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()
```

The code appears to be correct and it will work without any errors.It defines a function named reverse_string which takes string as input and produces the reverse of that string and returns that reversed string,and a main function that demonstrates the usage of the reverse_string.There are no syntax errors or logical errors in the code.

Output: Reversed string: !dlrow ,olleH

Code2:

There is an error in the given code because the variable age is taking input as a string but we are comparing that string with an integer 18 in the if statement,which could lead to a **Type error.**

The corrected code is:

```
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()
```

Output:

```
Please enter your age: 56
You are 56 years old and eligible.
```

Code3:

The code opens the file for writing inside the with block after opening it for reading.This will truncate the file and erase its content before reading it,resulting in the content being lost.The code reads the content and converts it to uppercase but doesn't handle potential exceptions that may occur when opening or manipulating the file.

The corrected code is:

```
def read_and_write_file(filename):
    try:
       #open the file for reading
        with open(filename, 'r') as file:
            content = file.read()
    #open the same file for writing(this will overwrite the content)
        with open(filename, 'w') as file:
             #write the content in uppercase to the file
            file.write(content.upper())
      #print a success message
        print(f"File '{filename}' processed successfully.")
    except Exception as e:
#Handle any exceptions that might occur
        print(f"An error occurred: {str(e)}")
```

```
def main():
    filename = "sample.txt"
    read_and_write_file(filename)

if __name__ == "__main__":
    main()
```

Code4:

The issue is that the merge_sort function is not returning any value,and it relies on modifying the input arr in-place.However,in the provided code,you are not returning the array explicitly.To fix this,you should add a return statement to return the sorted arr.

The corrected code is:

```
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 #added this line to return the sorted array

arr = [38, 27, 43, 3, 9, 82, 10]
sorted_arr=merge_sort(arr)#store the sorted array in a variable
print(f"The sorted array is: {sorted_arr}")
```

Output:

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