**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:**

The code which provided does not contain any syntax errors, and it successfully reverses the input string and prints the reversed string. However, it does add an extra space at the beginning of the reversed string. If we want to remove the leading space, we need modify the **reverse\_string** function.by adding **return reversed.strip()** in **reverse\_string** , Using this strip method to remove leading and trailing spaces.

**Solved code:**

def reverse\_string(s):

reversed = ""

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

reversed += s[i]

return reversed.strip() # Remove leading and trailing spaces

def main():

input\_string = "Hello, world!"

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

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

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

main()

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:**

In the provided code, there are a couple of issues:

1. The **input()** function returns a string, so when we check **age.isnumeric()**, we are comparing a string to an integer, which will result in a **TypeError**. To fix this, we should convert age to an integer before comparing it.
2. It's also a good practice to handle the case when the user enters a non-numeric input gracefully

**Here's the corrected code:**

def get\_age():

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

if age.isnumeric():

age = int(age) # Convert the input to an integer

if age >= 18:

return age

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()

**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()

**solution:**The provided code attempts to read the content of a file, convert it to uppercase, and then write the uppercase content back to the same file. There is a potential issue in this code. When we open the file for writing ('w' mode), it will truncate the file, deleting its contents before writing the new content. This means that if we run this code more than once on the same file, it will keep emptying the file. If we want to preserve the original content while writing the uppercase version, we should open the file in append mode ('a') instead of write mode ('w').

**Here's the corrected code:**

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

try:

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

content = file.read()

with open(filename, 'a') as file: # Use 'a' mode to append to the 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()

**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.

**Solution:**The issue in the code is that the recursive calls to **merge\_sort(left)** and **merge\_sort(right)** do not correctly update the left and right subarrays within each recursion. In Python, when we pass a list to a function, it is passed by reference, but the code doesn't capture the return values from the recursive calls. This means that the sorting of the left and right subarrays is not properly reflected in the original arr. To fix this, we need to assign the results of the recursive calls back to left and right.

**Here's the corrected code:**

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = arr[:mid]

right = arr[mid:]

# Recursively sort left and right subarrays

left = merge\_sort(left)

right = 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}")