**Assignment 1:**

**Code 1:**

def reverse\_string(s):

reversed\_str = ""

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

reversed\_str += s[i]

return reversed\_str

def main():

input\_string = "Hello, world!"

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

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

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

main()

**Correction in the code:** There's a minor issue with the variable name "reversed." It's a built-in Python function.

**Code 2:**

def get\_age():

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

try:

age = int(age)

if age >= 18:

return age

else:

return None

except ValueError:

return None

def main():

age = get\_age()

if age is not None:

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()

**Correction in the code:** The age variable is a string, so it has to be converted into an integer before comparing it to 18. We should use int(age) for the comparison.

we can use a **“try...except”** block to catch any exceptions when converting the input to an integer.

**Code 3:**

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

try:

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

content = file.read()

# Modify the content (e.g., convert to uppercase)

modified\_content = content.upper()

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

file.write(modified\_content)

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()

**Correction in the code:** The code is having potential issue because it reads the file and immediately overwrites it with the modified content, effectively erasing the original content.

If the goal is to read the content, modify it, and then save it, We should store the modified content in a separate variable before writing it back to the file.

**Code 4:**

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = arr[:mid]

right = arr[mid:]

left = merge\_sort(left) # Recursively sort the left half

right = merge\_sort(right) # Recursively sort the right half

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

return arr

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

merge\_sort(arr)

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

**Correction in the code:** The issue in the code is that the merge sort function sorts the sub-arrays correctly within its recursive calls, but it does not correctly merge these sorted sub-arrays back into the original array.