

ASSIGNMENT – 7.2

HT NO:2303A51360

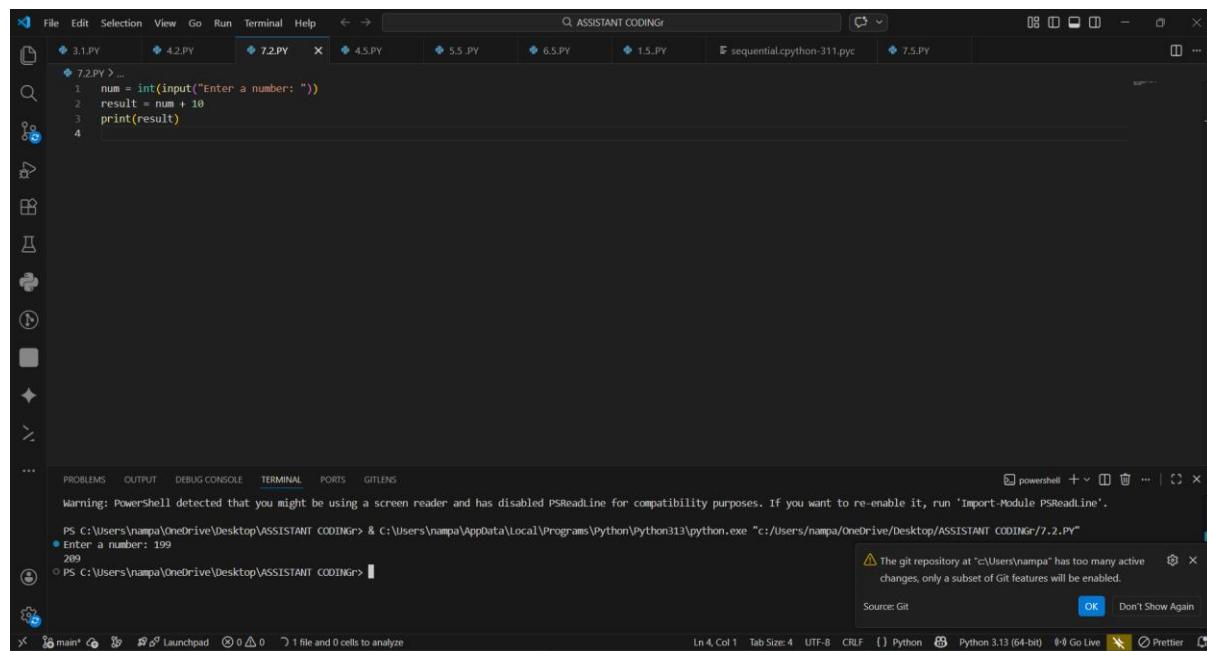
BATCH NO:29

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:



```
1 num = int(input("Enter a number: "))  
2 result = num + 10  
3 print(result)  
4  
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> Enter a number: 199  
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several tabs for Python files like 3.1.PY, 4.2.PY, 7.2.PY, 4.5.PY, 5.5.PY, 1.5.PY, sequential.cpython-311.pyc, and 7.5.PY. The main editor area contains the provided Python code. Below the editor is a terminal window showing the script's execution and a user input of 199. At the bottom, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS. A status bar at the bottom indicates the file is main*, has 0 changes, and 1 file and 0 cells to analyze. On the far right, a git status message says "The git repository at 'C:\Users\nampa' has too many active changes; only a subset of Git features will be enabled." There are OK and Don't Show Again buttons for this message. The bottom right corner also shows Python 3.13 (64-bit), Go Live, and Prettier icons.

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.

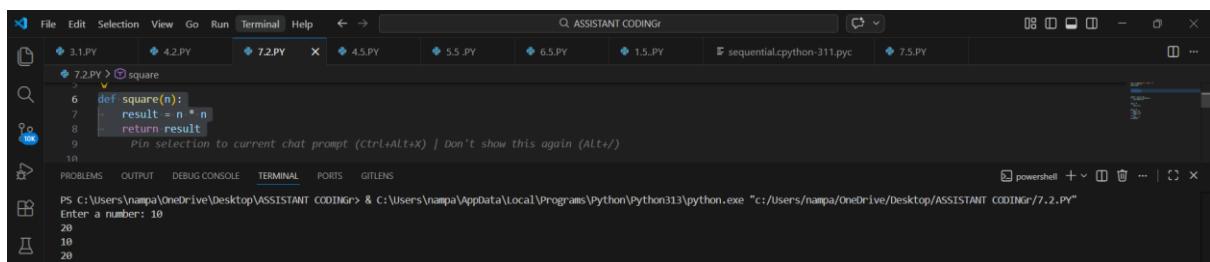
The error happens because arithmetic operations cannot be performed between a string and an integer without converting the input to a numeric type

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:



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

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:

```
File Edit Selection View Go Run Terminal Help <- > ASSISTANT CODING
3.1.PY 4.2.PY 7.2.PY X 4.5.PY 5.5.PY 1.5.PY sequential.python-311.py 7.5.PY
11
12     numbers = [10, 20, 30]
13     for i in range(0, len(numbers)):
14         print(numbers[i])
15
16 Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this again (Alt+/)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
powerShell + - x
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/7.2.py"
Enter a number: 20
30
20
30
60
C
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

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:

The screenshot shows the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. Below the menu is a tab bar with files 3.1.PY, 4.2.PY, 7.2.PY (which is the active file), 4.5.PY, 5.5.PY, 6.5.PY, 1.5.PY, sequential.cpython-311.pyc, and 7.5.PY. The main editor area contains the following code:

```
total = 60
if True:
    print(total)
```

A tooltip "Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this again (Alt+/)" appears over the code. Below the editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS. The DEBUG CONSOLE tab is selected, showing the output of running the script:

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadline for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadline'.
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/7.2.PY"
Enter a number: 40
50
10
20
30
60
C
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

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"
```

```
print(grade)
```

```
#analyze the grading conditions and correct the logical flow.
```

CODE:

```
File Edit Selection View Go Run Terminal Help ASSISTANT CODING 7.2.PY 3.1.PY 4.2.PY 4.5.PY 5.5.PY 6.5.PY 1.5.PY sequential.cpython-311.pyc 7.5.PY

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/7.2.py"
● Enter a number: 200
200
20
30
60
C
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING>
```

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.