

ASSIGNMENT – 7.2

HT NO:2303A51980

BATCH NO:30

TASK-1:

PROMPT:

```
num = input("Enter a number: ")
```

```
result = num + 10
```

```
print(result)
```

#identify the cause of the runtime error and modify

the program so it executes correctly.

CODE:

The screenshot shows a dark-themed code editor interface. On the left, there's a sidebar with icons for file operations like search, refresh, and save. The main area has a title bar with "OPEN EDITORS 3 unsaved". Below the title bar, there are three tabs: "7.2.py" (which is the active tab), "7.2.PY", and another "7.2.py". The code editor itself contains the following Python script:

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

Below the code editor is a terminal window showing the execution of the script:

```
/usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py
● syedsufiyan@syeds-MacBook-Air-3 7.2.py % /usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py
Enter a number: 200
710
● syedsufiyan@syeds-MacBook-Air-3 7.2.py %
```

The terminal output shows that the script runs successfully, but it fails to handle non-integer input as expected. A message at the bottom right of the terminal window says, "You've reached the limit for chat... Upgrade".

OBSERVATION:

The program takes user input using `input()`, which returns a string by default.

When the code tries to add 10 to this string, a type mismatch occurs, causing a runtime error.

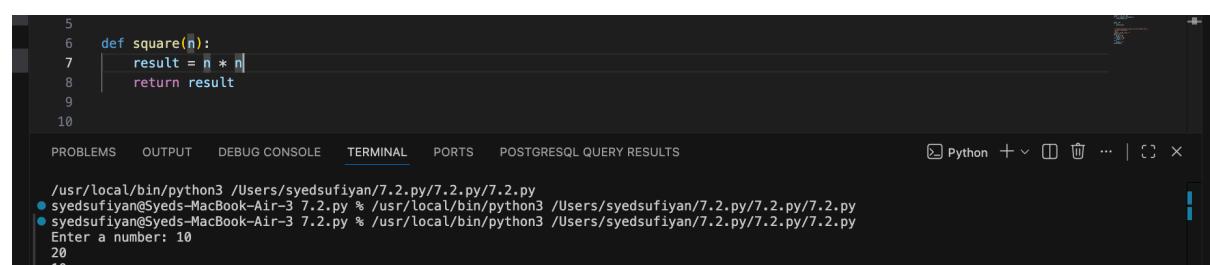
The error happens because arithmetic operations cannot be performed between a string and an integer without converting the input to a numeric type

TASK-2

PROMPT:

```
def square(n):
    result = n * n
    # Analyze the function and ensure the correct value
    # is returned.
    #fixes the missing return statement and the function
    #returns the correct output.
```

CODE:



A screenshot of a terminal window in a dark-themed IDE. The window shows a Python script named `7.2.py` with the following code:

```
5
6  def square(n):
7      result = n * n
8      return result
9
10
```

Below the code, the terminal tab is active, showing the command `/usr/local/bin/python3 /Users/syedsufiyan/7.2.py` and its output:

```
/usr/local/bin/python3 /Users/syedsufiyan/7.2.py
syedsufiyan@Syeds-MacBook-Air-3:~%
```

The user then enters `Enter a number: 10`, followed by `20` and `10`.

OBSERVATION:

In the given function, the value of $n * n$ is calculated and stored in the variable result, but it is never returned from the function.

Because of the missing return statement, the function does not send any value back to the caller and returns None by default.

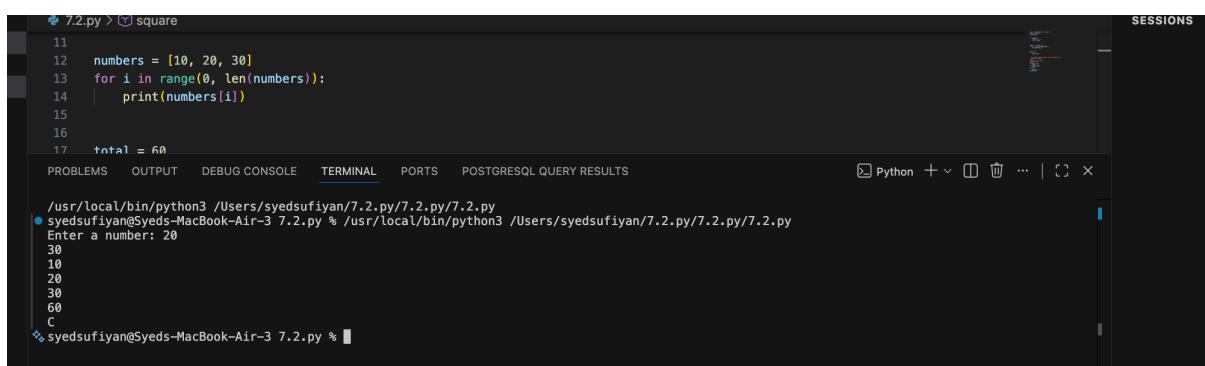
As a result, the expected output is not produced even though the computation is performed. The bug occurs due to the absence of a return statement in the function definition

TASK-3

PROMPT:

```
numbers = [10, 20, 30]
for i in range(0, len(numbers)+1):
    print(numbers[i])
#incorrect loop boundary and correct the iteration
logic.
```

CODE:



The screenshot shows a code editor interface with a dark theme. The code in the editor is:

```
7.2.py > square
11
12     numbers = [10, 20, 30]
13     for i in range(0, len(numbers)):
14         print(numbers[i])
15
16
17     total = 60
```

Below the code editor is a terminal window showing the execution of the script:

```
/usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py
syedsufiyan@Syeds-MacBook-Air-3:~$ /usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py
Enter a number: 20
30
20
30
60
C
syedsufiyan@Syeds-MacBook-Air-3:~$
```

OBSERVATION:

The loop runs one step beyond the last valid index because it uses `len(numbers) + 1` as the upper limit. This causes the program to try accessing an index that does not exist in the list, resulting in an `IndexError`. The error occurs due to an incorrect loop boundary that goes out of the list's valid range.

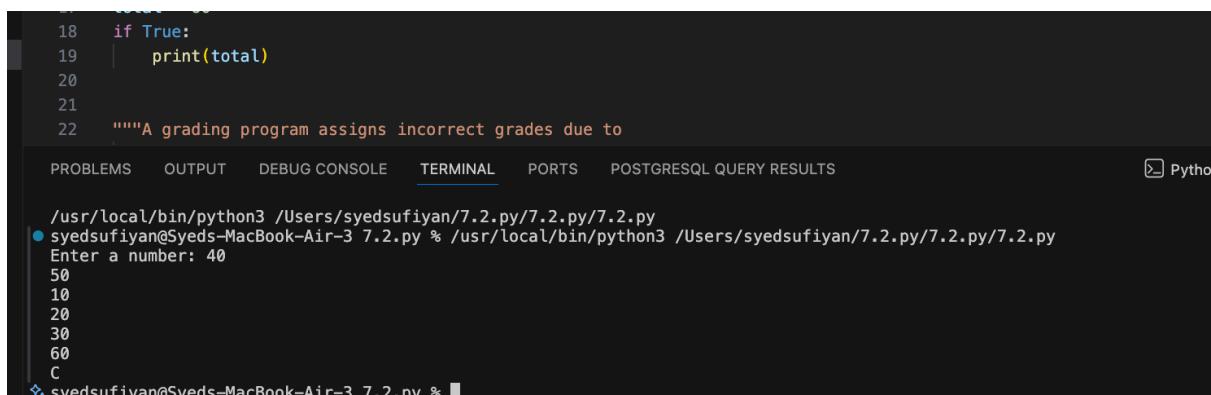
TASK-4

PROMPT:

if True:
pass
print(total)

To detect the uninitialized variable and correct the program.

CODE:



A screenshot of a terminal window in a code editor. The code is as follows:

```
18 if True:  
19     print(total)  
20  
21  
22 """A grading program assigns incorrect grades due to
```

The terminal output shows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRESQL QUERY RESULTS Python  
/usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py  
● syedsufiyan@Syeds-MacBook-Air-3 7.2.py % /usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py  
Enter a number: 40  
50  
10  
20  
30  
60  
C
```

The terminal prompt shows the user's name and the file path: `syedsufiyan@Syeds-MacBook-Air-3 7.2.py %`.

OBSERVATION:

In the given program, the variable total is used in the print statement without being assigned any value beforehand.

Since total is never initialized, Python raises a NameError when trying to access it.

The error occurs because the program attempts to use a variable before defining it.

To fix this, the variable must be initialized with a value before it is used in any calculation or output.

TASK-5

PROMPT:

"""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 = "B"
```

```
else:
```

```
    grade = "C"
```

OBSERVATION:

#analyze the grading conditions and correct the logical flow.

CODE:

```
22     A grading program assigns incorrect grades due to
23     improper conditional
24     logic.
25     Example (Buggy Code):"""
26
27     marks = 85
28     if marks >= 90:
29         grade = "A"
30     elif marks >= 80:
31         grade = "C"
32     else:
33         grade = "B"
34     print(grade)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRESQL QUERY RESULTS

● /usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py
syedsufiyan@Syeds-MacBook-Air-3:~ % /usr/local/bin/python3 /Users/syedsufiyan/7.2.py/7.2.py/7.2.py
Enter a number: 200
210
10
20
30
30
60

OBSERVATION:

In the given program, the grading conditions are logically incorrect.

For marks greater than or equal to 80, the grade is assigned as "C"

instead of a higher grade, and the else block assigns "B" for lower marks.

This causes wrong grade assignment because higher marks

should correspond to better grades. The logical flow of conditions is reversed, leading to incorrect output. The issue occurs due to improper ordering and incorrect grade mapping in the conditional statements.