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()

**COMMENTS:This code is correct and executes without any errors although the number of lines in the code can be reduced by using slicing technique to define the reverse\_string function. The optimized code would be**

**def reverse\_string(s):**

**return s[::-1]**

**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()

**COMMENT: There is TypeError in the code.In the line of the code which is highlighted in bold, age.isnumeric teats age variable as a string that accepts only numerical values while in the next part we give a condition that age >= 18 and python can accept this statement without any error only if the variable age is of the integer data type. Hence to rectify this error we will have to convert the string data typed variable age into integer type. The rectified code is as follows**

**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:**

**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\_\_":

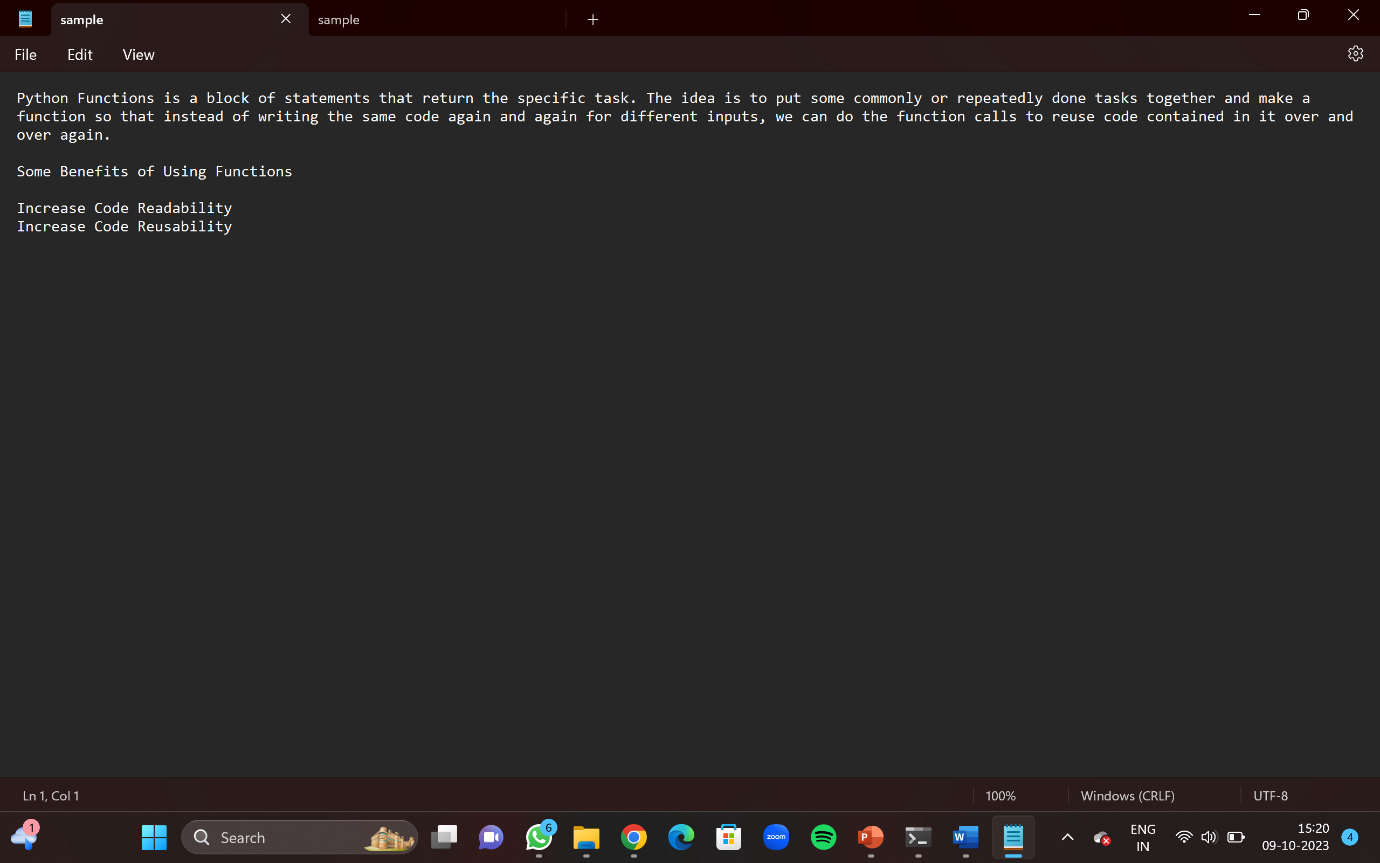
    main()

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

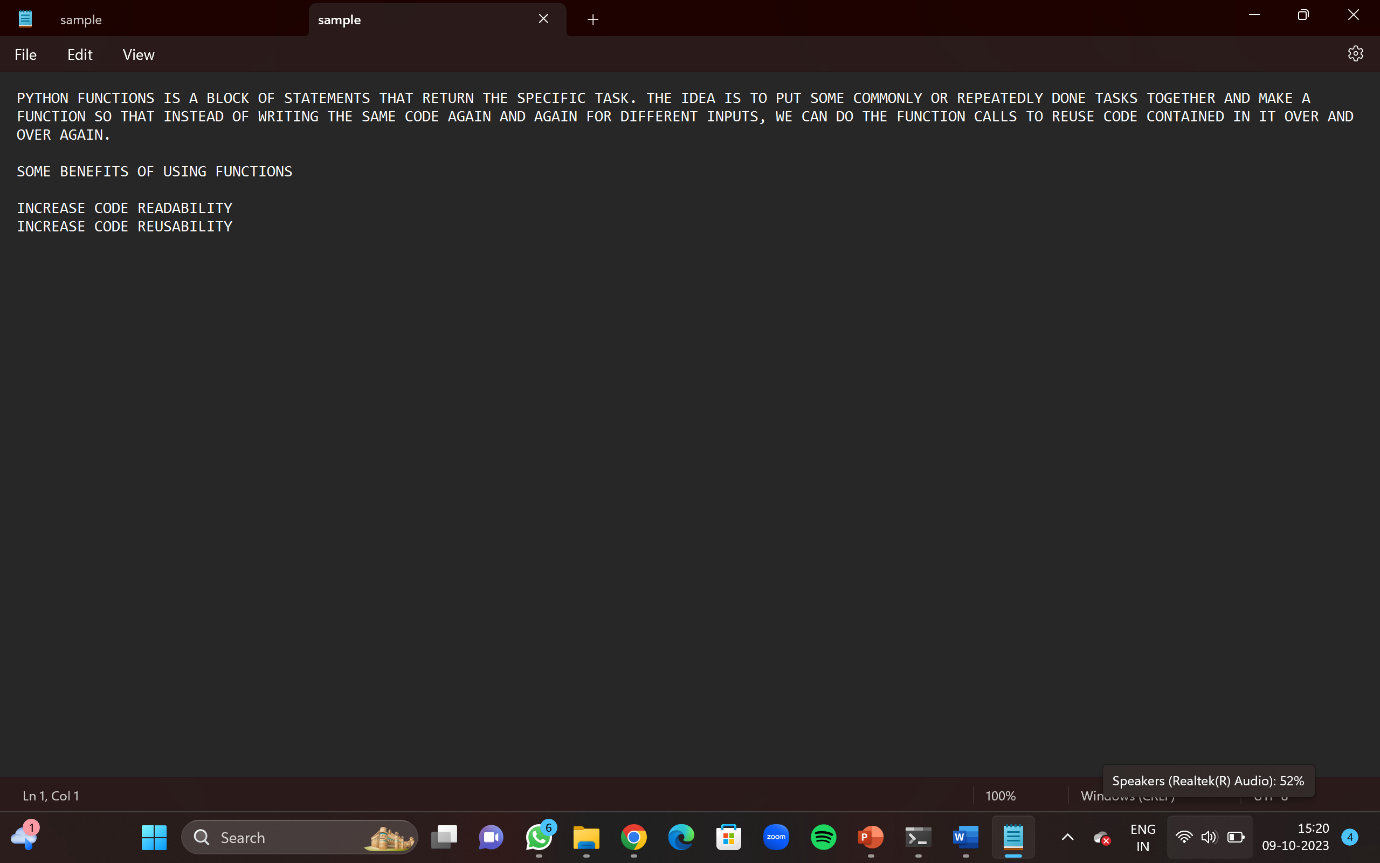
**COMMENT: The above given code executed with no errors.**

****

**The sample.txt file before execution of the code:**

****

**The sample.txt file after execution of the code:**

  
**Although one thing is to be kept in mind. We have to make sure the sample.txt file exists in the same directory as the python file that you are executing the code in. Else you will get an error saying No such file or directory**

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.

**COMMENT: The given code is missing one line of code. After the merge\_sort function has been written, the return keyword to end the function and return the calculated value of the function to the function name and variable is not written. The corrected code would be**

**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**

**return arr**

**arr = [38, 27, 43, 83, 91, 82, 10, 10]**

**merge\_sort(arr)**

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