Assignment – 7.2

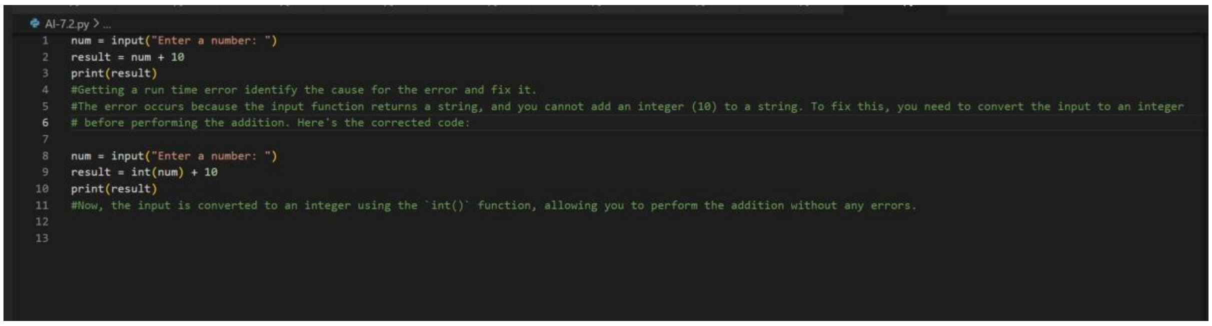
2303A51599

Task 1 – Runtime Error Due to Invalid Input Type

PROMPT :

Getting a run time error identify the cause for the error and fix it.

CODE :

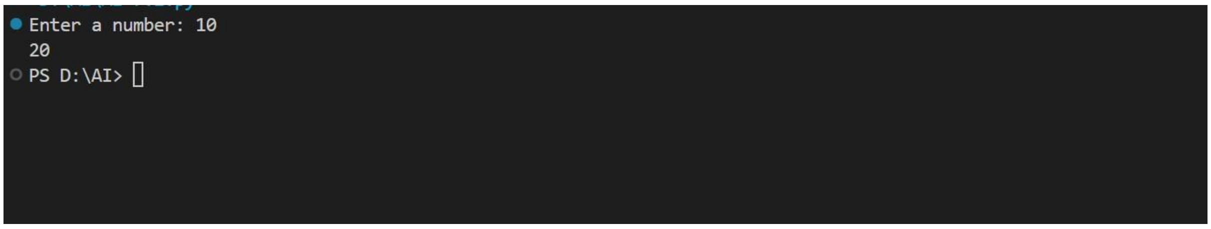


AI-7.2.py > ...

```python
num = input("Enter a number: ")
result = num + 10
print(result)
#Getting a run time error identify the cause for the error and fix it.
#The error occurs because the input function returns a string, and you cannot add an integer (10) to a string. To fix this, you need to convert the input to an integer
# before performing the addition. Here's the corrected code:

num = input("Enter a number: ")
result = int(num) + 10
print(result)
#Now, the input is converted to an integer using the `int()` function, allowing you to perform the addition without any errors.
```

OUTPUT :



```
Enter a number: 10
20
PS D:\AI> 
```

Justification :

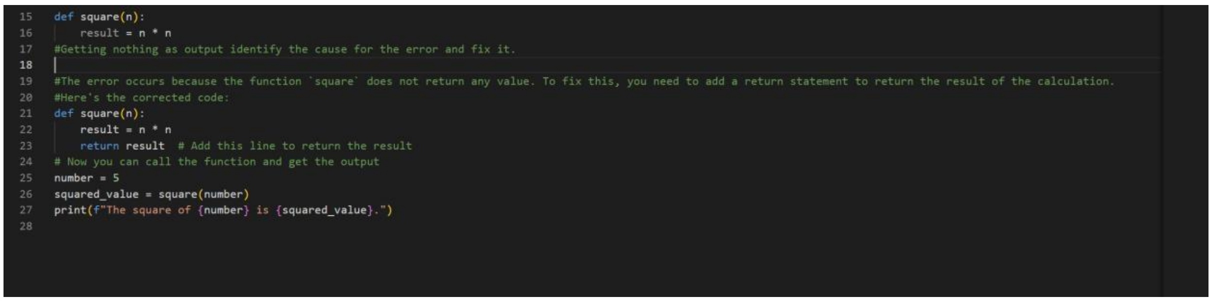
AI helped me understand that the input() function in Python returns a string by default, which caused a TypeError when performing arithmetic operations. By converting the input into an integer using type casting, the runtime error was resolved successfully. This task improved my understanding of data types and their importance in arithmetic operations. It also increased my confidence in identifying and fixing type-related runtime errors. Task 2

PROMPT :

Task 1 :-Incorrect Function Return Value

Not getting an output identify the cause for the error and fix it.

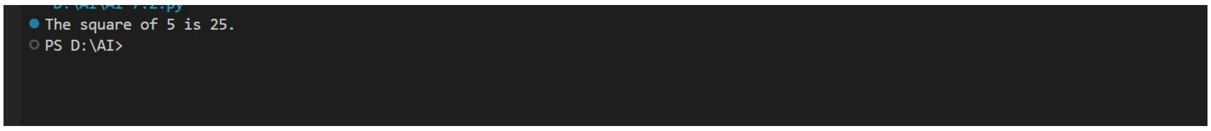
CODE :



```python
def square(n):
    result = n * n
#Getting nothing as output identify the cause for the error and fix it.

#The error occurs because the function `square` does not return any value. To fix this, you need to add a return statement to return the result of the calculation.
#Here's the corrected code:
def square(n):
    result = n * n
    return result  # Add this line to return the result
# Now you can call the function and get the output
number = 5
squared_value = square(number)
print(f"The square of {number} is {squared_value}.")
```

OUTPUT :



```
The square of 5 is 25.
PS D:\AI>
```

Justification :

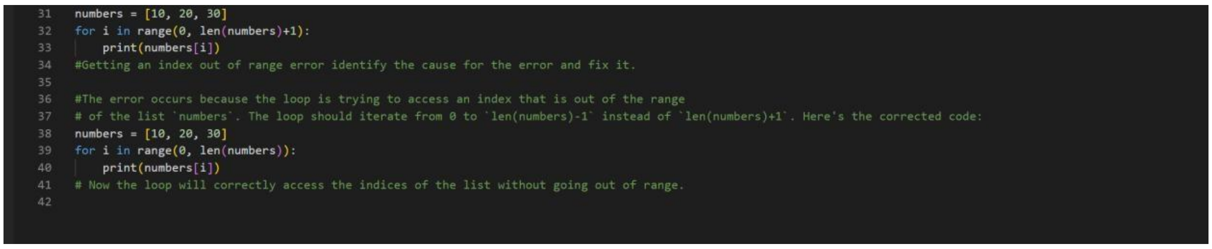
AI assisted in identifying that the function was missing a return statement, which caused it to return None instead of the expected value. By adding the appropriate return statement, the function was able to correctly provide the computed result. This helped me better understand how functions work internally and why returning values is important. It strengthened my knowledge of function structure and behavior in Python.

Task 3 IndexError in List Traversal

PROMPT :

Getting an index out of range error identify the cause for the error and fix it.

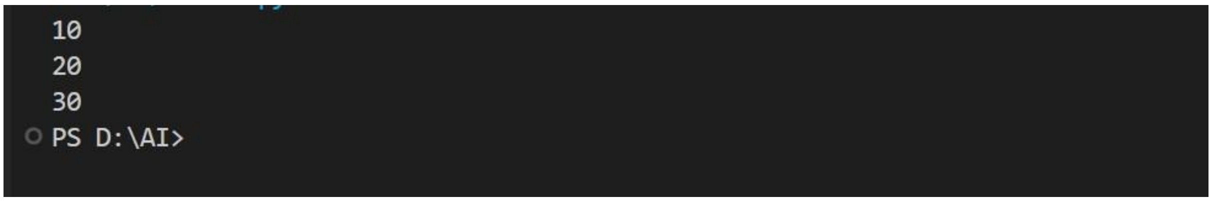
CODE :



```python
numbers = [10, 20, 30]
for i in range(0, len(numbers)+1):
    print(numbers[i])
#Getting an index out of range error identify the cause for the error and fix it.

#The error occurs because the loop is trying to access an index that is out of the range
# of the list `numbers`. The loop should iterate from 0 to `len(numbers)-1` instead of `len(numbers)+1`. Here's the corrected code:
numbers = [10, 20, 30]
for i in range(0, len(numbers)):
    print(numbers[i])
# Now the loop will correctly access the indices of the list without going out of range.
```

OUTPUT :



```
10
20
30
PS D:\AI>
```

Justification :

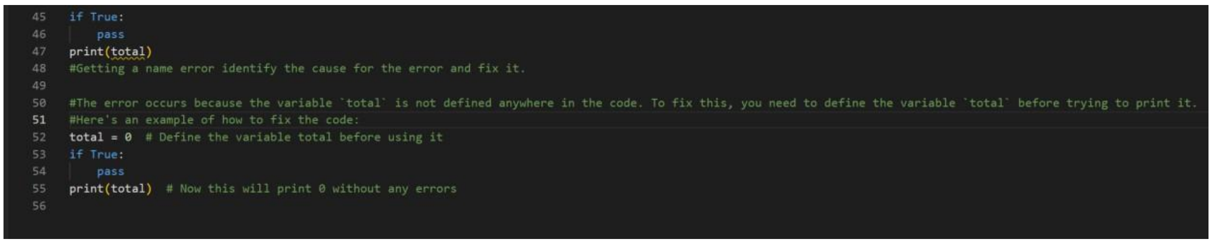
AI detected that the loop boundary condition exceeded the valid index range of the list, which caused an IndexError. By correcting the loop range and removing the extra increment, the program executed without errors. This task helped me understand the importance of proper indexing and boundary conditions when working with lists. It also improved my debugging skills related to iteration logic.

Task 4 Uninitialized Variable Usage

PROMPT :

Getting a name error identify the cause for the error and fix it.

CODE :

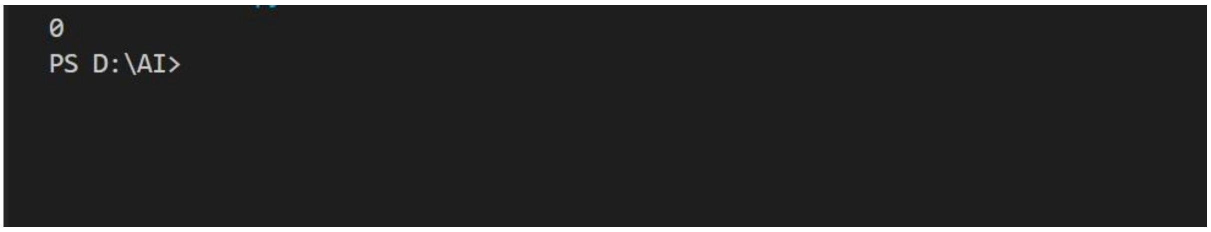


```python
if True:
    pass
print(total)
#Getting a name error identify the cause for the error and fix it.

#The error occurs because the variable `total` is not defined anywhere in the code. To fix this, you need to define the variable `total` before trying to print it.
#Here's an example of how to fix the code:
total = 0  # Define the variable total before using it
if True:
    pass
print(total)  # Now this will print 0 without any errors

```

OUTPUT :



```
0
PS D:\AI>
```

Justification :

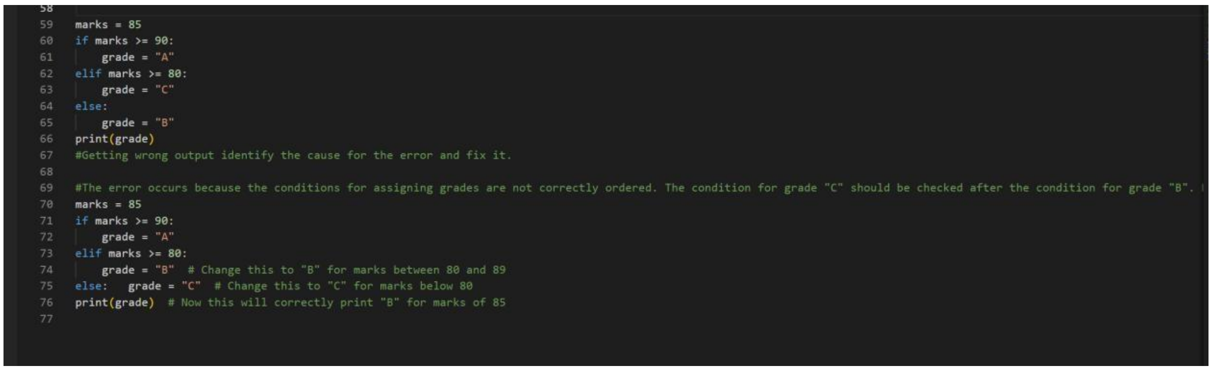
AI identified that a variable was being used before it was initialized, leading to a NameError. By initializing the variable before using it in the program, the error was successfully resolved. This task highlighted the importance of defining variables before usage. It enhanced my understanding of variable scope and initialization in Python.

Task 5 Logical Error in Student Grading System

PROMPT :

Getting wrong output identify the cause for the error and fix it.

CODE :

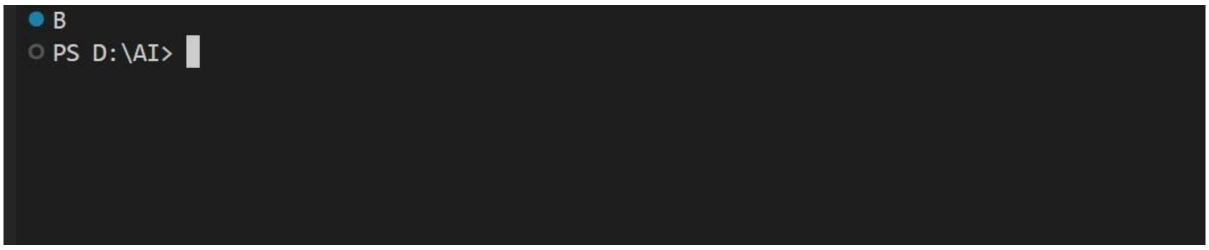


```python
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "C"
else:
    grade = "B"
print(grade)
#Getting wrong output identify the cause for the error and fix it.

#The error occurs because the conditions for assigning grades are not correctly ordered. The condition for grade "C" should be checked after the condition for grade "B".
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "B"  # Change this to "B" for marks between 80 and 89
else:   grade = "C"  # Change this to "C" for marks below 80
print(grade)  # Now this will correctly print "B" for marks of 85

```

OUTPUT :



```
B
PS D:\AI>
```

Justification :

AI analyzed the grading logic and found that the conditional structure was incorrectly assigning grades. By reorganizing and correcting the conditional statements, the program was able to assign accurate grades based on marks. This task improved my understanding of logical flow and condition hierarchy in decision-making statements. It also helped me realize how logical errors can affect program correctness even without syntax issues.