

## School of Computer Science and Artificial Intelligence

**Lab Assignment # 7.2**

Program : B. Tech (CSE)  
Specialization :  
Course Title : AI Assisted Coding  
Course Code : 23CS002PC304  
Semester II  
Academic Session : 2025-2026  
Name of Student : A.NAGASAI  
Enrollment No. : 2403A51L18  
Batch No. : 51  
Date : 30-01-2026

**Task 1 – Runtime Error Due to Invalid Input Type**

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

```
[1] ① 5s ▶ num = input("Enter a number: ")
      result = num + 10
      print(result)

  ⌄ ... Enter a number: 2
  -----
  TypeError                                     Traceback (most recent call last)
  /tmp/ipython-input-1898169331.py in <cell line: 0>()
      1 num = input("Enter a number: ")
  ----> 2 result = num + 10
      3 print(result)

  TypeError: can only concatenate str (not "int") to str
```

Next steps: Explain error

**Output:**

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

Enter a number: 2  
12

## Task 2 – Incorrect Function Return Value

### **(Buggy Code):**

```
def square(n):
```

**result = n \* n**

```
s def square(n):
    result = n * n
...
File "/tmp/ipython-input-3910404483.py", line 2
    result = n * n
          ^
IndentationError: expected an indented block after function definition on line 1
```

## Output:

◆ Gemini

```
[ ] def square(n):
- result = n * n
+     result = n * n
```

```
[10]  ✓ 0s   def square(n):
          result = n * n
```

## Task-3 Index Error in List Traversal

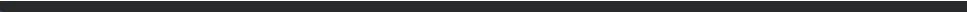
```
(Buggy Code): numbers = [10, 20,  
30]    for    i    in    range(0,  
len(numbers)+1):  
print(numbers[i])
```

```
[11] ① 0s
▶ numbers = [10, 20, 30]
for i in range(0, len(numbers)+1):
    print(numbers[i])

...
File "/tmp/ipython-input-726334973.py", line 3
    print(numbers[i])
    ^
IndentationError: expected an indented block after 'for' statement on line 2

Next steps: Explain error
```

## Output:



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

▼ ... 10  
20  
30

## Task 4 – Uninitialized Variable Usage

(Buggy Code):

```
if True: pass
```

```
print(total)
```

The screenshot shows a code editor window with the following code:

```
[13] 0s
if True:
    pass
    print(total)

...   File "/tmp/ipython-input-1170978020.py", line 2
        pass
        ^
IndentationError: expected an indented block after 'if' statement on line 1
```

Below the code, a message says "Next steps: Explain error".

**Output:**

The screenshot shows a code editor window with the following code:

```
Gemini
if True:
    -pass
    + pass
    print(total)
-
```

The screenshot shows a code editor window with the following code:

```
Gemini
if True:
    | | pass
    +total = 0 # Or any other initial value
    | | print(total)
```

... 0

## Task 5 – Logical Error in Student Grading System

(Buggy Code): marks = 85

```
if marks >= 90: grade =
```

```
"A" elif marks >= 80:
```

```
grade = "C" else:
```

```
grade ="B"
```

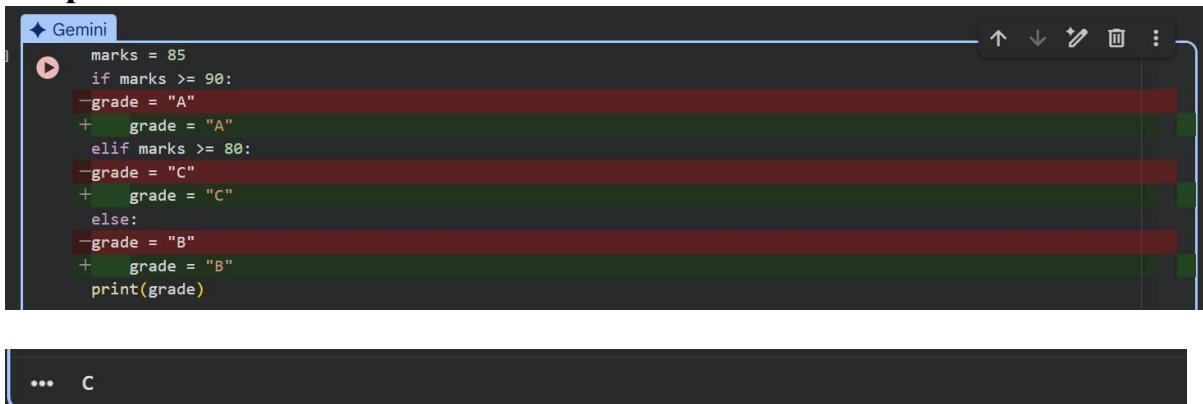
```
print(grade)
```

The screenshot shows a code editor window with the following code:

```
[16] 0s
marks = 85
if marks >= 90:
    grade = "A"
    elif marks >= 80:
        grade = "C"
    else:
        grade = "B"
    print[grade]

...   File "/tmp/ipython-input-2691675298.py", line 3
        grade = "A"
        ^
IndentationError: expected an indented block after 'if' statement on line 2
```

Below the code, a message says "Next steps: Explain error".

**Output:**

```
◆ Gemini
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "C"
else:
    grade = "B"
print(grade)
```

... C

The screenshot shows a code editor window titled "Gemini". Inside, there is a Python script. The script defines a variable `marks` with the value `85`. It then uses an `if` statement to check if `marks` is greater than or equal to `90`. If true, it assigns the string `"A"` to the variable `grade`. If false, it checks if `marks` is greater than or equal to `80`. If true, it assigns the string `"C"` to `grade`. Otherwise, it assigns the string `"B"`. Finally, it prints the value of `grade` using the `print` function. Below the code, the output is shown as "... C", indicating that the program has run and printed the letter "C".