perplexity

create a pdf containing all the programs in python and also write the headng of the program on the top of program in comment

Okay, I will help you create a PDF containing the Python programs based on the images you sent. I'll extract the code from the images, add headers as comments to each program, and then provide you with the Python code to generate a PDF containing all of them.

Here's the approach:

1. **Extract Code:** I'll manually extract the Python code from the images.
2. **Add Headers:** I'll add comment headers to each program, indicating what the program does.
3. **Create PDF Generation Code:** I'll provide you with a Python script that uses the `reportlab` library to create a PDF and add each program to it.

Here are the programs:

```
# 1. Reverse File Content Using File Pointer Manipulation

def reverse_file_content(input_file, reversed_file):
    """
    Reads a text file, reverses its content, and writes it to a new file.
    Also, displays the last 10 characters from the reversed file.
    """
    try:
        with open(input_file, 'r') as file:
            content = file.read()

        reversed_content = content[::-1]

        with open(reversed_file, 'w') as file:
            file.write(reversed_content)

        with open(reversed_file, 'r') as file:
            file.seek(0, 2)  # Go to the end of the file
            file_size = file.tell()
            start_position = max(file_size - 10, 0)  # Ensure start position is not negat
            file.seek(start_position, 0)  # Go to the start position
            last_chars = file.read()

        print(f"Last 10 characters of reversed file: {last_chars}")

    except FileNotFoundError:
        print(f"Error: File '{input_file}' not found.")
```

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

# Example usage:
# reverse_file_content('input.txt', 'reversed_output.txt')


# 2. Merging Multiple Files into One

def merge_files(input_files, output_file):
    """
    Takes multiple text files as input and merges their content into a single output file
    Ensures that each file's content starts on a new line in the merged file.
    """
    try:
        with open(output_file, 'w') as outfile:
            for file_name in input_files:
                try:
                    with open(file_name, 'r') as infile:
                        content = infile.read()
                        outfile.write(content + '\n')  # Add content and a newline
                except FileNotFoundError:
                    print(f"Error: File '{file_name}' not found.")
                except Exception as e:
                    print(f"An error occurred while reading '{file_name}': {e}")
        print(f"Successfully merged files into '{output_file}'")

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

# Example usage:
# merge_files(['file1.txt', 'file2.txt', 'file3.txt'], 'merged_output.txt')


# 3. Find and Replace Words in a Large File Efficiently
def find_and_replace(input_file, output_file, old_word, new_word):
    """
    Reads a large text file line by line, replaces all occurrences of a specific word wit
    and writes the modified content to a new file. Uses readline() to optimize memory usa
    """
    try:
        with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
            for line in infile:
                modified_line = line.replace(old_word, new_word)
                outfile.write(modified_line)
        print(f"Successfully replaced '{old_word}' with '{new_word}' in '{input_file}' ar

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

# Example usage:
# find_and_replace('large_data.txt', 'modified_data.txt', 'AI', 'Artificial Intelligence'
```

```
# 4. Read Specific Sections from a Large File Using seek() and tell()
def extract_text_section(input_file, start_position, end_position):
    """Reads a large text file and extracts text between specified start and end position
    try:
        with open(input_file, 'r') as file:
            file.seek(start_position)
            extracted_content = file.read(end_position - start_position)
            return extracted_content
    except FileNotFoundError:
        return f"Error: File '{input_file}' not found."
    except Exception as e:
        return f"An error occurred: {e}"

# Example usage:
# extracted_text = extract_text_section('large_data.txt', 10, 50)
# print(extracted_text)


# 5. Writing Multiple Lines to a File
def write_multiple_lines(output_file):
    """
    Asks the user to enter lines until they type "STOP" and writes them to a file.
    Uses writelines() to write all the lines at once.
    """
    lines = []
    print("Enter text (type 'STOP' to finish):")
    while True:
        line = input()
        if line.upper() == "STOP":
            break
        lines.append(line + '\n')  # Add a newline character to each line

    try:
        with open(output_file, 'w') as file:
            file.writelines(lines)
        print(f"Contents written to '{output_file}' successfully.")
    except Exception as e:
        print(f"An error occurred: {e}")

# Example usage:
# write_multiple_lines('output.txt')
```

Here's the Python code to generate the PDF:

```
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from reportlab.lib import colors

def add_program_to_pdf(canvas, title, code, x, y, width, height):
    """Adds a program with a title and code to the PDF."""
    canvas.setFillColor(colors.black)
    canvas.setFont("Helvetica", 12)
    canvas.drawString(x, y, title)
    y -= 18  # Space after title
```

```
    canvas.setFont("Courier", 10) # Use Courier for code
    canvas.setFillColor(colors.blue) # set color for code

    # Split the code into lines and draw each line
    code_lines = code.splitlines()
    for line in code_lines:
        if y < 50:  # Check if we're near the bottom of the page
            canvas.showPage()  # Start a new page
            y = letter[^1] - 50  # Reset y position
            canvas.setFont("Courier", 10)
            canvas.setFillColor(colors.blue)

        canvas.drawString(x, y, line)
        y -= 12  # Adjust space between lines

    return y  # Return the updated Y position

def create_pdf(filename, programs):
    """Creates a PDF file and adds the given programs to it."""
    c = canvas.Canvas(filename, pagesize=letter)
    x = 50
    y = letter[^1] - 50  # Start from top with a margin
    width = letter[^0] - 2 * x
    height = 200

    for title, code in programs.items():
        y = add_program_to_pdf(c, title, code, x, y, width, height)
        if y < 50:  # Start a new page if necessary
            c.showPage()
            y = letter[^1] - 50

    c.save()
    print(f"PDF '{filename}' created successfully.")

if __name__ == '__main__':
    # Define your programs here
    programs = {
        "1. Reverse File Content Using File Pointer Manipulation": """def reverse_file_co
    \"\"\"
    Reads a text file, reverses its content, and writes it to a new file.
    Also, displays the last 10 characters from the reversed file.
    \"\"\"
    try:
        with open(input_file, 'r') as file:
            content = file.read()

        reversed_content = content[::-1]

        with open(reversed_file, 'w') as file:
            file.write(reversed_content)

        with open(reversed_file, 'r') as file:
            file.seek(0, 2)  # Go to the end of the file
            file_size = file.tell()
            start_position = max(file_size - 10, 0)  # Ensure start position is not negat
            file.seek(start_position, 0)  # Go to the start position
```

```
            last_chars = file.read()

        print(f"Last 10 characters of reversed file: {last_chars}")

    except FileNotFoundError:
        print(f"Error: File '{input_file}' not found.")
    except Exception as e:
        print(f"An error occurred: {e}")""",
        "2. Merging Multiple Files into One": """def merge_files(input_files, output_file
    \"\"\"
    Takes multiple text files as input and merges their content into a single output file
    Ensures that each file's content starts on a new line in the merged file.
    \"\"\"
    try:
        with open(output_file, 'w') as outfile:
            for file_name in input_files:
                try:
                    with open(file_name, 'r') as infile:
                        content = infile.read()
                        outfile.write(content + '\\n')  # Add content and a newline
                except FileNotFoundError:
                    print(f"Error: File '{file_name}' not found.")
                except Exception as e:
                    print(f"An error occurred while reading '{file_name}': {e}")
        print(f"Successfully merged files into '{output_file}'")

    except Exception as e:
        print(f"An error occurred: {e}")""",
        "3. Find and Replace Words in a Large File Efficiently": """def find_and_replace(
    \"\"\"
    Reads a large text file line by line, replaces all occurrences of a specific word wit
    and writes the modified content to a new file. Uses readline() to optimize memory usa
    \"\"\"
    try:
        with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
            for line in infile:
                modified_line = line.replace(old_word, new_word)
                outfile.write(modified_line)
        print(f"Successfully replaced '{old_word}' with '{new_word}' in '{input_file}' ar

    except FileNotFoundError:
        print(f"Error: File '{input_file}' not found.")
    except Exception as e:
        print(f"An error occurred: {e}")""",
        "4. Read Specific Sections from a Large File Using seek() and tell()": """def ext
    \"\"\"Reads a large text file and extracts text between specified start and end posit
    try:
        with open(input_file, 'r') as file:
            file.seek(start_position)
            extracted_content = file.read(end_position - start_position)
            return extracted_content
    except FileNotFoundError:
        return f"Error: File '{input_file}' not found."
    except Exception as e:
        return f"An error occurred: {e}""" ,
        "5. Writing Multiple Lines to a File": """def write_multiple_lines(output_file):
```

```
    \"\"\"
    Asks the user to enter lines until they type "STOP" and writes them to a file.
    Uses writelines() to write all the lines at once.
    \"\"\"
    lines = []
    print("Enter text (type 'STOP' to finish):")
    while True:
        line = input()
        if line.upper() == "STOP":
            break
        lines.append(line + '\\n')  # Add a newline character to each line

    try:
        with open(output_file, 'w') as file:
            file.writelines(lines)
        print(f"Contents written to '{output_file}' successfully.")
    except Exception as e:
        print(f"An error occurred: {e}")"""
    }

    create_pdf("file_handling_programs.pdf", programs)
```

Explanation:

1. **Install `reportlab`:** If you don't have it, install it using `pip install reportlab`.
2. `add_program_to_pdf` **Function:**
    - Takes the canvas, title, and code as input.
    - Sets the font and color for the title and code.
    - Splits the code into lines and adds each line to the PDF.
    - Handles page breaks if the content exceeds the page height.
3. `create_pdf` **Function:**
    - Creates a `canvas` object.
    - Iterates through the `programs` dictionary (which contains the title and code for each program).
    - Calls `add_program_to_pdf` to add each program to the PDF.
    - Saves the PDF.
4. `if __name__ == '__main__'`: **Block:**
    - This is where you define your programs (the `programs` dictionary). I've included the code from the images you sent.
    - Calls `create_pdf` to generate the PDF file.

How to use:

1. **Save the Code:** Save the above code as a `.py` file (e.g., `create_pdf.py`).
2. **Run the Code:** Execute the script from your terminal: `python create_pdf.py`

3. **Find the PDF:** A PDF file named `file_handling_programs.pdf` will be created in the same directory as the script.

This PDF will contain all the Python programs you provided, with each program's title at the top as a comment. Let me know if you have any questions.

⁂