**Task-1-Debugging**

**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()

**Output:**

Reversed string: !d1row,olleH

**Explanation:**

The "reversed" variable, which is incompatible with the built-in reverse() function, is a minor issue.To avoid a name conflict, the variable name "reversed" has been changed to "reversed-str".

**Code-2:**

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()

**Output:**

Please enter your age: 19  
You are 19 years old and eligible.

**Explanation:**

Since the input() function returns a string, age must first be converted to an integer before it can be compared to 18.

Instead of using the "isnumeric()" function, which examines whether a string contains numeric characters, use the "isdigit()" method to discover whether a string exclusively contains numbers. (0-9)

**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()

**Output:**

File 'pendulum.txt' processed successfully.  
**Note:**I used “pendulum.txt” file

**Explanation:**

The given code immediately opens the file in write mode after opening it in read mode first ('r'). You lose the original text before it is changed to uppercase because the file's content is truncated. After reading the material, you should exit the file and reopen it in write mode to enter the updated content. This will resolve the problem and benefit your code.

In this code:

1. To open the file for reading and writing, we use the "with" command. Because of this, even if an exception happens, the file will always be closed after the code block ends.

2. Using the "with" phrase, we close the file after reading the contents.

3. Following the writing of the revised text, we once more close the file using the "with" phrase.

4. During file operations, potential exceptions are detected and displayed using the proper error handling.

**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}")

**Output:**

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

**Explanation:**

In this code, the merge sort method is utilized. The function does not return the sorted array, which is a minor issue. Either the function should create a new sorted array and return it, or it should return the sorted array from the merge sort.

Within this updated code:

The sorted array is what we have the merge\_sort method return.

When executing merge\_sort(arr), we allocate the sorted array to a new variable called sorted\_arr.

Using the sorted\_arr variable, we print the sorted array.