Objective: To identify and fix errors in a Python program that manipulates strings.

Code 1:  
def reverse\_string(s):

    reversed = ""

    for i in range(len(s) - 1, -1, -1):

        reversed += s[i]

    return reversed

def main():

    input\_string = "Hello, world!"

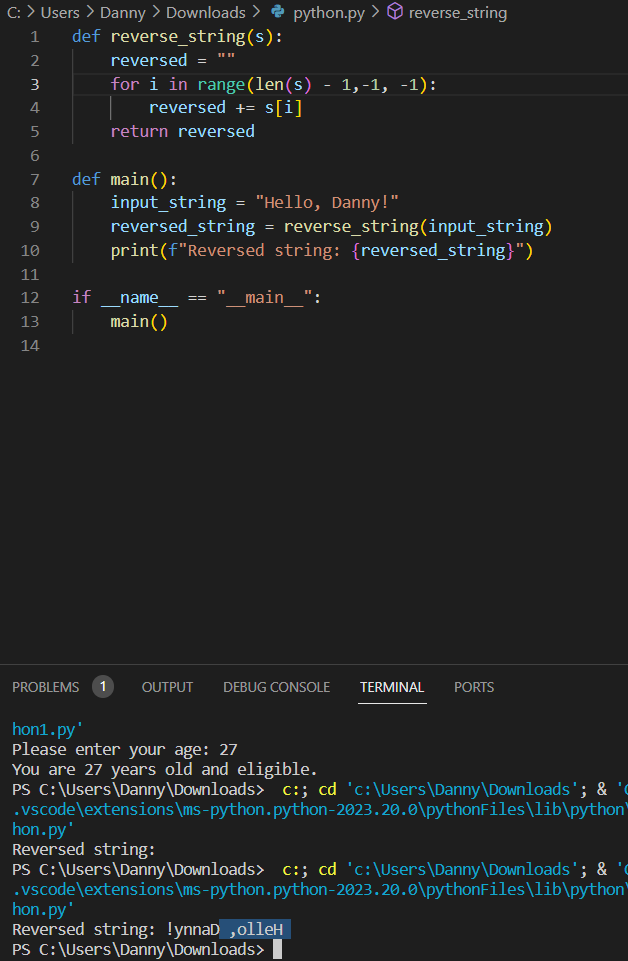
    reversed\_string = reverse\_string(input\_string)

    print(f"Reversed string: {reversed\_string}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

SOLUTION : Code works properly.



Code2:

Objective: To identify and fix errors in a Python program that validates user input.  
  
def get\_age():

    age = input("Please enter your age: ")

    if age.isnumeric() and age >= 18:

        return int(age)

    else:

        return None

def main():

    age = get\_age()

    if age:

        print(f"You are {age} years old and eligible.")

    else:

        print("Invalid input. You must be at least 18 years old.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

SOLUTION: The problem is that the default data type of input is string

This causes problem when using relation operator for comparison.

The age variable is changed to int datatype for comparison.

def get\_age():

    age = input("Please enter your age: ")

    if age.isnumeric() and int(age) >= 18:

        return int(age)

    else:

        return None

def main():

    age = get\_age()

    if age>=18:

        print(f"You are {age} years old and eligible.")

    else:

        print("Invalid input. You must be at least 18 years old.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



Objective: To identify and fix errors in a Python program that reads and writes to a file.  
Code3:

def read\_and\_write\_file(filename):

    try:

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

            content = file.read()

        with open(filename, 'w') as file:

            file.write(content.upper())

        print(f"File '{filename}' processed successfully.")

    except Exception as e:

        print(f"An error occurred: {str(e)}")

def main():

    filename = "sample.txt"

    read\_and\_write\_file(filename)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

submit the corrected code with comments explaining the issues they found and the solutions they implemented.

SOLUTION:

The error for the code is wrong file destination.

Correct the file destination by creating the missing file.

def read\_and\_write\_file(filename):

    try:

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

            content = file.read()

        with open(filename, 'w') as file:

            file.write(content.upper())

        print(f"File '{filename}' processed successfully.")

    except Exception as e:

        print(f"An error occurred: {str(e)}")

def main():

    filename = "C:/Users/Danny/OneDrive/Desktop/sample.txt"

    read\_and\_write\_file(filename)

if \_\_name\_\_ == "\_\_main\_\_":

    main()



Code4:

def merge\_sort(arr):

    if len(arr) <= 1:

        return arr

    mid = len(arr) // 2

    left = arr[:mid]

    right = arr[mid:]

    merge\_sort(left)

    merge\_sort(right)

    i = j = k = 0

    while i < len(left) and j < len(right):

        if left[i] < right[j]:

            arr[k] = left[i]

            i += 1

        else:

            arr[k] = right[j]

            j += 1

        k += 1

    while i < len(left):

        arr[k] = left[i]

        i += 1

        k += 1

    while j < len(right):

        arr[k] = right[j]

        j += 1

        k += 1

arr = [38, 27, 43, 3, 9, 82, 10]

merge\_sort(arr)

print(f"The sorted array is: {arr}")

The code aims to implement the merge sort algorithm. However, there is a bug in the code. When the student runs this code, it will raise an error or produce incorrect output. The student's task is to identify and correct the bug.

Hint: Pay close attention to the recursive calls and the merging step.

SOLUTION: The sorted array gets printed properly without any issue.

SOLUTION: def merge\_sort(arr):

    if len(arr) <= 1:

        return arr

    mid = len(arr) // 2

    left = arr[:mid]

    right = arr[mid:]

    merge\_sort(left)

    merge\_sort(right)

    i = j = k = 0

    while i < len(left) and j < len(right):

        if left[i] < right[j]:

            arr[k] = left[i]

            i += 1

        else:

            arr[k] = right[j]

            j += 1

        k += 1

    while i < len(left):

        arr[k] = left[i]

        i += 1

        k += 1

    while j < len(right):

        arr[k] = right[j]

        j += 1

        k += 1

arr = [38, 27, 43, 3, 9, 82, 10]

merge\_sort(arr)

print(f"The sorted array is: {arr}")

