Name: Shryitha Suddala

Hall Ticket:2303A51637

Batch:29

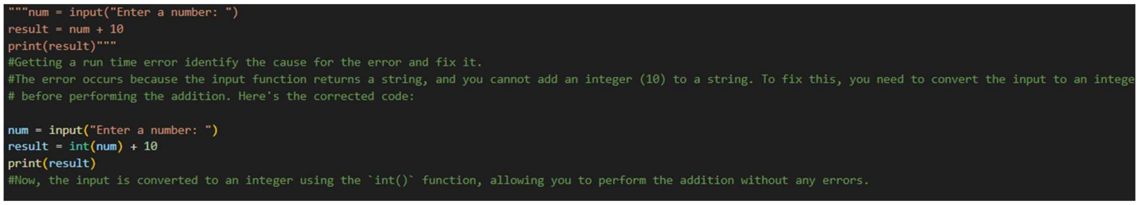
Assessment: 7.2

Task 1 – Runtime Error Due to Invalid Input Type

Prompt:

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

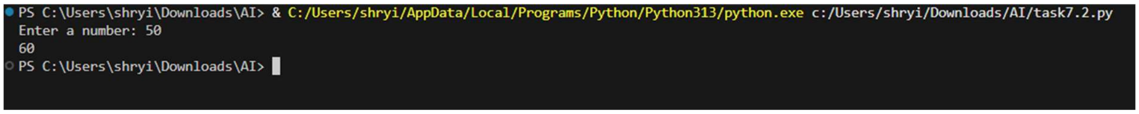
Code :



```
"""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 intege
# 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:



```
PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/task7.2.py
Enter a number: 50
60
PS C:\Users\shryi\Downloads\AI> 
```

Justification :

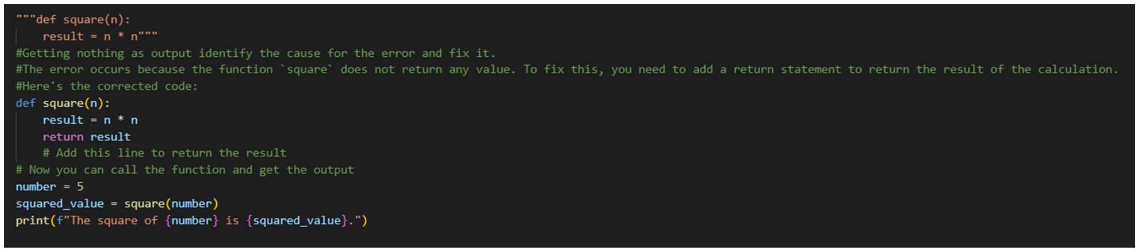
The runtime error occurred because the input received from the user was treated as a string, while the program attempted to perform an arithmetic operation on it. Python does not automatically convert string input into numbers. By explicitly converting the input value to an integer, the program was able to perform the addition operation correctly. This task clarified how Python handles user input and reinforced the need for proper type conversion before performing calculations.

Task 2 :Incorrect Function Return Value

Prompt:

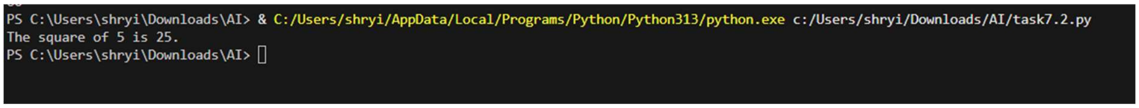
Not getting an output identify the cause for the error and fix it.

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 :



```
PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/task7.2.py
The square of 5 is 25.
PS C:\Users\shryi\Downloads\AI> 
```

Justification :

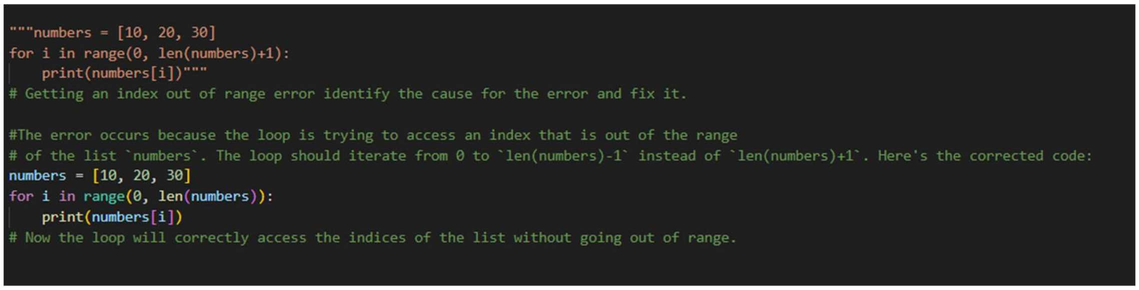
The function executed correctly but did not produce any visible output because it did not return the computed value. In Python, functions return None by default if no return statement is specified. Adding a return statement ensured that the calculated result was passed back to the calling function. This task helped reinforce the concept of function output flow and the role of return statements in retrieving results from functions.

Task 3: IndexError in List Traversal

Prompt :

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

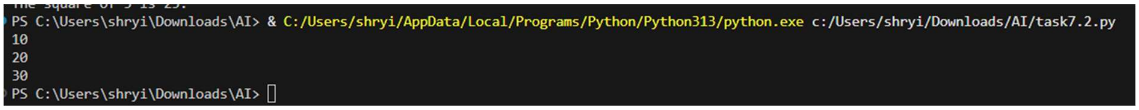
Code :



```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 :



```
PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/task7.2.py
10
20
30
PS C:\Users\shryi\Downloads\AI> 
```

Justification :

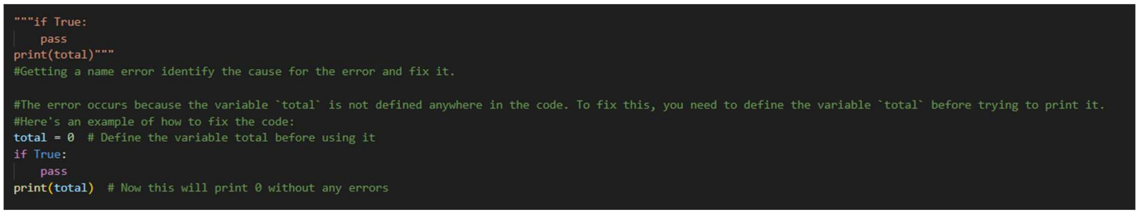
The error was caused by attempting to access a list index that does not exist. Python lists are zero-indexed, and accessing elements beyond their length results in an IndexError. By adjusting the loop to iterate only within the valid index range of the list, the program executed safely. This task improved my understanding of list indexing and safe iteration techniques in Python.

Task 4 :Uninitialized Variable Usage

Prompt :

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

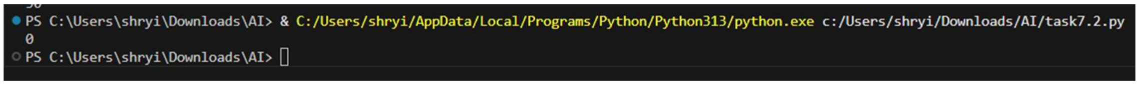
Code:



```
"""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:



```
● PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/task7.2.py
  0
○ PS C:\Users\shryi\Downloads\AI> ▯
```

Justification :

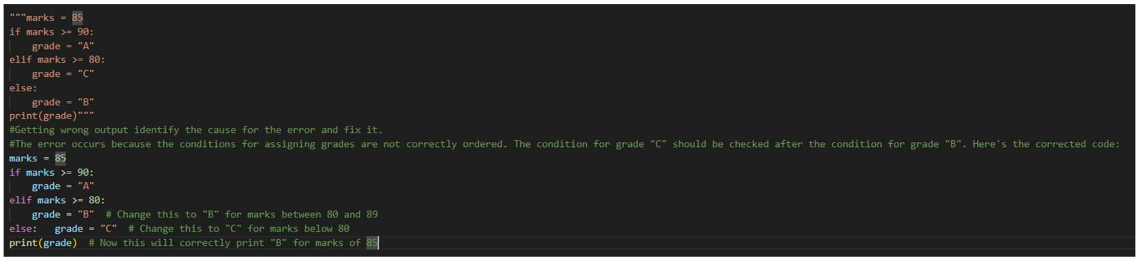
The NameError occurred because the variable was referenced before being defined in the program. Python requires variables to be initialized before they are accessed. Initializing the variable with a default value resolved the issue. This task emphasized the importance of variable declaration order and helped strengthen my understanding of how Python interprets variable names during execution.

Task 5: Logical Error in Student Grading System

Prompt:

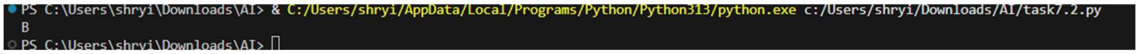
Getting wrong output identify the cause for the error and fix it.

Code :



```
"""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". Here's the corrected code:
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:



```
● PS C:\Users\shryi\Downloads\AI> & C:/Users/shryi/AppData/Local/Programs/Python/Python313/python.exe c:/Users/shryi/Downloads/AI/task7.2.py
  B
○ PS C:\Users\shryi\Downloads\AI> ▯
```

Justification :

The incorrect output was due to improper ordering of conditional statements. Since conditions are evaluated sequentially, the program matched a broader condition before checking the more specific one. Reordering the conditions ensured that higher-grade criteria were evaluated first. This task enhanced my understanding of logical sequencing and the importance of condition priority in decision-making statements.