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()

# ERRORS:

=>There are no errors in the above program. It is a valid Python code that reverses a string using a for loop and string concatenation

.

# The Error explanation for the code is:

The function reverse\_string(s) takes a string s as a parameter and returns a new string that is the reverse of s.

The function initializes an empty string reversed to store the reversed string.

The function uses a for loop to iterate over the characters of s in reverse order, from the last index to the first index, using the range function with a negative step of -1.

The function appends each character of s to the reversed string using the += operator.

The function returns the reversed string after the loop ends.

The main function calls the reverse\_string function with the input string “Hello, world!” and assigns the result to the variable reversed\_string.

The main function prints the reversed\_string using the formatted string literal f"Reversed string: {reversed\_string}".

The main function is executed only if the file is run as a script, not as a module, by checking the \_\_name\_\_ variable against the string "\_\_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()

# ERRORS:

* The error is in the line if age.isnumeric() and age >= 18:. This line will raise a TypeError exception because you are trying to compare a string (age) with an integer (18). You should convert age to an integer before comparing it with 18, or use the int() function inside the isnumeric() method, like this: if age.isnumeric() and int(age) >= 18:.
* The suggestion is to use a loop to ask the user for their age until they enter a valid input, instead of returning None and ending the program. You can use a while loop with a break statement, like this:

# CORRECT CODE:

# 

def get\_age():

while True:

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

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

return int(age)

else:

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

# 

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()

# The errors and suggestions are:

* You used ‘r’ mode to open the file for reading, and then ‘w’ mode to open the same file for writing. This will erase the original content of the file before you can read it. You should use ‘r+’ mode instead, which allows you to read and write the same file without erasing it.
* You did not seek to the beginning of the file before writing the content in uppercase. This will append the uppercase content to the end of the file, instead of replacing the original content. You should use file.seek(0) to move the file pointer to the beginning of the file before writing.
* You did not truncate the remaining content of the file after writing the uppercase content. This will leave some lowercase content at the end of the file, if the original content was longer than the uppercase content. You should use file.truncate() to remove any content after the current file pointer.

# CORRECT CODE:

A function to read and write a file in uppercase

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

try:

# Use 'r+' mode to read and write the same file

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

content = file.read()

# Seek to the beginning of the file

file.seek(0)

# Write the content in uppercase

file.write(content.upper())

# Truncate any remaining content

file.truncate()

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.

(next page)

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.

ERRORS AND EXPLANATION:

* The error is in the line merge\_sort(left) and merge\_sort(right). These lines are missing the return statement, which means that the recursive calls will not return the sorted subarrays to the parent call. You should add the return statement to these lines, like this: left = merge\_sort(left) and right = merge\_sort(right).
* The suggestion is to use the += operator to simplify the code and avoid using the index variable k. You can append the elements of the subarrays to the main array using the += operator, like this: arr += left[i:] and arr += right[j:].

# CORRECT CODE:

# A function to implement the merge sort algorithm

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = arr[:mid]

right = arr[mid:]

# Use the return statement to get the sorted subarrays

left = merge\_sort(left)

right = merge\_sort(right)

i = j = 0

# Clear the main array

arr.clear()

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

if left[i] < right[j]:

arr.append(left[i])

i += 1

else:

arr.append(right[j])

j += 1

# Use the += operator to append the remaining elements

arr += left[i:]

arr += right[j:]

return arr

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

merge\_sort(arr)

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

#### The output for the above program is:

The sorted array is: [3, 9, 10, 27, 38, 43, 82]