Assignment-7.2

Batch 30

2303a51927

Task 1 – Runtime Error Due to Invalid Input Type

• A Python program accepts user input and performs arithmetic operations. However, the program throws a runtime error because the input is treated as a string instead of a numeric type.

Use AI tools to identify the cause of the runtime error and modify the program so it executes correctly.

Prompt:

Analyze the following Python code. It throws a runtime error when performing arithmetic operations on user input. Identify the issue and modify the code so that it executes correctly.

Buggy Code and Output:

```
Lab_7.2.py > ...
1   num = input("Enter a number: ")
2   result = num + 10
3   print(result)
4
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.py"
Enter a number: 10
Traceback (most recent call last):
  File "c:\Users\DELL\Documents\AI Assitant\Lab_7.2.py", line 2, in <module>
    result = num + 10
             ~~~~^~~~
TypeError: can only concatenate str (not "int") to str
```

AI-Corrected Code and Output:

```
num = int(input("Enter a number: "))
result = num + 10
print(result)
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsAp
Enter a number: 10
20
```

Justification:

The runtime error occurs because the input() function in Python returns a string by default, even when numeric input is provided by the user. The program attempts to add an integer value to this string, which results in a TypeError during execution. AI assistance helped in

identifying this data type mismatch efficiently. The AI explained that arithmetic operations require numeric data types and cannot be performed on strings. It suggested converting the input into an integer using type casting. After applying the correction, the program executed successfully without any runtime errors. This task demonstrates how AI helps resolve input-related runtime issues effectively.

TASK 2: Incorrect Function Return Value

A function is designed to calculate the square of a number, but it does not return the computed result properly.

Use AI assistance to analyze the function and ensure the correct value is returned.

Prompt :

Review this Python function that is intended to return the square of a number. Identify why it does not return any value and correct the function.

Buggy Code and Output:

```
8
9   def square(n):
0       result = n * n
1
2
```

AI-Corrected Code and Output:

```
12  def square(n):
13      return n * n
14
15  print(square(5))
16
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.
25
```

Justification:

The function correctly calculates the square of a number but fails to return the computed result. In Python, a function must explicitly return a value using the return statement, otherwise it returns None by default. AI assistance helped identify this logical issue in the function implementation. The AI explained that without a return statement, the calculated value cannot be accessed by the calling function. It suggested adding a return statement to fix the issue. After modification, the function returned the correct output. This task highlights how AI helps detect and correct logical errors in functions.

Task 3 – IndexError in List Traversal

A Python program iterates over a list using incorrect index limits,

causing an IndexError.

Use AI to identify the incorrect loop boundary and correct the iteration

logic.

Prompt:

Identify the cause of the IndexError in the list traversal code and correct the loop condition.

Buggy Code and Output:

```
numbers = [10, 20, 30]
for i in range(0, len(numbers)+1):
    print(numbers[i])
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.py"
10
20
30
Traceback (most recent call last):
  File "c:\Users\DELL\Documents\AI Assitant\Lab_7.2.py", line 19, in <module>
    print(numbers[i])
          ~~~~~~~^^^
IndexError: list index out of range
```

AI-Corrected Code and Output:

```
numbers = [10, 20, 30]
for i in range(len(numbers)):
    print(numbers[i])
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.py"
10
20
30
```

Justification:

The program raises an IndexError because the loop attempts to access a list index that is outside the valid range. In Python, list indices start from zero and go up to one less than the length of the list. The loop condition incorrectly exceeds this range, leading to an out-of-bounds access. AI assistance helped identify this off-by-one error in the loop logic. The AI explained the correct index boundaries for list traversal. It suggested correcting the loop condition to prevent invalid access. After correction, the program executed safely without errors.

Task 4 – Uninitialized Variable Usage

A program uses a variable in a calculation before assigning it any value.

Use AI tools to detect the uninitialized variable and correct the program.

Prompt:

Detect and fix the error caused by using an uninitialized variable in the given Python code.

Buggy Code and Output:

```
25  if True:
26      pass
27  print(total)
28
29
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.py"
Traceback (most recent call last):
  File "c:\Users\DELL\Documents\AI Assitant\Lab_7.2.py", line 27, in <module>
    print(total)
          ^^^^^
NameError: name 'total' is not defined
```

AI-Corrected Code and Output:

```
28
29  total = 0
30  if True:
31      total += 10
32  print(total)
33
```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.py"
10
```

Justification:

The error occurs because a variable is used before it is assigned any value. Python does not allow the use of uninitialized variables and raises a runtime error when such access occurs. AI assistance detected that the variable had not been initialized in the program. The AI explained the importance of assigning an initial value before using a variable. It suggested initializing the variable at the beginning of the program. After applying the fix, the program ran successfully without errors. This task demonstrates how AI helps enforce safe variable usage practices.

Task 5 – Logical Error in Student Grading System

A grading program assigns incorrect grades due to improper conditional

logic.

Use AI to analyze the grading conditions and correct the logical flow.

Prompt:

Analyze the grading logic in the Python program and correct the conditions so that grades are assigned accurately.

Buggy Code and Output:

```
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "C"
else:
    grade = "B"
print(grade)
```

AI-Corrected Code and Output:

```
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "B"
else:
    grade = "C"
print(grade)

```

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_7.2.py"
B
```

Justification:

The grading program executes successfully but produces incorrect grades due to faulty conditional logic. The conditions used to assign grades are improperly structured, leading to wrong output. AI assistance helped analyze the logic flow of the conditional statements. The AI explained how incorrect condition ordering and grade mapping affect the final result. It suggested correcting the grading conditions to match the required criteria. After modification, the grades were assigned accurately. This task shows how AI helps identify and correct logical errors in decision-making programs.