

## ASSIGNMENT-7.2

NAME: HABEEBA KHANAM

HT.NO:2303A51474

BATCH:29

**LAB 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs**

### TASK DESCRIPTION

#### 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.

Example (Buggy Code):

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

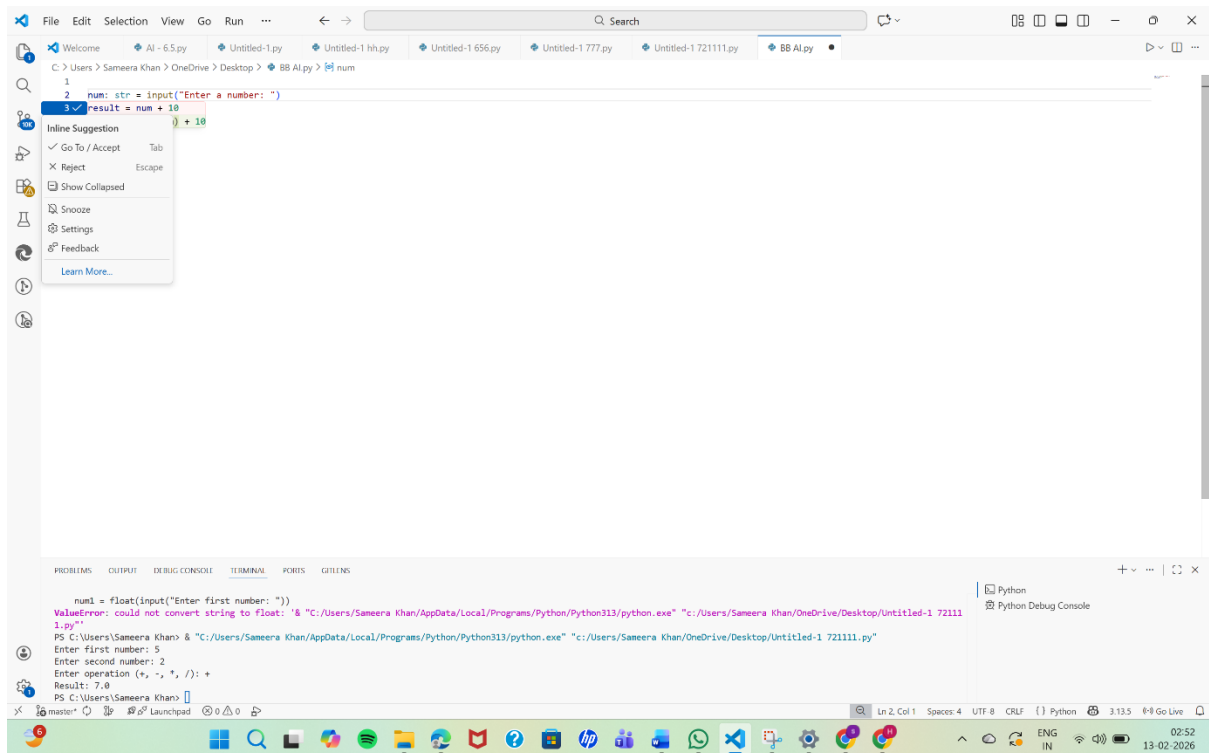
- **TASK:**

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

#### EXPECTED OUTPUT -1:

- AI converts the input to the appropriate numeric type and eliminates the runtime error.

# CODE:



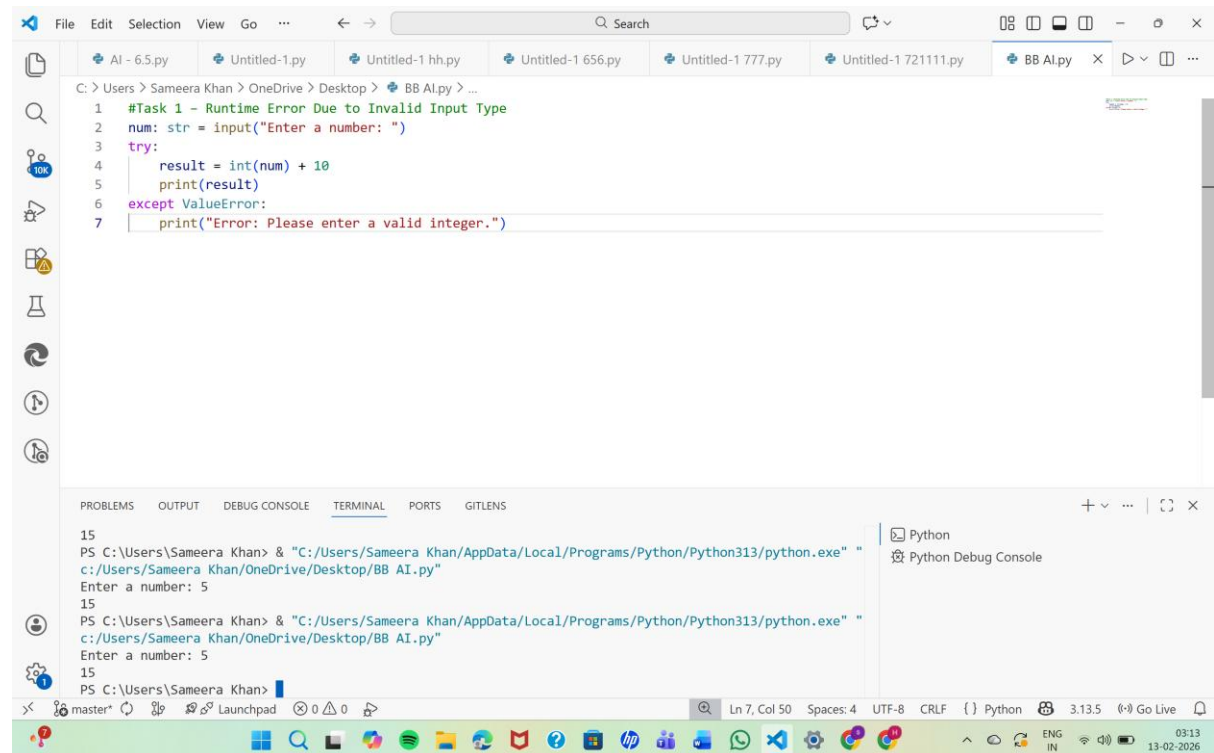
The screenshot shows the Visual Studio Code editor with a Python file named 'BB AI.py'. The code contains a simple addition program. An inline suggestion menu is open over the line `result = num + 10`, showing options like 'Go To / Accept', 'Tab', 'Reject', 'Escape', 'Show Collapsed', 'Snooze', 'Settings', 'Feedback', and 'Learn More...'. The terminal at the bottom shows the execution of the script, which results in a `ValueError: could not convert string to float: '1.721111'` because the input '1.721111' is not a valid integer.

```
1 num: str = input("Enter a number: ")
2 result = num + 10
```

ValueError: could not convert string to float: '1.721111'

PS C:\Users\Sameera Khan> "C:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\Sameera Khan\OneDrive\Desktop\Untitled-1 721111.py"

Enter first number: 5  
Enter second number: 2  
Enter operation (+, -, \*, /): +  
Result: 7.0



The screenshot shows the Visual Studio Code editor with a Python file named 'BB AI.py'. The code is updated to include a try-except block to handle the ValueError. The terminal at the bottom shows the execution of the script, which now correctly handles the input '5' and prints the result '15'.

```
1 #Task 1 - Runtime Error Due to Invalid Input Type
2 num: str = input("Enter a number: ")
3 try:
4     result = int(num) + 10
5     print(result)
6 except ValueError:
7     print("Error: Please enter a valid integer.")
```

PS C:\Users\Sameera Khan> "C:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\Sameera Khan\OneDrive\Desktop\BB AI.py"

Enter a number: 5  
15

PS C:\Users\Sameera Khan> "C:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\Sameera Khan\OneDrive\Desktop\BB AI.py"

Enter a number: 5  
15

# JUSTIFICATION:

- `int(input())` converts the string input into an integer.
- Arithmetic operations can now be performed safely.
- The runtime error is eliminated.

## **TASK DESCRIPTION**

### **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.

Example (Buggy Code):

```
def square(n):  
    result = n * n
```

### **TASK:**

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

### **EXPECTED OUTPUT -2:**

AI fixes the missing return statement and the function returns the correct output

### **CODE:**

The screenshot shows the VS Code editor with a Python file named 'BB AI.py'. The code defines a function `square(n)` that calculates the square of `n` and returns the result. The function is called with `square(5)`, and the output is `25`. The terminal shows the command `python BB AI.py` being executed, and the output is `25`. The error message 'Error Due to Invalid Input Type' is visible in the terminal, indicating that the input is not a valid integer.

```
def square(n):  
    result = n * n  
    return result  
print(square(5)) # Output: 25
```

The screenshot shows the VS Code editor with a Python file named 'BB AI.py'. The code defines a function `square(n)` that calculates the square of `n` and returns the result. The function is called with `square(5)`, and the output is `25`. The terminal shows the command `python BB AI.py` being executed, and the output is `25`. The error message 'Error Due to Invalid Input Type' is visible in the terminal, indicating that the input is not a valid integer.

```
#Task 2 - Incorrect Function Return Value  
def square(n):  
    result = n * n  
    return result  
print(square(5)) # Output: 25
```

# JUSTIFICATION:

- `n * n` correctly computes the square.
- `return result` ensures the value is passed back.

- The function now behaves as expected.

## **TASK DESCRIPTION**

### **TASK 3 – INDEX ERROR IN LIST TRAVERSAL**

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

causing an `IndexError`.

Example (Buggy Code):

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

#### **TASK:**

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

logic.

#### **EXPECTED OUTPUT -3:**

AI fixes the loop condition and prevents out-of-range list access.

#### **CODE:**

The screenshot shows the Visual Studio Code editor with a Python file named 'BB AI.py'. The code is as follows:

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

An inline suggestion menu is open for the range function on line 2, showing options: 'Go To / Accept' (Tab), 'Reject' (Escape), 'Show Collapsed', 'Snooze', 'Settings', and 'Feedback'. The terminal at the bottom shows the command prompt running the script, which results in an 'IndexError: list index out of range'.

The screenshot shows the Visual Studio Code editor with the same Python file 'BB AI.py'. The code is as follows:

```
1 # Task 3 - IndexError in List Traversal
2
3 numbers = [10, 20, 30]
4 for i in range(0, len(numbers)):
5     print(numbers[i])
```

The terminal at the bottom shows the command prompt running the script, which results in an 'IndexError: list index out of range'.

## JUSTIFICATION:

- `range(0, len(numbers))` prevents out-of-range access.

- The enhanced loop avoids index handling entirely.
- The Index Error is fully eliminated.

## TASK 4 – UNINITIALIZED VARIABLE USAGE

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

Example (Buggy Code):

if True:

pass

print(total)

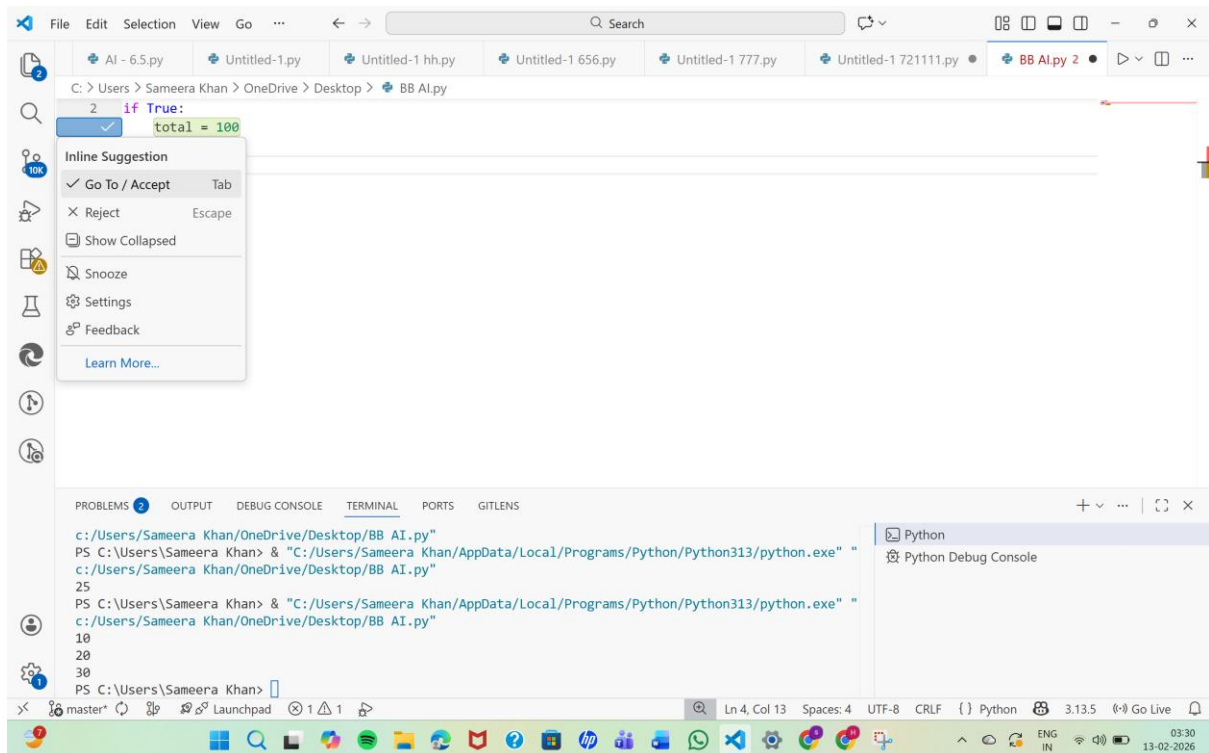
### TASK:

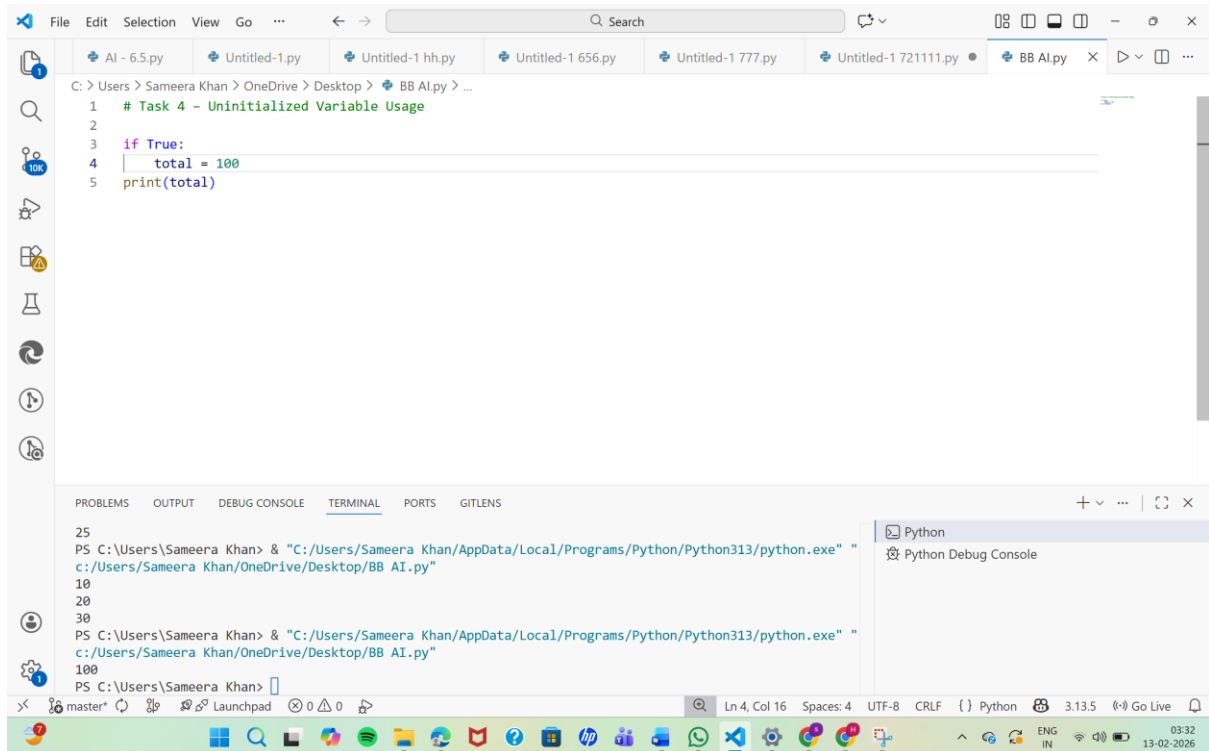
Use AI tools to detect the uninitialized variable and correct the program.

### EXPECTED OUTPUT -4:

AI initializes the variable correctly before it is used

### CODE:





```
File Edit Selection View Go ... Search
AI - 6.5.py Untitled-1.py Untitled-1 hh.py Untitled-1 656.py Untitled-1 777.py Untitled-1 721111.py BB AI.py
C: > Users > Sameera Khan > OneDrive > Desktop > BB AI.py > ...
1 # Task 4 - Uninitialized Variable Usage
2
3 if True:
4     total = 100
5     print(total)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "
c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"
100
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "
c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"
100
PS C:\Users\Sameera Khan>
```

## JUSTIFICATION:

- Initializing total ensures it exists in memory.
- The program can now safely access and print the variable.
- The runtime error is eliminated.

## TASK 5 – LOGICAL ERROR IN STUDENT GRADING SYSTEM

A grading program assigns incorrect grades due to improper conditional logic.

Example (Buggy Code):

marks = 85



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

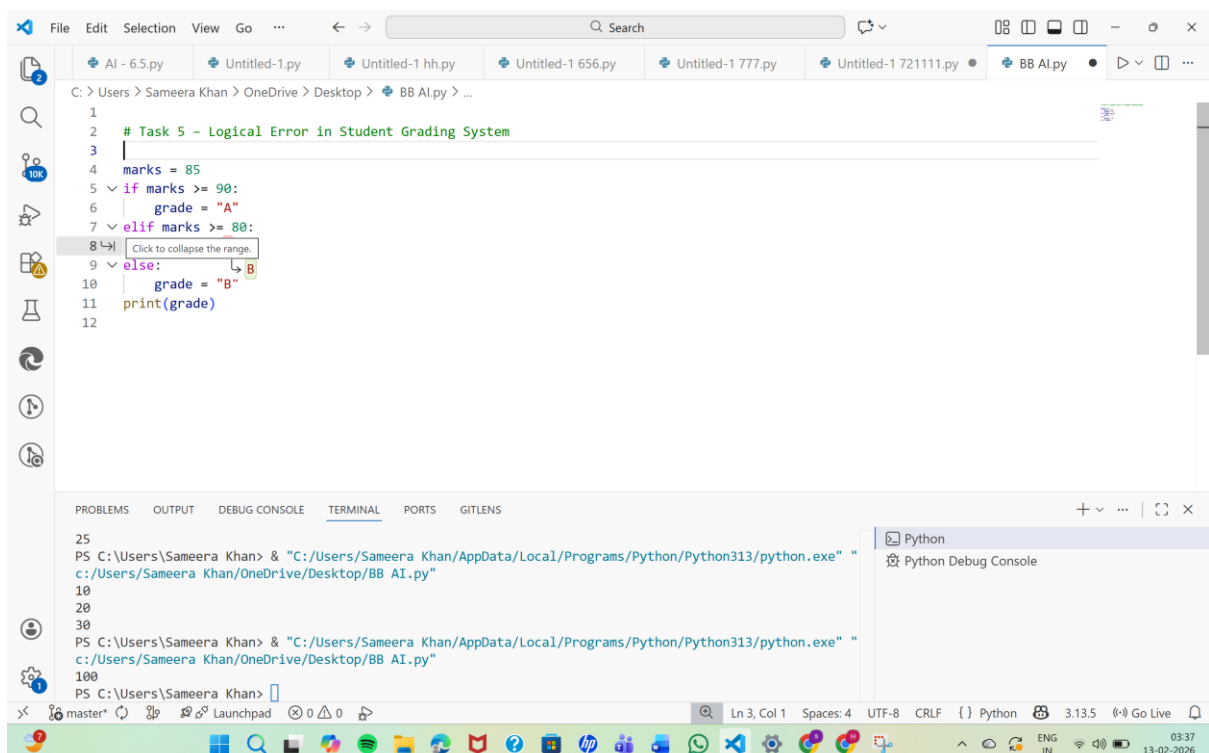
## TASK:

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

## EXPECTED OUTPUT -5:

AI corrects the conditional logic so grades are assigned accurately.

## CODE:



The screenshot shows a Visual Studio Code editor window with a Python file named 'BB AI.py'. The code is as follows:

```
1  
2 # Task 5 - Logical Error in Student Grading System  
3  
4 marks = 85  
5 if marks >= 90:  
6     grade = "A"  
7 elif marks >= 80:  
8  
9 else:  
10    grade = "B"  
11 print(grade)  
12
```

The output of the script is displayed in the terminal at the bottom of the editor:

```
PS C:\Users\Sameera Khan> "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "  
c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"  
10  
20  
30  
PS C:\Users\Sameera Khan> "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "  
c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"  
100  
PS C:\Users\Sameera Khan>
```

The terminal output shows that the script is executed successfully, and the grade 'B' is printed for the given marks of 85.

VS Code interface showing a Python file named `BB AI.py`. The code contains a logical error in the student grading system. The terminal output shows the command being run:

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"
```

VS Code interface showing the corrected Python file named `BB AI.py`. The code is now correct. The terminal output shows the command being run:

```
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"
```

## JUSTIFICATION:

- Initializing total ensures it exists in memory.
- The program can now safely access and print the variable.
- The runtime error is eliminated.

