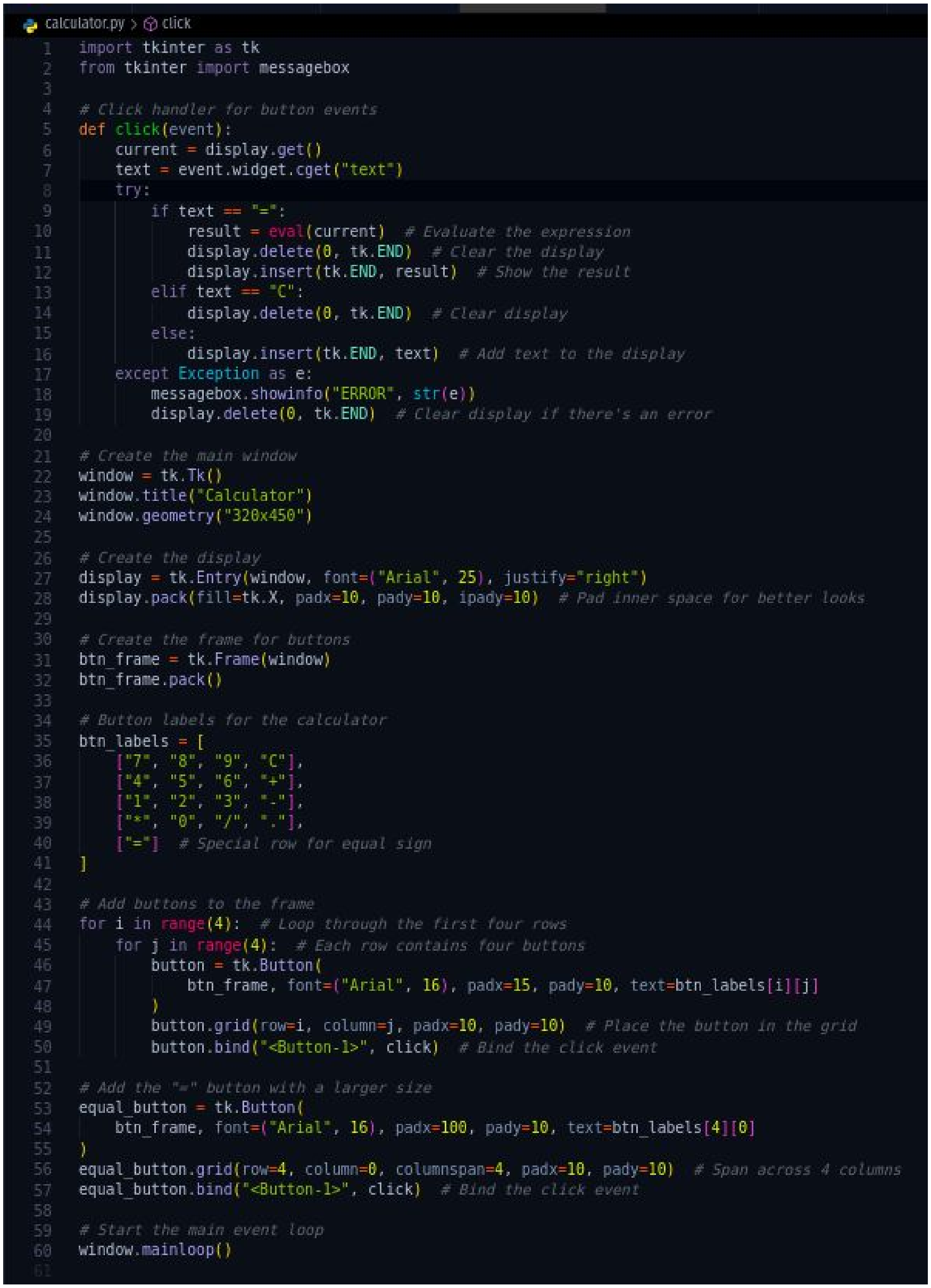
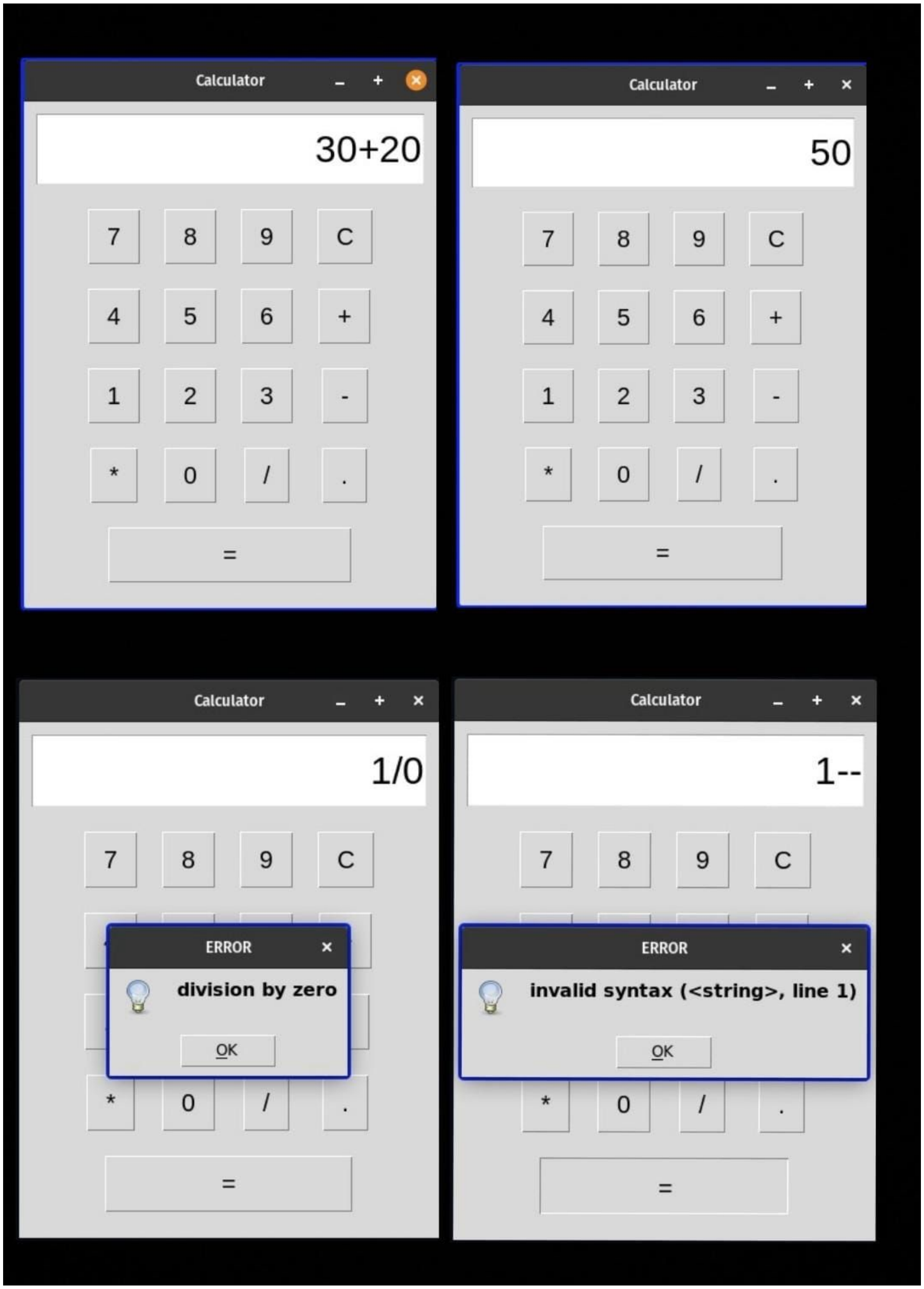
Question No:1

program



```python
import tkinter as tk
from tkinter import messagebox

# Click handler for button events
def click(event):
    current = display.get()
    text = event.widget.cget("text")
    try:
        if text == "=":
            result = eval(current)  # Evaluate the expression
            display.delete(0, tk.END)  # Clear the display
            display.insert(tk.END, result)  # Show the result
        elif text == "C":
            display.delete(0, tk.END)  # Clear display
        else:
            display.insert(tk.END, text)  # Add text to the display
    except Exception as e:
        messagebox.showinfo("ERROR", str(e))
        display.delete(0, tk.END)  # Clear display if there's an error

# Create the main window
window = tk.Tk()
window.title("Calculator")
window.geometry("320x450")

# Create the display
display = tk.Entry(window, font=("Arial", 25), justify="right")
display.pack(fill=tk.X, padx=10, pady=10, ipady=10)  # Pad inner space for better looks

# Create the frame for buttons
btn_frame = tk.Frame(window)
btn_frame.pack()

# Button labels for the calculator
btn_labels = [
    ["7", "8", "9", "C"],
    ["4", "5", "6", "+"],
    ["1", "2", "3", "-"],
    ["*", "0", "/", "."],
    ["="]  # Special row for equal sign
]

# Add buttons to the frame
for i in range(4):  # Loop through the first four rows
    for j in range(4):  # Each row contains four buttons
        button = tk.Button(
            btn_frame, font=("Arial", 16), padx=15, pady=10, text=btn_labels[i][j]
        )
        button.grid(row=i, column=j, padx=10, pady=10)  # Place the button in the grid
        button.bind("<Button-1>", click)  # Bind the click event

# Add the "=" button with a larger size
equal_button = tk.Button(
    btn_frame, font=("Arial", 16), padx=100, pady=10, text=btn_labels[4][0]
)
equal_button.grid(row=4, column=0, columnspan=4, padx=10, pady=10)  # Span across 4 columns
equal_button.bind("<Button-1>", click)  # Bind the click event

# Start the main event loop
window.mainloop()
```

Output:



```
30+20
```

```
50
```

```
1/0
```

ERROR

division by zero

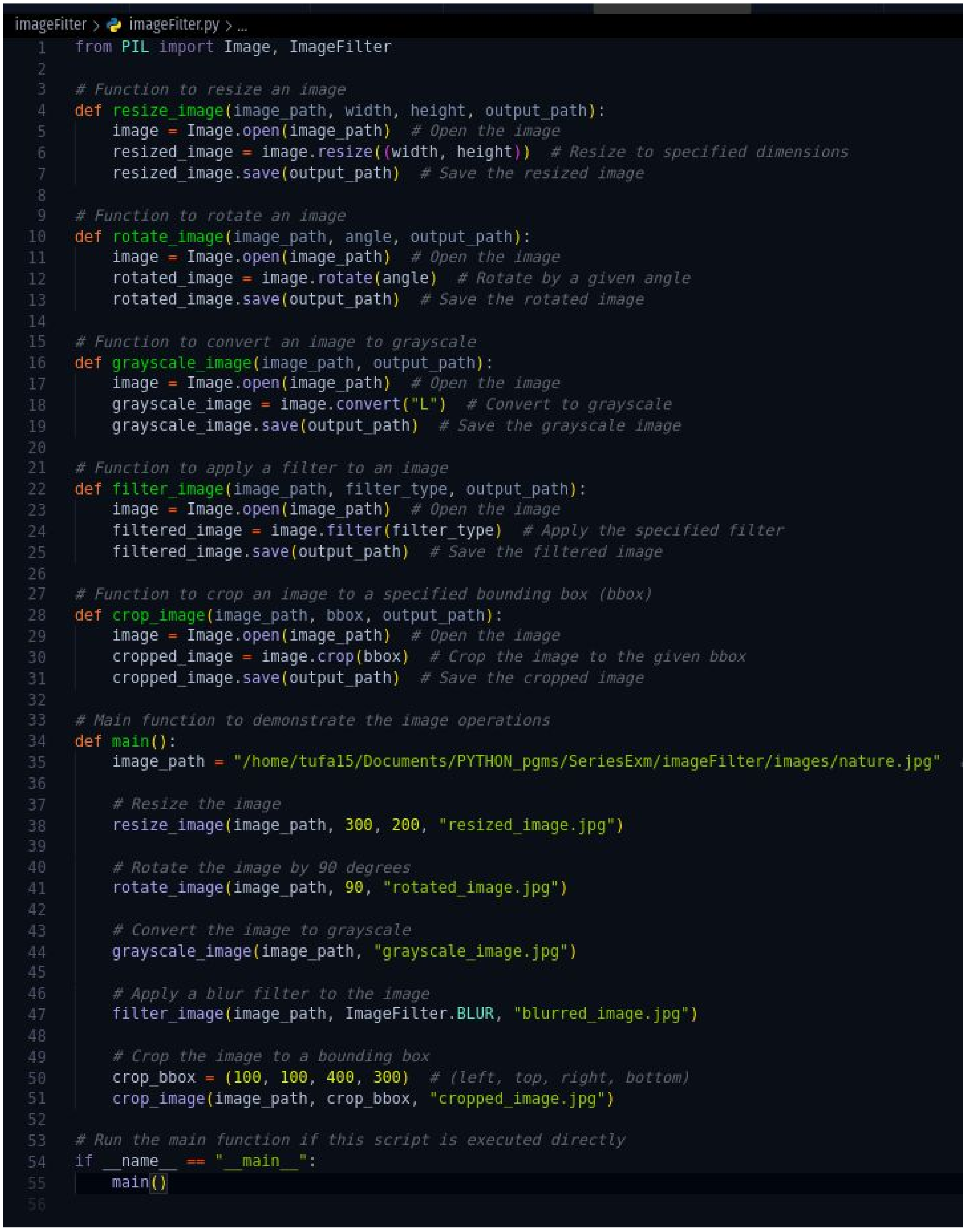
```
1--
```

ERROR

invalid syntax (<string>, line 1)

Question No:2

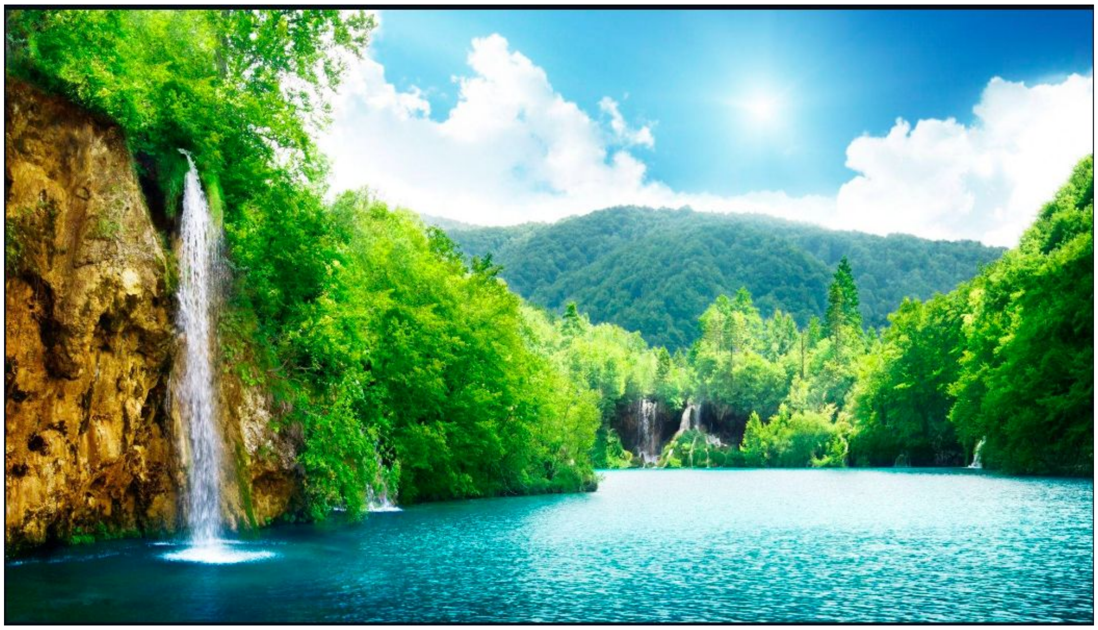
program



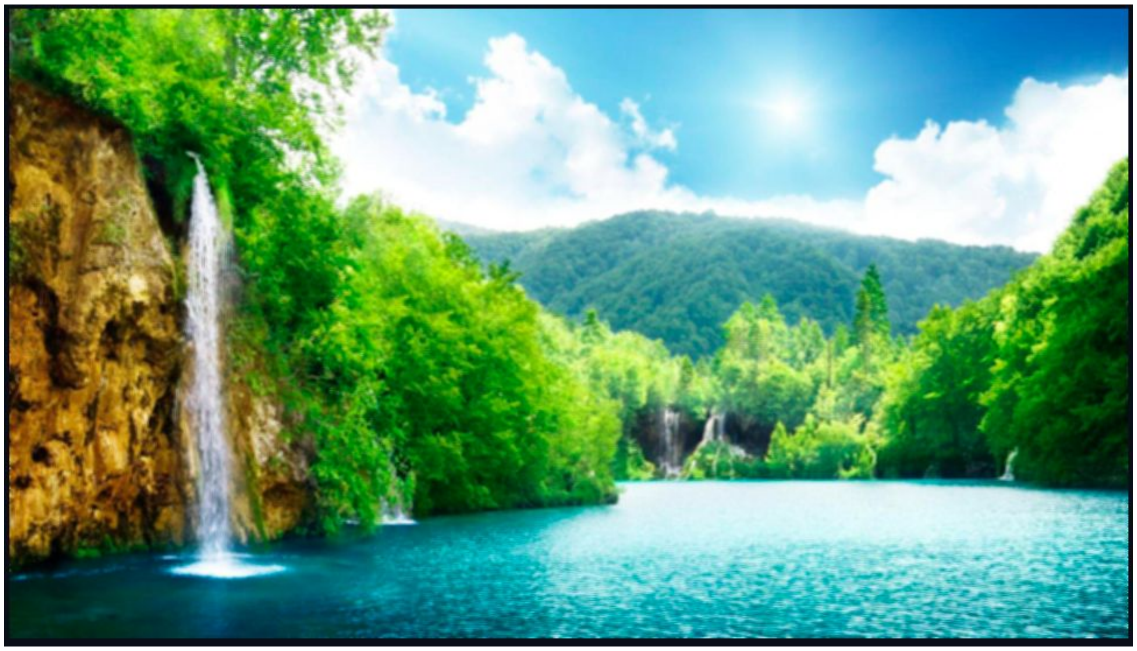
```python
from PIL import Image, ImageFilter

# Function to resize an image
def resize_image(image_path, width, height, output_path):
    image = Image.open(image_path)  # Open the image
    resized_image = image.resize((width, height))  # Resize to specified dimensions
    resized_image.save(output_path)  # Save the resized image

# Function to rotate an image
def rotate_image(image_path, angle, output_path):
    image = Image.open(image_path)  # Open the image
    rotated_image = image.rotate(angle)  # Rotate by a given angle
    rotated_image.save(output_path)  # Save the rotated image

# Function to convert an image to grayscale
def grayscale_image(image_path, output_path):
    image = Image.open(image_path)  # Open the image
    grayscale_image = image.convert("L")  # Convert to grayscale
    grayscale_image.save(output_path)  # Save the grayscale image

# Function to apply a filter to an image
def filter_image(image_path, filter_type, output_path):
    image = Image.open(image_path)  # Open the image
    filtered_image = image.filter(filter_type)  # Apply the specified filter
    filtered_image.save(output_path)  # Save the filtered image

# Function to crop an image to a specified bounding box (bbox)
def crop_image(image_path, bbox, output_path):
    image = Image.open(image_path)  # Open the image
    cropped_image = image.crop(bbox)  # Crop the image to the given bbox
    cropped_image.save(output_path)  # Save the cropped image

# Main function to demonstrate the image operations
def main():
    image_path = "/home/tufal5/Documents/PYTHON_pgms/SeriesExm/imageFilter/images/nature.jpg"

    # Resize the image
    resize_image(image_path, 300, 200, "resized_image.jpg")

    # Rotate the image by 90 degrees
    rotate_image(image_path, 90, "rotated_image.jpg")

    # Convert the image to grayscale
    grayscale_image(image_path, "grayscale_image.jpg")

    # Apply a blur filter to the image
    filter_image(image_path, ImageFilter.BLUR, "blurred_image.jpg")

    # Crop the image to a bounding box
    crop_bbox = (100, 100, 400, 300)  # (left, top, right, bottom)
    crop_image(image_path, crop_bbox, "cropped_image.jpg")

# Run the main function if this script is executed directly
if __name__ == "__main__":
    main()
```

Output:

Input Image

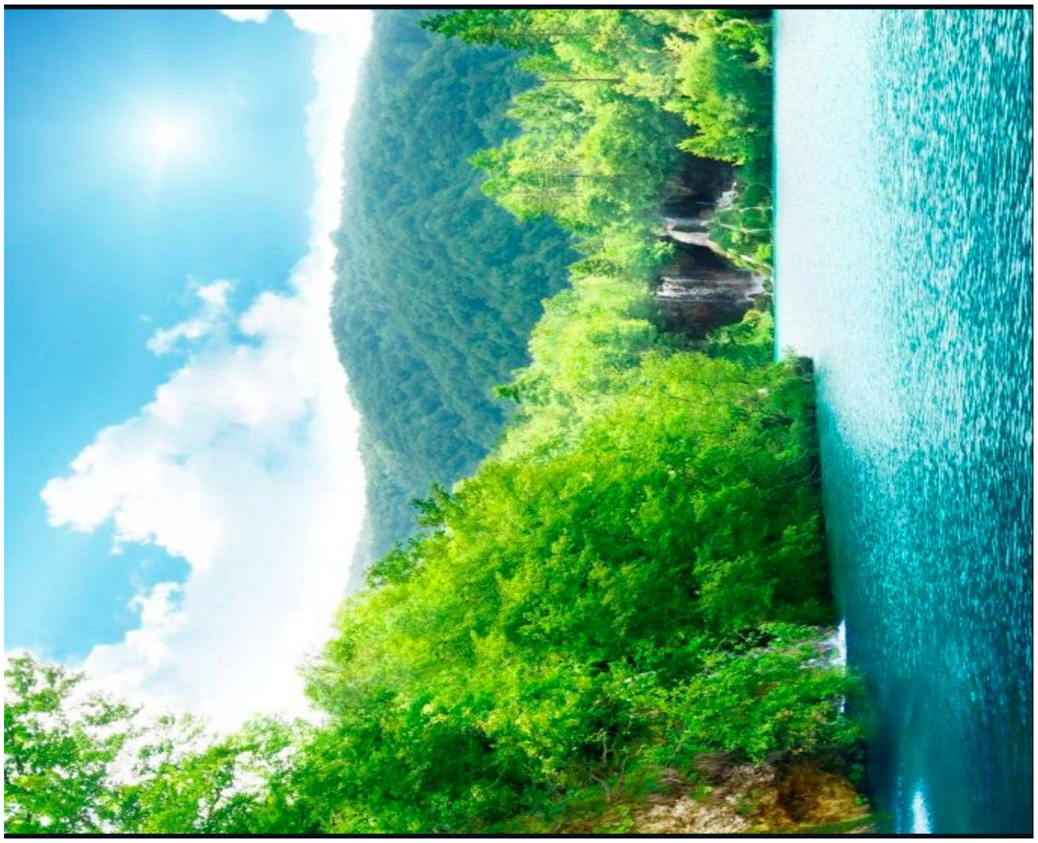


Blur Image



Grayscale Image

Rotate Image

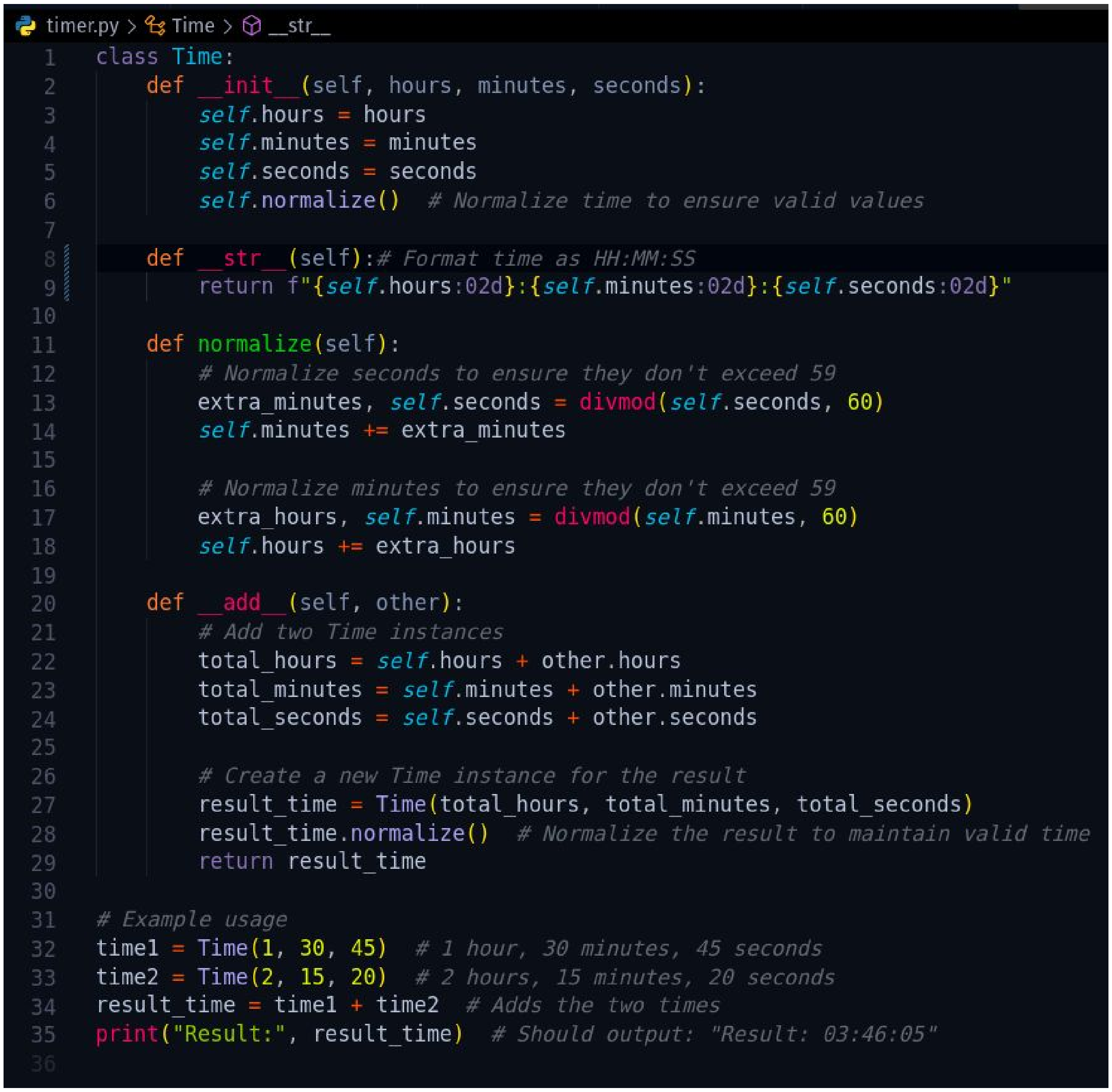


Resized Image:

Croped Image:

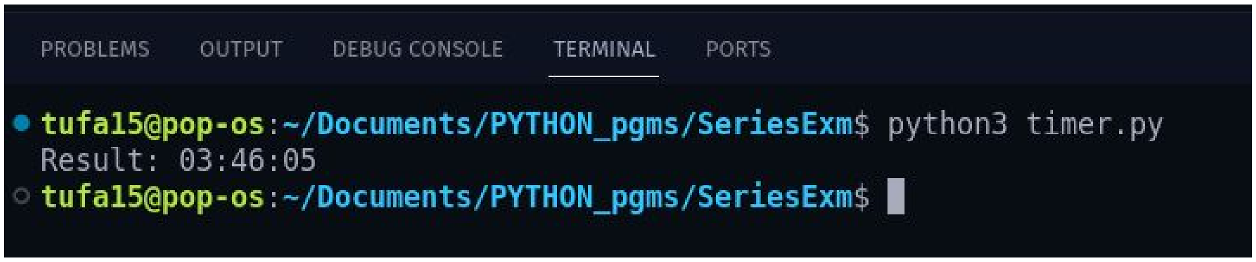
Question No:3

program



```
timer.py > Time > __str__
class Time:
    def __init__(self, hours, minutes, seconds):
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds
        self.normalize()   # Normalize time to ensure valid values

    def __str__(self):# Format time as HH:MM:SS
        return f"{self.hours:02d}:{self.minutes:02d}:{self.seconds:02d}"

    def normalize(self):
        # Normalize seconds to ensure they don't exceed 59
        extra_minutes, self.seconds = divmod(self.seconds, 60)
        self.minutes += extra_minutes

        # Normalize minutes to ensure they don't exceed 59
        extra_hours, self.minutes = divmod(self.minutes, 60)
        self.hours += extra_hours

    def __add__(self, other):
        # Add two Time instances
        total_hours = self.hours + other.hours
        total_minutes = self.minutes + other.minutes
        total_seconds = self.seconds + other.seconds

        # Create a new Time instance for the result
        result_time = Time(total_hours, total_minutes, total_seconds)
        result_time.normalize()   # Normalize the result to maintain valid time
        return result_time

# Example usage
time1 = Time(1, 30, 45)  # 1 hour, 30 minutes, 45 seconds
time2 = Time(2, 15, 20)  # 2 hours, 15 minutes, 20 seconds
result_time = time1 + time2   # Adds the two times
print("Result:", result_time)   # Should output: "Result: 03:46:05"
```

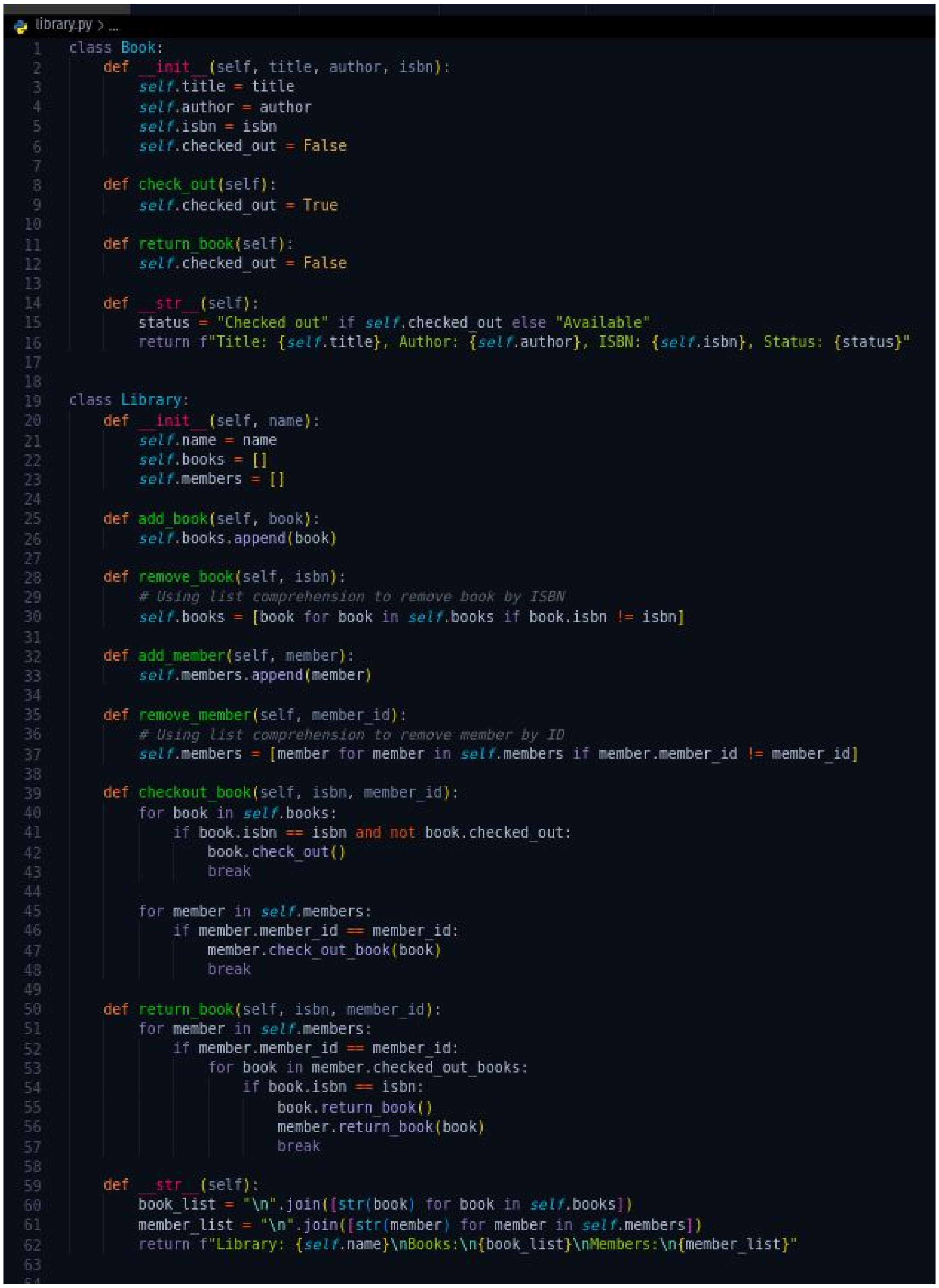
Output:



```
tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm$ python3 timer.py
Result: 03:46:05
tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm$ 
```

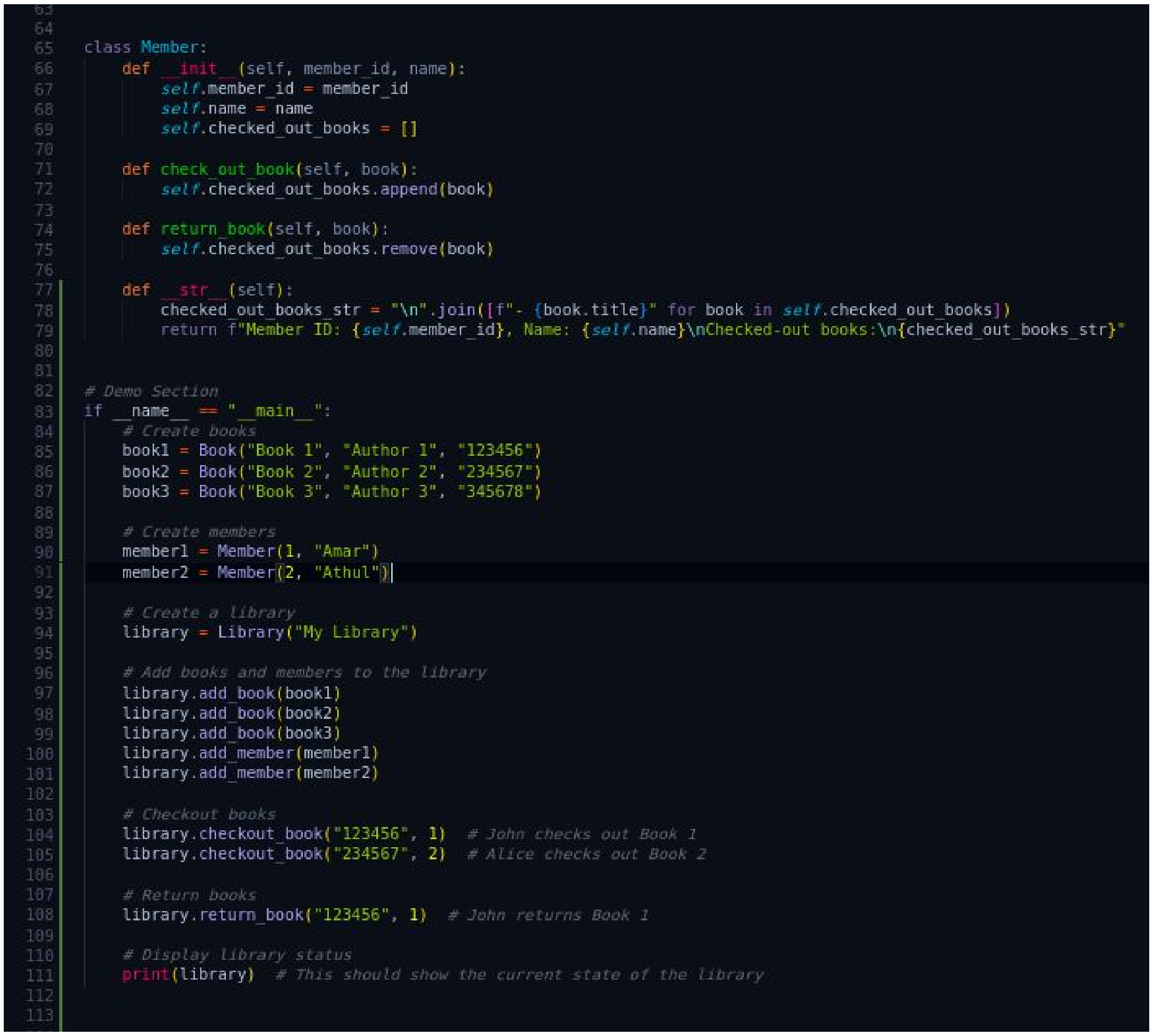
Question No:4

program



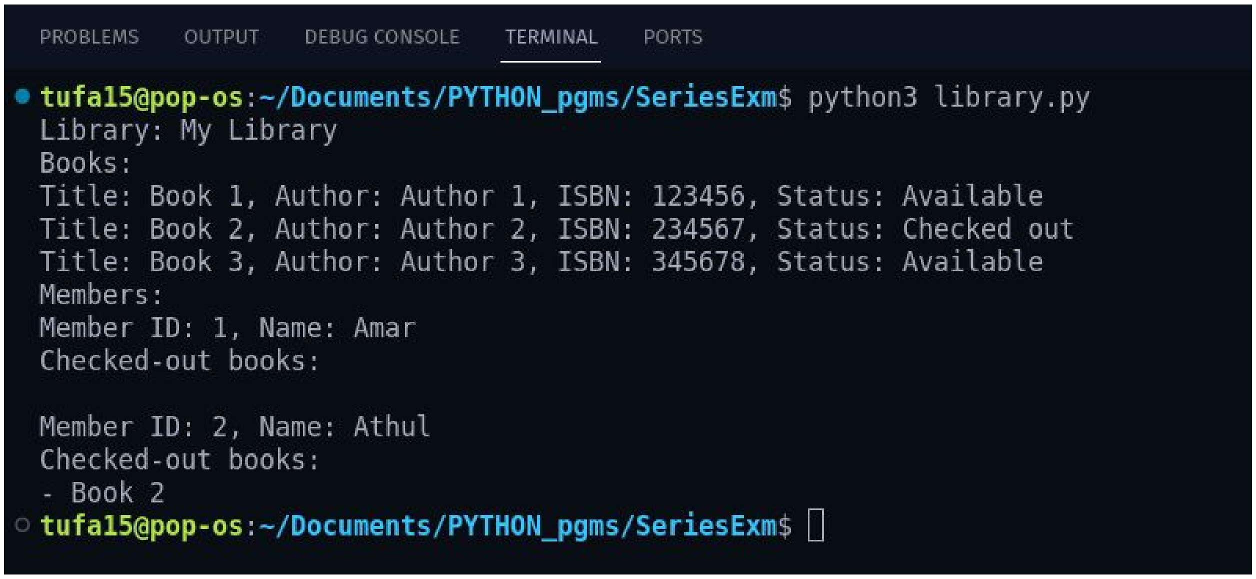
library.py > ...

```python
class Book:
    def __init__(self, title, author, isbn):
        self.title = title
        self.author = author
        self.isbn = isbn
        self.checked_out = False

    def check_out(self):
        self.checked_out = True

    def return_book(self):
        self.checked_out = False

    def __str__(self):
        status = "Checked out" if self.checked_out else "Available"
        return f"Title: {self.title}, Author: {self.author}, ISBN: {self.isbn}, Status: {status}"


class Library:
    def __init__(self, name):
        self.name = name
        self.books = []
        self.members = []

    def add_book(self, book):
        self.books.append(book)

    def remove_book(self, isbn):
        # Using list comprehension to remove book by ISBN
        self.books = [book for book in self.books if book.isbn != isbn]

    def add_member(self, member):
        self.members.append(member)

    def remove_member(self, member_id):
        # Using list comprehension to remove member by ID
        self.members = [member for member in self.members if member.member_id != member_id]

    def checkout_book(self, isbn, member_id):
        for book in self.books:
            if book.isbn == isbn and not book.checked_out:
                book.check_out()
                break

        for member in self.members:
            if member.member_id == member_id:
                member.check_out_book(book)
                break

    def return_book(self, isbn, member_id):
        for member in self.members:
            if member.member_id == member_id:
                for book in member.checked_out_books:
                    if book.isbn == isbn:
                        book.return_book()
                        member.return_book(book)
                        break

    def __str__(self):
        book_list = "\n".join([str(book) for book in self.books])
        member_list = "\n".join([str(member) for member in self.members])
        return f"Library: {self.name}\nBooks:\n{book_list}\nMembers:\n{member_list}"

```



```python

class Member:
    def __init__(self, member_id, name):
        self.member_id = member_id
        self.name = name
        self.checked_out_books = []

    def check_out_book(self, book):
        self.checked_out_books.append(book)

    def return_book(self, book):
        self.checked_out_books.remove(book)

    def __str__(self):
        checked_out_books_str = "\n".join([f"- {book.title}" for book in self.checked_out_books])
        return f"Member ID: {self.member_id}, Name: {self.name}\nChecked-out books:\n{checked_out_books_str}"


# Demo Section
if __name__ == "__main__":
    # Create books
    book1 = Book("Book 1", "Author 1", "123456")
    book2 = Book("Book 2", "Author 2", "234567")
    book3 = Book("Book 3", "Author 3", "345678")

    # Create members
    member1 = Member(1, "Amar")
    member2 = Member(2, "Athul")

    # Create a library
    library = Library("My Library")

    # Add books and members to the library
    library.add_book(book1)
    library.add_book(book2)
    library.add_book(book3)
    library.add_member(member1)
    library.add_member(member2)

    # Checkout books
    library.checkout_book("123456", 1)  # John checks out Book 1
    library.checkout_book("234567", 2)  # Alice checks out Book 2

    # Return books
    library.return_book("123456", 1)  # John returns Book 1

    # Display library status
    print(library)  # This should show the current state of the library

```

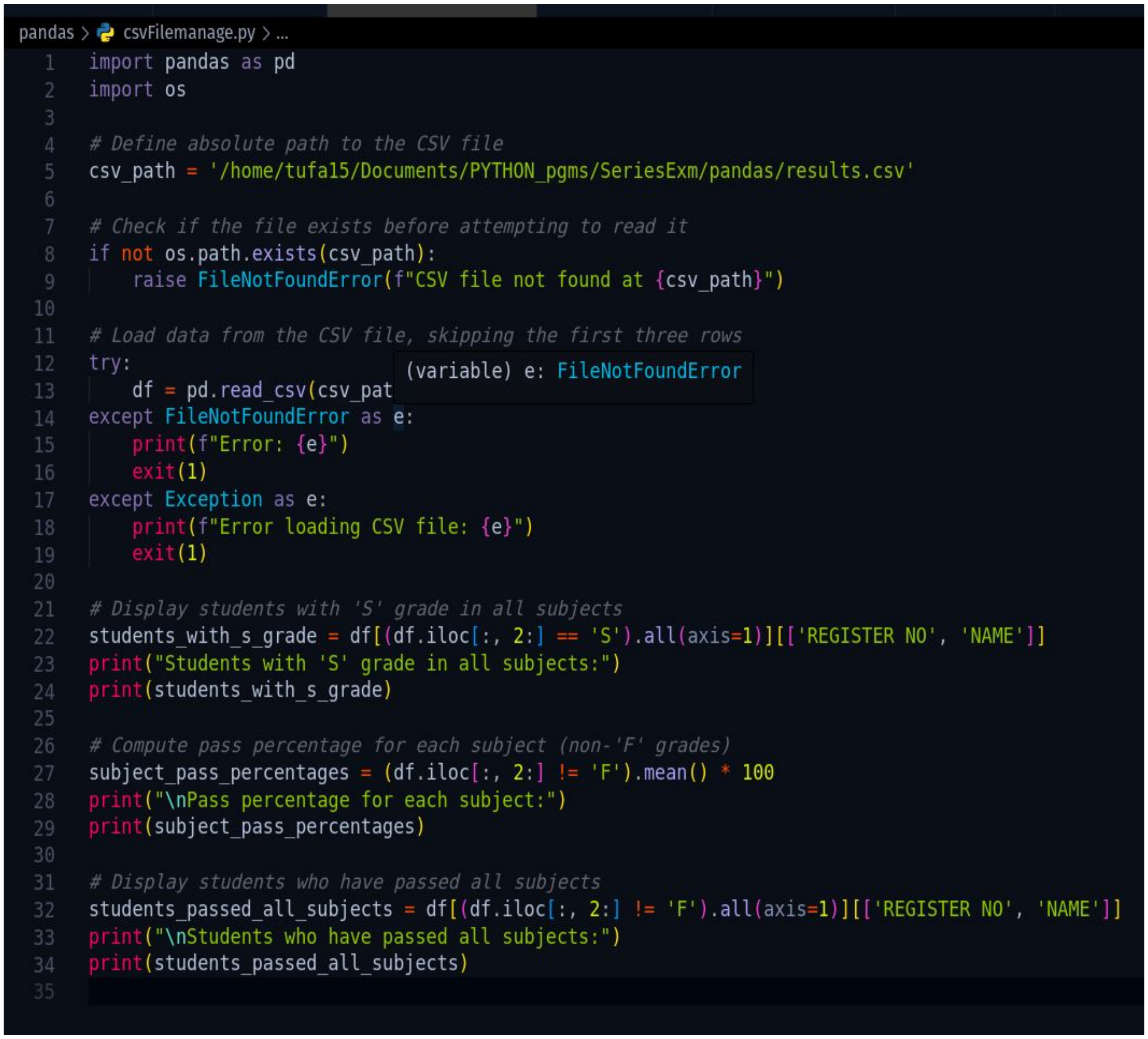
Output:



```
tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm$ python3 library.py
Library: My Library
Books:
Title: Book 1, Author: Author 1, ISBN: 123456, Status: Available
Title: Book 2, Author: Author 2, ISBN: 234567, Status: Checked out
Title: Book 3, Author: Author 3, ISBN: 345678, Status: Available
Members:
Member ID: 1, Name: Amar
Checked-out books:

Member ID: 2, Name: Athul
Checked-out books:
- Book 2
tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm$ 
```

Question No:5

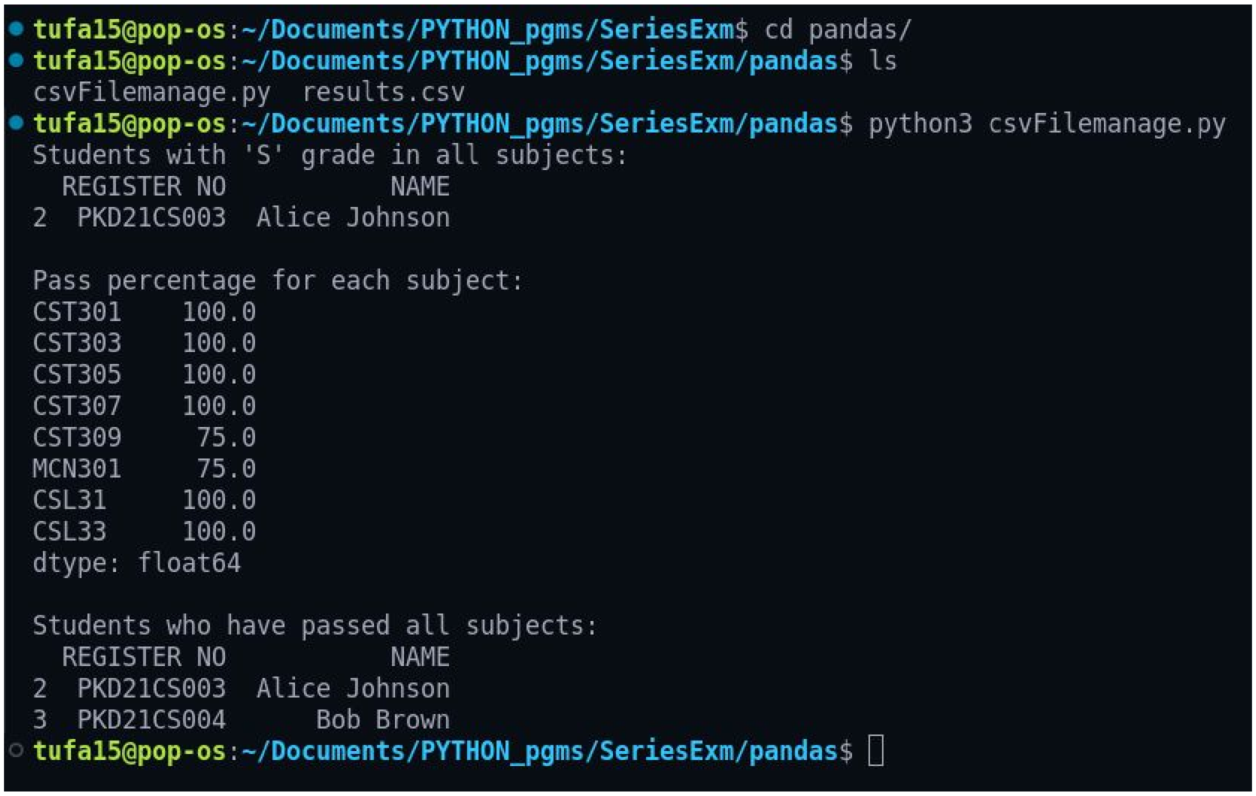
program



pandas > csvFilemanage.py > ...

```python
import pandas as pd
import os

# Define absolute path to the CSV file
csv_path = '/home/tufa15/Documents/PYTHON_pgms/SeriesExm/pandas/results.csv'

# Check if the file exists before attempting to read it
if not os.path.exists(csv_path):
    raise FileNotFoundError(f"CSV file not found at {csv_path}")

# Load data from the CSV file, skipping the first three rows
try:
    df = pd.read_csv(csv_pat  (variable) e: FileNotFoundError
except FileNotFoundError as e:
    print(f"Error: {e}")
    exit(1)
except Exception as e:
    print(f"Error loading CSV file: {e}")
    exit(1)

# Display students with 'S' grade in all subjects
students_with_s_grade = df[(df.iloc[:, 2:] == 'S').all(axis=1)][['REGISTER NO', 'NAME']]
print("Students with 'S' grade in all subjects:")
print(students_with_s_grade)

# Compute pass percentage for each subject (non-'F' grades)
subject_pass_percentages = (df.iloc[:, 2:] != 'F').mean() * 100
print("\nPass percentage for each subject:")
print(subject_pass_percentages)

# Display students who have passed all subjects
students_passed_all_subjects = df[(df.iloc[:, 2:] != 'F').all(axis=1)][['REGISTER NO', 'NAME']]
print("\nStudents who have passed all subjects:")
print(students_passed_all_subjects)

```

Output:



```
● tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm$ cd pandas/
● tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm/pandas$ ls
  csvFilemanage.py  results.csv
● tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm/pandas$ python3 csvFilemanage.py
  Students with 'S' grade in all subjects:
    REGISTER NO           NAME
  2  PKD21CS003  Alice Johnson

  Pass percentage for each subject:
  CST301    100.0
  CST303    100.0
  CST305    100.0
  CST307    100.0
  CST309     75.0
  MCN301     75.0
  CSL31     100.0
  CSL33     100.0
  dtype: float64

  Students who have passed all subjects:
    REGISTER NO           NAME
  2  PKD21CS003  Alice Johnson
  3  PKD21CS004      Bob Brown
○ tufa15@pop-os:~/Documents/PYTHON_pgms/SeriesExm/pandas$ 
```