Double-click (or enter) to edit

1. Write a python function that copies a file reading and writing up to 50 characters at a time.

```
File=open('demo.txt','r')

def copy_file(input_file_path, output_file_path):
    try:
        with open(input_file_path, 'rb') as input_file, open(output_file_path, 'wb') as out
            while True:
                chunk = input_file.read(50)
                if not chunk:
                    break
                output_file.write(chunk)
        print("File copied successfully.")
    except FileNotFoundError:
        print("File not found.")
    except Exception as e:
        print("An error occurred:", str(e))

input_file_path = 'demo.txt'
output_file_path = 'output.txt'
copy_file(input_file_path, output_file_path)
```

```
File copied successfully.
```

2. Print all numbers present in the text file and print the number of blank spaces in that file.

```
def count_numbers_and_blank_spaces(file_path):
    try:
        with open(file_path, 'r') as file:
            text = file.read()


            number_count = 0
            for char in text:
                if char.isdigit():
                    number_count += 1
                elif char.isspace():
                    blank_space_count += 1

            print(f"Numbers found in the file: {number_count}")
            print(f"Blank spaces found in the file: {blank_space_count}")
    except FileNotFoundError:
        print("File not found.")
    except Exception as e:
        print("An error occurred:", str(e))

file_path = 'demo.txt'
count_numbers_and_blank_spaces(file_path)
```

```
Numbers found in the file: 140
Blank spaces found in the file: 276
```

3. Write a function called sed that takes as arguments a pattern string, a replacement string. and two filenames; it should read the first file and write the contents into the second file (creating it if necessary). If the pattern string appears anywhere in the file, it should be replaced with the replacement string. If an error occurs while opening, reading, writing, or closing files, your program should catch the exception, print an error message, and exit.

```
def sed(pattern, replacement, input_file, output_file):
    try:
        with open(input_file, 'r') as infile:
            file_content = infile.read()

        modified_content = file_content.replace(pattern, replacement)

        with open(output_file, 'w') as outfile:
            outfile.write(modified_content)

        print(f"File '{input_file}' processed successfully. Result written to '{output_fil

    except FileNotFoundError:
        print(f"Error: File '{input_file}' not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


pattern = "old_pattern"
replacement = "new_pattern"
input_file = "demo.txt"
output_file = "output.txt"

sed(pattern, replacement, input_file, output_file)
```

```
     File 'demo.txt' processed successfully. Result written to 'output.txt'.
```

4. Log File Analysis: You have a log file containing records of user activities on a website. Each line in the file represents a log entry with details like timestamp, user ID, and action performed. Your task is to analyze this log file.

a. Write Python code to read the log file and extract specific information, such as the number of unique users or the most common action.

b. How would you handle large log files efficiently without loading the entire file into memory?

```
def analyze_log_file(log_file_path):
    try:
        with open(log_file_path, 'r') as log_file:
            log_entries = log_file.readlines()

        unique_users = set()
        action_counts = {}

        for entry in log_entries:
            parts = entry.strip().split(',')
            if len(parts) >= 3:
                timestamp, user_id, action = parts[0], parts[1], parts[2]
                unique_users.add(user_id)

                if action in action_counts:
                    action_counts[action] += 1
                else:
                    action_counts[action] = 1

        print(f"Number of unique users: {len(unique_users)}")
        most_common_action = max(action_counts, key=action_counts.get)
        print(f"Most common action: {most_common_action} ({action_counts[most_common_actio

    except FileNotFoundError:
        print("Error: Log file not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


log_file_path = '/content/logfile.txt'
analyze_log_file(log_file_path)



def analyze_log_file(log_file_path):
    try:
        def log_entries_generator():
            with open(log_file_path, 'r') as log_file:
                for line in log_file:
                    yield line

        unique_users = set()
        action_counts = {}

        for entry in log_entries_generator():
            parts = entry.strip().split(',')
            if len(parts) >= 3:
                timestamp, user_id, action = parts[0], parts[1], parts[2]
                unique_users.add(user_id)

                if action in action_counts:
                    action_counts[action] += 1
                else:
                    action_counts[action] = 1

        print(f"Number of unique users: {len(unique_users)}")
        most_common_action = max(action_counts, key=action_counts.get)
        print(f"Most common action: {most_common_action} ({action_counts[most_common_actio

    except FileNotFoundError:
        print("Error: Log file not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


log_file_path = '/content/logfile.txt'
analyze_log_file(log_file_path)
```

```
Number of unique users: 1
Most common action:  branch=c5558820f367665758d4d719b7553a02ff4954e0 (3 times)
Number of unique users: 1
Most common action:  branch=c5558820f367665758d4d719b7553a02ff4954e0 (3 times)
```

5. Text File Search and Replace: You have a text file with a large amount of text, and you want to search for specific words or phrases and replace them with new content.

a. Write Python code to search for and replace text within a text file.

b. How would you handle cases where you need to perform multiple replacements in a single pass?

```
search_text = "by"

replace_text = "by (replaced)"

with open(r'Sample.txt', 'r') as file:
    data = file.read()
    data = data.replace(search_text, replace_text)

with open(r'Sample.txt', 'w') as file:
    file.write(data)
print("Text replaced")
```

[illegible]

```
test_str = "abbabba"
print("The original string is : " + str(test_str))
new_str = ''.join(['a' if char == 'b' else 'b' if char == 'a' else char for char in test_st

print("The string after replacement of positions : " +new_str)
```

```
Text replaced
The original string is : abbabba
The string after replacement of positions : baabaab
```

6. Write a Python script that concatenates the contents of multiple text files into a single output file. Allow the user to specify the input files and the output file.

```
file=open("file1.txt","x")
file=open("file2.txt","x")
file=open("file3.txt","x")
def concatenate_files(input_files, output_file):
    try:
        with open(output_file, 'w') as output:
            for input_file in input_files:
                with open(input_file, 'r') as file:
                    output.write(file.read())

        print(f"Concatenation complete. Output written to {output_file}")

    except FileNotFoundError:
        print("One or more input files not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")

input_files = ['file1.txt', 'file2.txt', 'file3.txt']
output_file = 'output.txt'

concatenate_files(input_files, output_file)
```

```
Concatenation complete. Output written to output.txt
```

7. You are given a text file named input.txt containing a list of words, one word per line. Your task is to create a Python program that reads the contents of input.txt, processes the

words, and writes the result to an output file named output.txt.

a. The program should perform the following operations:

Read the words from input.txt.

ii. For each word in the input file, calculate the length of the word and store.it

in a dictionary where the word is the key, and the length is the value. iii. Write the word-length dictionary to output.txt in the following format:

Word1: Length1

Word2: Length?

iv. Close both input and output files properly.

v. Write Python code to accomplish this task. Ensure proper error handling for file operations.

```
strings = []
for i in range(4):
    user_input = input(f"Enter string {i + 1}: ")
    strings.append(user_input)


file_path = "input.txt"

try:
    with open(file_path, 'w') as file:
        for string in strings:
            file.write(string + '\n')
        print(f"The strings have been written to '{file_path}' successfully.")


    with open(file_path, 'r') as file:

        for line in file:
            string_length = len(line.strip())
            print(f"Length of '{line.strip()}': {string_length}")

except Exception as e:
    print(f"An error occurred: {str(e)}")
```

```
Enter string 1: apple
Enter string 2: banana
Enter string 3: cherry
Enter string 4: date
The strings have been written to 'input.txt' successfully.
Length of 'apple': 5
Length of 'banana': 6
Length of 'cherry': 6
Length of 'date': 4
```

8. Assume that you are developing a student gradebook system for a school. The system should allow teachers to input student grades for various subjects, store the data in files, and provide students with the ability to view their grades.

Design a Python program that accomplishes the following tasks:

i. Teachers should be able to input grades for students in different subjects.

ii. Store the student grade data in separate text files for each subject.

iii. Students should be able to view their grades for each subject.

iv. Implement error handling for file operations, such as file not found or permission issues.

```
import os

def input_grades(subject):
    try:
        with open(f"{subject}.txt", "a") as file:
            while True:
                student_name = input("Enter student name (or 'done' to finish): ")
                if student_name.lower() == "done":
                    break
                grade = input(f"Enter {student_name}'s grade for {subject}: ")
                file.write(f"{student_name}: {grade}\n")
        print(f"Grades for {subject} have been recorded successfully.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")

def view_grades(subject):
    try:
        with open(f"{subject}.txt", "r") as file:
            print(f"Grades for {subject}:")
            for line in file:
                print(line.strip())
    except FileNotFoundError:
        print(f"Grades for {subject} not found.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")

while True:
    print("\nStudent Gradebook System")
    print("1. Input Grades")
    print("2. View Grades")
    print("3. Exit")

    choice = input("Select an option (1/2/3): ")

    if choice == "1":
        subject = input("Enter the subject name: ")
        input_grades(subject)
    elif choice == "2":
        subject = input("Enter the subject name: ")
        view_grades(subject)
    elif choice == "3":
        break
    else:
        print("Invalid choice. Please select 1, 2, or 3.")
```

```
Student Gradebook System
1. Input Grades
2. View Grades
3. Exit
Select an option (1/2/3): 3
```