift)

# Save the decrypted content back to a new file (optional, e.g.,
decrypted.txt)
with open('decrypted.txt', 'w') as decrypted_file:
    decrypted_file.write(decrypted_content)

print("The content has been decrypted and saved to decrypted.txt.")
```

```
The content has been decrypted and saved to decrypted.txt.
```

```
#remove blank lines
# Open the data.txt file and read its content
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    lines = file.readlines()

# Remove blank lines (lines that are empty or contain only whitespace)
non_blank_lines = [line for line in lines if line.strip() != '']

# Write the cleaned content back to the file
with open('/content/drive/MyDrive/misbah.txt', 'w') as file:
    file.writelines(non_blank_lines)

print("All blank lines have been removed from data.txt.")
```

```
All blank lines have been removed from data.txt.
```

```
#find longest word in a file
import re

# Function to find the longest word in a text
def find_longest_word(text):
    # Use regular expression to find all words (sequences of 
alphabetic characters)
    words = re.findall(r'\b\w+\b', text)

    # Find the longest word (the one with the maximum length)
    longest_word = max(words, key=len) if words else None
    return longest_word

# Open the data.txt file and read its content
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

# Find the longest word in the content
longest_word = find_longest_word(content)

if longest_word:
    print(f"The longest word in the file is: {longest_word}")
else:
    print("No words found in the file.")
```

```
The longest word in the file is: awesome
```

```
#word frequency analysis
import re
from collections import Counter

# Function to count the frequency of each word in the text
def count_word_frequency(text):
    # Use regular expression to find all words (sequences of 
alphabetic characters)
    words = re.findall(r'\b\w+\b', text.lower())  # Convert to 
lowercase for case-insensitive counting

    # Count the frequency of each word using Counter
    word_frequency = Counter(words)

    return word_frequency

# Open the data.txt file and read its content
```

```
with open('/content/drive/MyDrive/misbah.txt', 'r') as file:
    content = file.read()

# Count the frequency of each word
word_frequency = count_word_frequency(content)

# Display the results
for word, frequency in word_frequency.items():
    print(f"'{word}': {frequency}")
```

```
'python': 2
'is': 2
'awesome': 1
'easy': 1
'to': 1
'learn': 1
```

```
#handle file not found error
# Function to read and process the file
def read_file(filename):
    try:
        # Attempt to open the file
        with open(filename, 'r') as file:
            content = file.read()
        return content
    except FileNotFoundError:
        # Handle the case where the file doesn't exist
        print(f"Error: The file '{filename}' does not exist.")
        return None  # Return None or any other value to indicate the
failure

# Specify the file name
filename = '/content/drive/MyDrive/misbah.txt'

# Attempt to read the file
file_content = read_file(filename)

if file_content is not None:
    # If the file exists, process the content
    print(f"File content:\n{file_content}")
else:
    # If the file doesn't exist, handle accordingly (file not found)
    print("The file could not be read.")
```

```
File content:
python is awesome!
python is easy to learn
```

```
#invalid input handling
# Function to get an integer input from the user with error handling
def get_integer_input(prompt):
    while True:
        try:
            # Attempt to convert the input to an integer
            return int(input(prompt))
        except ValueError:
            # Handle the case where the input is not an integer
            print("Error: Please enter a valid integer.")

# Input two integers from the user
num1 = get_integer_input("Enter the first integer: ")
num2 = get_integer_input("Enter the second integer: ")

# Perform division and handle potential ZeroDivisionError
try:
    result = num1 / num2
    print(f"The result of dividing {num1} by {num2} is: {result}")
except ZeroDivisionError:
    print("Error: Cannot divide by zero!")
```

```
Enter the first integer: 2
Enter the second integer: 5
The result of dividing 2 by 5 is: 0.4
```

```
#handle key error
# Function to access a value from the dictionary safely
def access_dict_value(my_dict, key):
    try:
        # Attempt to access the value using the key
        return my_dict[key]
    except KeyError:
        # Handle the case where the key does not exist in the
dictionary
        print(f"Error: The key '{key}' does not exist in the
dictionary.")
        return None  # Return None or any default value as appropriate

# Sample dictionary
my_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}

# Try to access different keys in the dictionary
key_to_access = 'name'  # Valid key
result = access_dict_value(my_dict, key_to_access)
if result is not None:
    print(f"Value for '{key_to_access}': {result}")

key_to_access = 'occupation'  # Invalid key
result = access_dict_value(my_dict, key_to_access)
if result is not None:
    print(f"Value for '{key_to_access}': {result}")
```

```
Value for 'name': Alice
Error: The key 'occupation' does not exist in the dictionary.
```

```python
#file read permission
# Function to read the file with error handling
def read_file(filename):
    try:
        # Attempt to open and read the file
        with open(filename, 'r') as file:
            content = file.read()
        return content
    except PermissionError:
        # Handle the case where permission to read the file is denied
        print(f"Error: Permission denied to read the file 
'{filename}'.")
        return None  # Return None or any other value to indicate 
failure
    except FileNotFoundError:
        # Handle the case where the file doesn't exist
        print(f"Error: The file '{filename}' does not exist.")
        return None
    except Exception as e:
        # Handle any other unforeseen errors
        print(f"An unexpected error occurred: {e}")
        return None

# Specify the filename
filename = 'data.txt'

# Attempt to read the file
file_content = read_file(filename)

if file_content is not None:
    print(f"File content:\n{file_content}")
else:
    print("The file could not be read.")
```

```
Error: The file 'data.txt' does not exist.
The file could not be read.
```

```python
#catch multiple exceptions
# Function to read the file with error handling
def read_file(filename):
    try:
        # Attempt to open and read the file
        with open(filename, 'r') as file:
            content = file.read()
        return content
    except FileNotFoundError:
        # Handle the case where the file doesn't exist
        print(f"Error: The file '{filename}' does not exist.")
        return None  # Return None or any other value to indicate
failure
    except PermissionError:
        # Handle the case where permission to read the file is denied
        print(f"Error: Permission denied to read the file
'{filename}'.")
        return None  # Return None or any other value to indicate
failure
    except Exception as e:
        # Handle any other unforeseen errors
        print(f"An unexpected error occurred: {e}")
        return None

# Specify the filename
filename = 'data.txt'

# Attempt to read the file
file_content = read_file(filename)

if file_content is not None:
    print(f"File content:\n{file_content}")
else:
    print("The file could not be read.")
```

```
Error: The file 'data.txt' does not exist.
The file could not be read.
```

```
#custom exception
# Define a custom exception for negative numbers
class NegativeNumberError(Exception):
    def _init_(self, message="Negative numbers are not allowed"):
        self.message = message
        super()._init_(self.message)

# Function to get user input and check for negative numbers
def get_positive_number():
    try:
        # Get input from the user
        number = float(input("Enter a number: "))

        # Raise the custom exception if the number is negative
        if number < 0:
            raise NegativeNumberError(f"Error: {number} is a negative
number.")

        print(f"You entered a positive number: {number}")

    except NegativeNumberError as e:
        # Handle the custom exception
        print(e)
    except ValueError:
        # Handle the case where the input is not a valid number
        print("Error: Please enter a valid number.")

# Call the function to test
get_positive_number()
```

```
Enter a number: 4
You entered a positive number: 4.0
```

```
#handle inedx error
# Function to safely access a list element by index
def access_list_element(my_list, index):
    try:
        # Attempt to access the element at the specified index
        element = my_list[index]
        return element
    except IndexError:
        # Handle the case where the index is out of range
        print(f"Error: Index {index} is out of range for the list.")
        return None  # Return None or any other value to indicate
failure
    except Exception as e:
        # Handle any other unforeseen errors
        print(f"An unexpected error occurred: {e}")
        return None

# Example list
my_list = [10, 20, 30, 40, 50]

# Try accessing a valid index
index = 2
result = access_list_element(my_list, index)
if result is not None:
    print(f"Element at index {index}: {result}")

# Try accessing an invalid index (out of range)
index = 10
result = access_list_element(my_list, index)
if result is not None:
    print(f"Element at index {index}: {result}")
```

```
Element at index 2: 30
Error: Index 10 is out of range for the list.
```

```
#resource cleanup with finally
def read_file():
    file = None
    try:
        # Try to open and read the file
        file = open('data.txt', 'r')
        content = file.read()
        print("File content:\n", content)
    except FileNotFoundError:
        print("Error: The file 'data.txt' was not found.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
    finally:
        # Ensure the file is closed
        if file:
            file.close()
            print("File has been closed.")

# Call the function
read_file()
```

```
Error: The file 'data.txt' was not found.
```

```
#neted exception handling
def divide_numbers():
    try:
        # Try block for getting user input and performing operations
        num1 = input("Enter the first number: ")
        num2 = input("Enter the second number: ")

        try:
            # Convert input to integers (this might raise ValueError)
            num1 = int(num1)
            num2 = int(num2)

            # Attempt to divide (this might raise ZeroDivisionError)
            result = num1 / num2
            print(f"Result: {num1} / {num2} = {result}")

        except ValueError:
            # Handle the case where conversion to int fails
            print("Error: Please enter valid integers.")

        except ZeroDivisionError:
            # Handle the case where division by zero is attempted
            print("Error: Cannot divide by zero.")

    except Exception as e:
        # Handle any other general errors (e.g., unexpected input or
other errors)
        print(f"An unexpected error occurred: {e}")

# Run the function
divide_numbers()
```

```
Enter the first number: 34
Enter the second number: 56
Result: 34 / 56 = 0.6071428571428571
```