Assignment-1

Submitted by

B.Chaitanya

Csc-2ndyear

SVCE

Code:1

```python
# Original code
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()
```

```python
# Modified code
def reverse_string(s):
    reversed_str = ""
    for char in reversed(s):
        reversed_str += char
    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()
```

Explanation:

1. **Variable Naming:**

Original: The variable **reversed** was used, but it is a built-in function name in Python. It's not a good practice to use built-in names for variables.
Modified: Changed the variable name to **reversed_str** to avoid conflicts with the built-in name.

1. **Reversing a String:**

Original: Used a for loop with a range to iterate over the string in reverse and concatenate characters to the reversed variable.
Modified: Used the **reversed()** function to reverse the string, making the code more concise and readable.
These changes improve code readability and adhere to best practices in Python programming

Code:2

```python
# Original code
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()
```

```python
# Modified code
def get_age():
    age_str = input("Please enter your age: ")
    if age_str.isnumeric() and int(age_str) >= 18:
        return int(age_str)
    else:
        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.")
```

Explanation:

1. **Variable Naming:**
   - Original: Used the variable name `age` for both the string input and the integer representation of age. It's clearer to differentiate between the two.
   - Modified: Changed the variable name to `age_str` for the string input to make the code more readable.

2. **Comparison Issue:**
   - Original: Used `age >= 18` directly without converting `age` to an integer, which can lead to a TypeError.
   - Modified: Checked if `int(age_str) >= 18` to ensure that the comparison is done after converting the input to an integer.

3. **Return Value:**
   - Original: Returned `None` if the input age was invalid. However, `None` is a valid value and might be misleading.
   - Modified: Changed the return value to `None` if the input is invalid and used `is not None` for checking in the `main` function.

Code 3

```python
# Original code
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()
```

```python
# Modified code
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 FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    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()
```

Explanation:

1. **Error Handling:**
   - Original: Used a broad `Exception` to catch any errors that might occur during file processing.
   - Modified: Added a specific exception `FileNotFoundError` to handle the case where the file is not found. This provides more accurate error information.

2. **Handling File Not Found:**
   - Original: If the file is not found, the code would proceed to the next block and attempt to open the file for writing, which would create an empty file.
   - Modified: Added a specific block to handle `FileNotFoundError` and print an appropriate error message. This prevents the code from proceeding when the file is not found.

Code 4

```python
# Original code
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}")
```

```python
# Modified code
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

# Additional code to return the sorted array
def main():
    arr = [38, 27, 43, 3, 9, 82, 10]
    merge_sort(arr)
    print(f"The sorted array is: {arr}")

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

Explanation:

1**. Print in a Function:**

- Original: The sorted array was printed directly after calling the `merge_sort` function.
- Modified: Created a `main` function to encapsulate the array initialization, sorting, and printing. This is a cleaner structure and follows the convention of having the main logic inside a function.