SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

Program Name: B. Tech	**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	Venkataramana Veeramsetty	
Instructor(s) Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	

Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/III	**Regulation**	R24
Date and Day of Assignment	Week4 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	

Assignment Number:7.1 (Present assignment number)/**24**(Total number of assignments)

Q.No.	**Question**	***Expected Time to complete***
1	**Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs** Lab Objectives: • To identify and correct syntax, logic, and runtime errors in Python programs using AI tools.	Week4 - Monday

- To understand common programming bugs and AI-assisted debugging suggestions.
- To evaluate how AI explains, detects, and fixes different types of coding errors.
- To build confidence in using AI to perform structured debugging practices.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Use AI tools to detect and correct syntax, logic, and runtime errors.
- Interpret AI-suggested bug fixes and explanations.
- Apply systematic debugging strategies supported by AI-generated insights.
- Refactor buggy code using responsible and reliable programming patterns.

Task Description #1 (Syntax Errors – Missing Parentheses in Print Statement)

Task: Provide a Python snippet with a missing parenthesis in a print statement (e.g., print "Hello"). Use AI to detect and fix the syntax error.

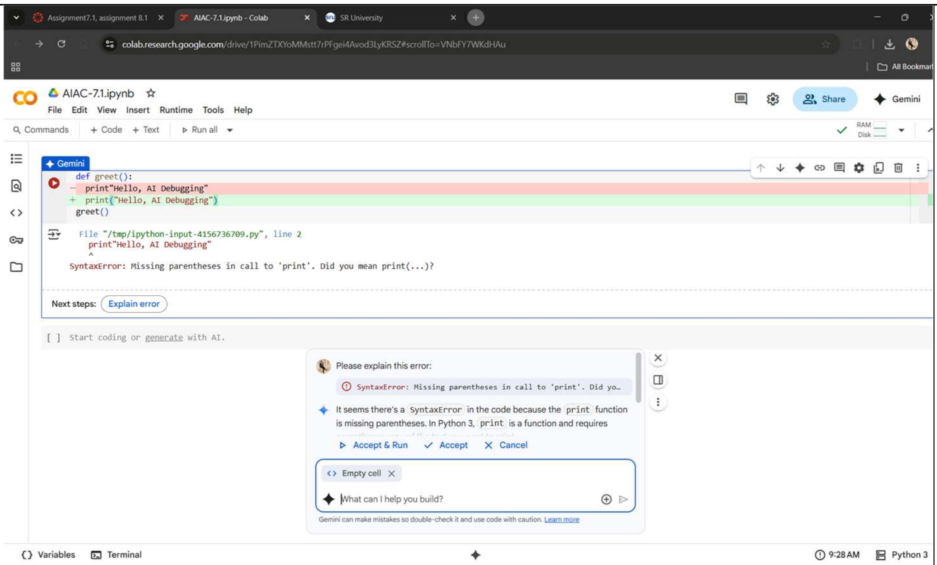
```
# Bug: Missing parentheses in print statement
        def greet():
            print "Hello, AI Debugging Lab!"
        greet()
```

Requirements:

- Run the given code to observe the error.
- Apply AI suggestions to correct the syntax.
- Use at least 3 assert test cases to confirm the corrected code works.

Expected Output #1:

- Corrected code with proper syntax and AI explanation.



```
def greet():
-  print"Hello, AI Debugging"
+  print("Hello, AI Debugging")
greet()
```

```
  File "/tmp/ipython-input-4156736709.py", line 2
    print"Hello, AI Debugging"
    ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

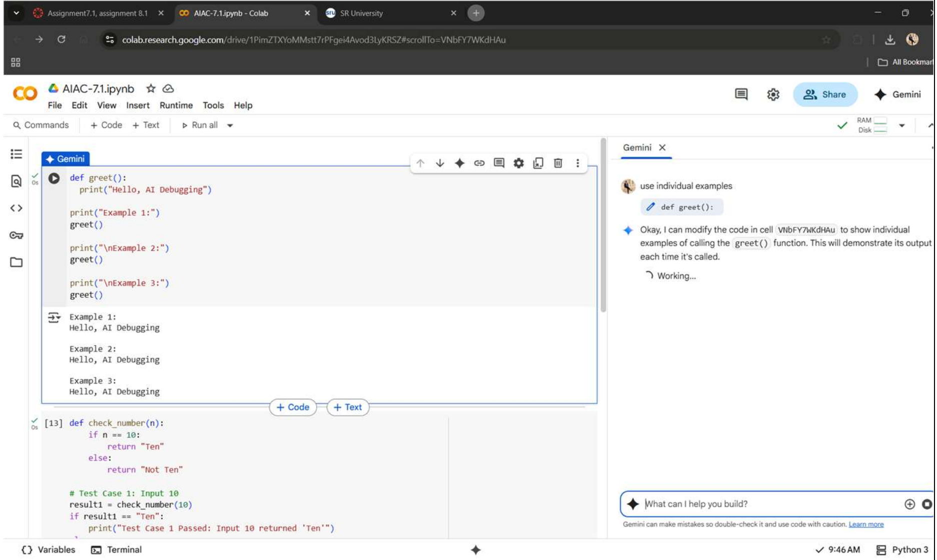
Please explain this error:

`SyntaxError: Missing parentheses in call to 'print'. Did yo…`

It seems there's a `SyntaxError` in the code because the `print` function is missing parentheses. In Python 3, `print` is a function and requires

AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Gemini
Next steps:
Explain error
Start coding or generate with AI.
Accept & Run
Accept
Cancel
Empty cell
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
9:28 AM
Python 3

OUTPUT:



```
def greet():
  print("Hello, AI Debugging")

print("Example 1:")
greet()

print("\nExample 2:")
greet()

print("\nExample 3:")
greet()
```

```
Example 1:
Hello, AI Debugging

Example 2:
Hello, AI Debugging

Example 3:
Hello, AI Debugging
```

```
[13] def check_number(n):
         if n == 10:
             return "Ten"
         else:
             return "Not Ten"

     # Test Case 1: Input 10
     result1 = check_number(10)
     if result1 == "Ten":
         print("Test Case 1 Passed: Input 10 returned 'Ten'")
```

use individual examples

`def greet():`

Okay, I can modify the code in cell `VNbFY7WKdHAu` to show individual examples of calling the `greet()` function. This will demonstrate its output each time it's called.

Working...

AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Gemini
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
9:46 AM
Python 3

Task Description #2 (Logic Error – Incorrect Condition in an If Statement)

Task: Supply a function where an if-condition mistakenly uses = instead of ==. Let AI identify and fix the issue.

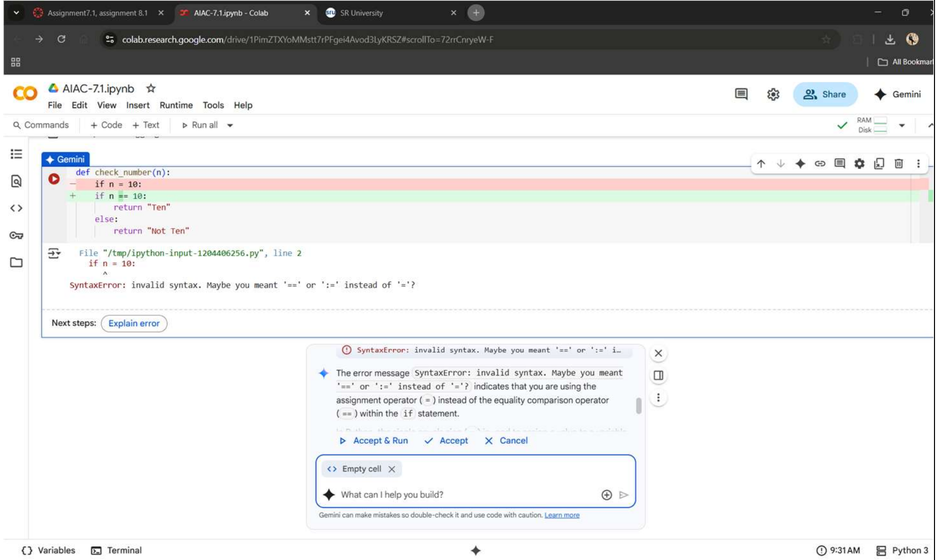
```
# Bug: Using assignment (=) instead of comparison (==)
    def check_number(n):
        if n = 10:
            return "Ten"
        else:
            return "Not Ten"
```

Requirements:

- Ask AI to explain why this causes a bug.
- Correct the code and verify with 3 assert test cases.

Expected Output #2:

- Corrected code using == with explanation and successful test execution.


Assignment7.1, assignment 8.1
AIAC-7.1.ipynb - Colab
SR University
colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=72rrCnryeW-F
All Bookmarks
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Gemini

```
def check_number(n):
-    if n = 10:
+    if n == 10:
        return "Ten"
    else:
        return "Not Ten"
```

```
File "/tmp/ipython-input-1204406256.py", line 2
    if n = 10:
    ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

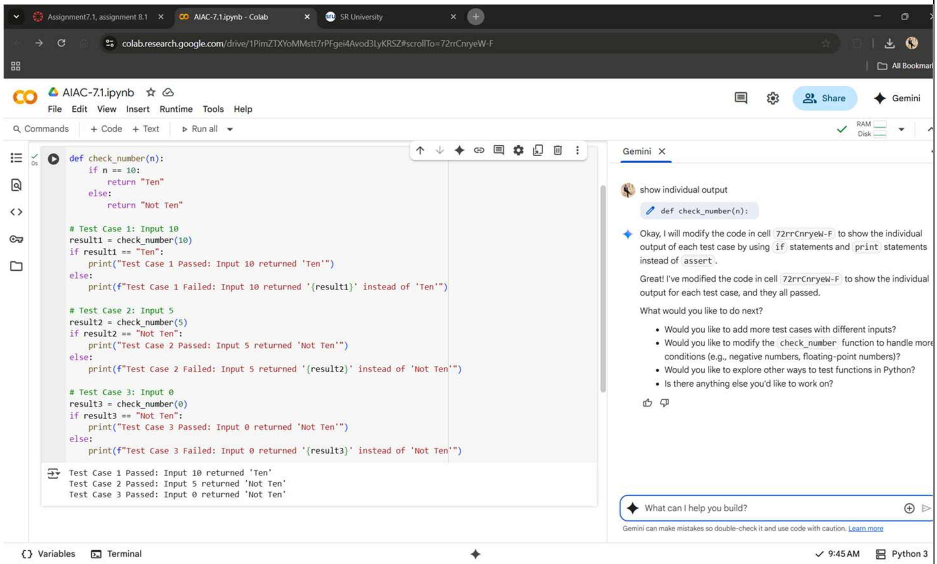
Next steps: Explain error

SyntaxError: invalid syntax. Maybe you meant '==' or ':=' i…

The error message `SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?` indicates that you are using the assignment operator (`=`) instead of the equality comparison operator (`==`) within the `if` statement.

Accept & Run
Accept
Cancel
Empty cell
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
9:31 AM
Python 3

OUTPUT:


Assignment7.1, assignment 8.1
AIAC-7.1.ipynb - Colab
SR University
colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=72rrCnryeW-F
All Bookmarks
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk

```
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"

# Test Case 1: Input 10
result1 = check_number(10)
if result1 == "Ten":
    print("Test Case 1 Passed: Input 10 returned 'Ten'")
else:
    print(f"Test Case 1 Failed: Input 10 returned '{result1}' instead of 'Ten'")

# Test Case 2: Input 5
result2 = check_number(5)
if result2 == "Not Ten":
    print("Test Case 2 Passed: Input 5 returned 'Not Ten'")
else:
    print(f"Test Case 2 Failed: Input 5 returned '{result2}' instead of 'Not Ten'")

# Test Case 3: Input 0
result3 = check_number(0)
if result3 == "Not Ten":
    print("Test Case 3 Passed: Input 0 returned 'Not Ten'")
else:
    print(f"Test Case 3 Failed: Input 0 returned '{result3}' instead of 'Not Ten'")
```

```
Test Case 1 Passed: Input 10 returned 'Ten'
Test Case 2 Passed: Input 5 returned 'Not Ten'
Test Case 3 Passed: Input 0 returned 'Not Ten'
```

Gemini

show individual output

`def check_number(n):`

Okay, I will modify the code in cell `72rrCnryeW-F` to show the individual output of each test case by using `if` statements and `print` statements instead of `assert`.

Great! I've modified the code in cell `72rrCnryeW-F` to show the individual output for each test case, and they all passed.

What would you like to do next?

- Would you like to add more test cases with different inputs?
- Would you like to modify the `check_number` function to handle more conditions (e.g., negative numbers, floating-point numbers)?
- Would you like to explore other ways to test functions in Python?
- Is there anything else you'd like to work on?

What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
9:45 AM
Python 3

Task Description #3 (Runtime Error – File Not Found)

Task: Provide code that attempts to open a non-existent file and crashes. Use AI to apply safe error handling.

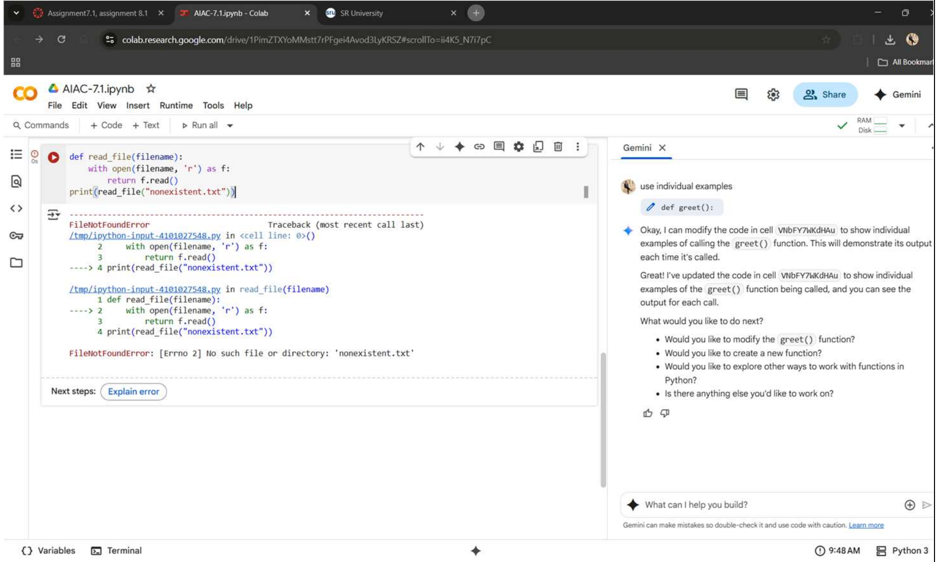
```
# Bug: Program crashes if file is missing
    def read_file(filename):
        with open(filename, 'r') as f:
            return f.read()
    print(read_file("nonexistent.txt"))
```

Requirements:

- Implement a try-except block suggested by AI.
- Add a user-friendly error message.
- Test with at least 3 scenarios: file exists, file missing, invalid path.

Expected Output #3:

- Safe file handling with exception management.



```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()
print(read_file("nonexistent.txt"))
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-4101027548.py in <cell line: 0>()
      2     with open(filename, 'r') as f:
      3         return f.read()
----> 4 print(read_file("nonexistent.txt"))

/tmp/ipython-input-4101027548.py in read_file(filename)
      1 def read_file(filename):
----> 2     with open(filename, 'r') as f:
      3         return f.read()
      4 print(read_file("nonexistent.txt"))

FileNotFoundError: [Errno 2] No such file or directory: 'nonexistent.txt'
```

Gemini

use individual examples

`def greet():`

Okay, I can modify the code in cell `VNbFY7WKdHAu` to show individual examples of calling the `greet()` function. This will demonstrate its output each time it's called.

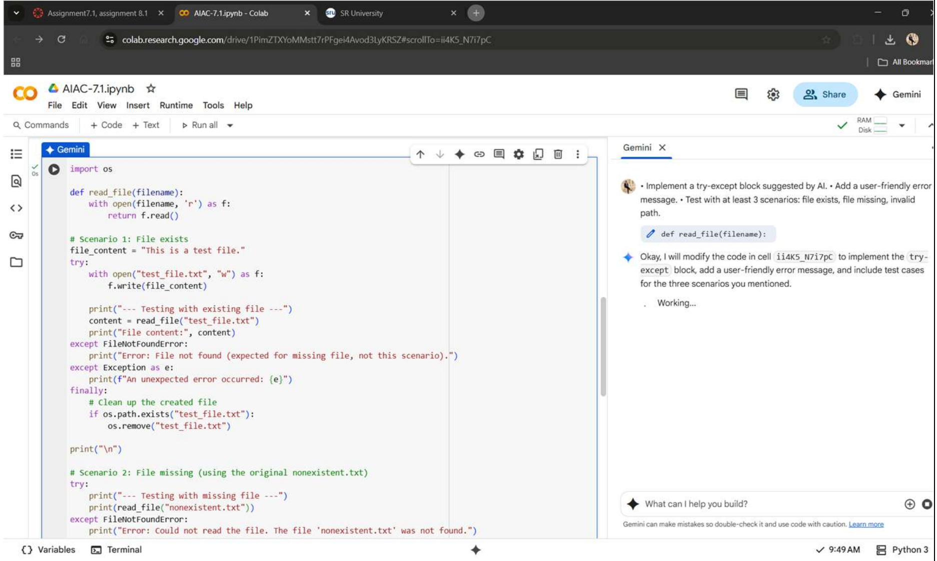
Great! I've updated the code in cell `VNbFY7WKdHAu` to show individual examples of the `greet()` function being called, and you can see the output for each call.

What would you like to do next?

- Would you like to modify the `greet()` function?
- Would you like to create a new function?
- Would you like to explore other ways to work with functions in Python?
- Is there anything else you'd like to work on?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more


Assignment7.1, assignment 8.1
AIAC-7.1.ipynb - Colab
SR University
colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=ii4K5_N7i7pC
All Bookmar
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Gemini
RAM
Disk
Variables
Terminal
9:49 AM
Python 3

```
import os

def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()

# Scenario 1: File exists
file_content = "This is a test file."
try:
    with open("test_file.txt", "w") as f:
        f.write(file_content)

    print("--- Testing with existing file ---")
    content = read_file("test_file.txt")
    print("File content:", content)
except FileNotFoundError:
    print("Error: File not found (expected for missing file, not this scenario).")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
finally:
    # Clean up the created file
    if os.path.exists("test_file.txt"):
        os.remove("test_file.txt")

print("\n")

# Scenario 2: File missing (using the original nonexistent.txt)
try:
    print("--- Testing with missing file ---")
    print(read_file("nonexistent.txt"))
except FileNotFoundError:
    print("Error: Could not read the file. The file 'nonexistent.txt' was not found.")
```

Gemini

• Implement a try-except block suggested by AI. • Add a user-friendly error message. • Test with at least 3 scenarios: file exists, file missing, invalid path.

`def read_file(filename):`

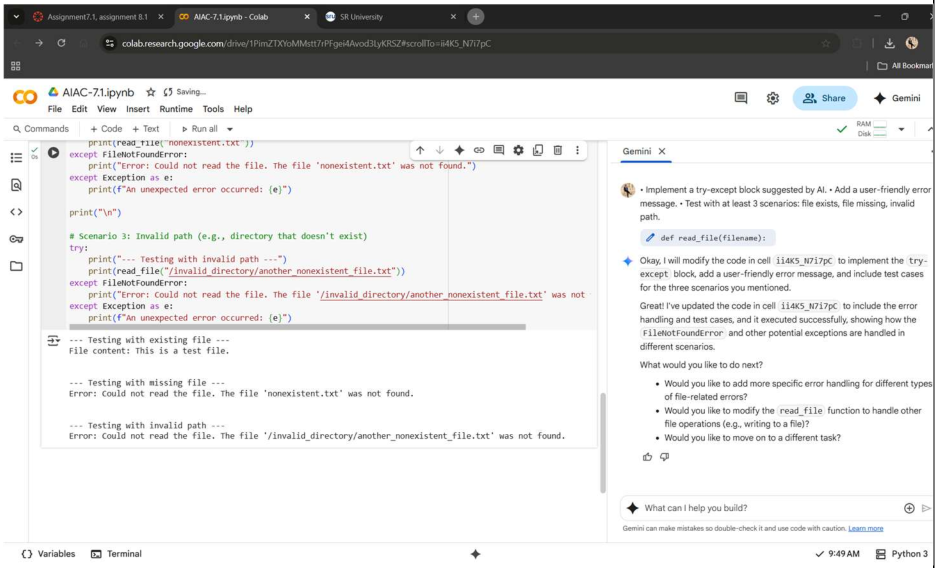
Okay, I will modify the code in cell `ii4K5_N7i7pC` to implement the `try-except` block, add a user-friendly error message, and include test cases for the three scenarios you mentioned.

Working...

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

OUTPUT:


Assignment7.1, assignment 8.1
AIAC-7.1.ipynb - Colab
SR University
colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=ii4K5_N7i7pC
All Bookmar
AIAC-7.1.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Gemini
RAM
Disk
Variables
Terminal
9:49 AM
Python 3

```
    print(read_file("nonexistent.txt"))
except FileNotFoundError:
    print("Error: Could not read the file. The file 'nonexistent.txt' was not found.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")

print("\n")

# Scenario 3: Invalid path (e.g., directory that doesn't exist)
try:
    print("--- Testing with invalid path ---")
    print(read_file("/invalid_directory/another_nonexistent_file.txt"))
except FileNotFoundError:
    print("Error: Could not read the file. The file '/invalid_directory/another_nonexistent_file.txt' was not
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
--- Testing with existing file ---
File content: This is a test file.


--- Testing with missing file ---
Error: Could not read the file. The file 'nonexistent.txt' was not found.


--- Testing with invalid path ---
Error: Could not read the file. The file '/invalid_directory/another_nonexistent_file.txt' was not found.
```

Gemini

• Implement a try-except block suggested by AI. • Add a user-friendly error message. • Test with at least 3 scenarios: file exists, file missing, invalid path.

`def read_file(filename):`

Okay, I will modify the code in cell `ii4K5_N7i7pC` to implement the `try-except` block, add a user-friendly error message, and include test cases for the three scenarios you mentioned.

Great! I've updated the code in cell `ii4K5_N7i7pC` to include the error handling and test cases, and it executed successfully, showing how the `FileNotFoundError` and other potential exceptions are handled in different scenarios.

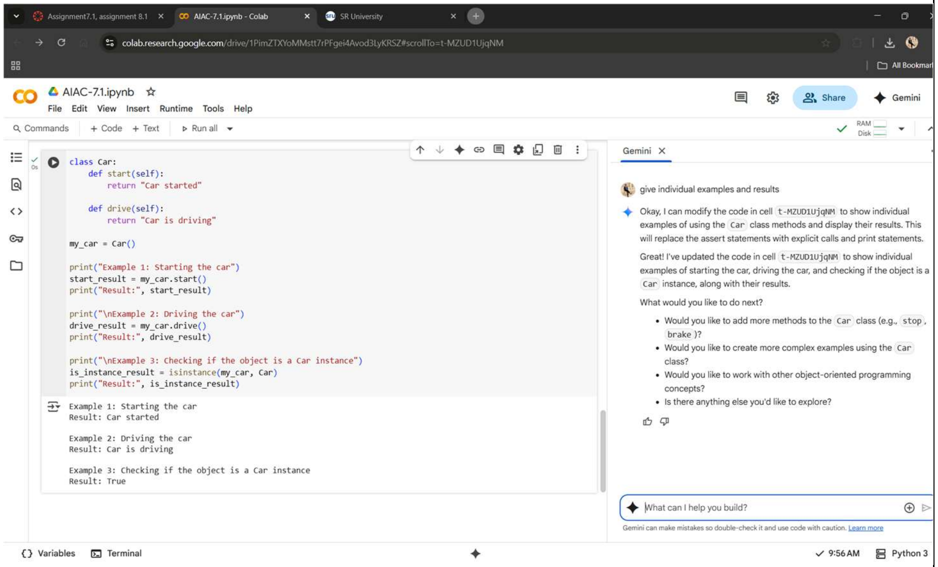
What would you like to do next?

- Would you like to add more specific error handling for different types of file-related errors?
- Would you like to modify the `read_file` function to handle other file operations (e.g., writing to a file)?
- Would you like to move on to a different task?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

OUTPUT:


Assignment7.1, assignment 8.1
AIAC-7.1.ipynb - Colab
SR University
colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=t-MZUD1UjqNM
All Bookmar
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Gemini
RAM
Disk
Variables
Terminal
9:56 AM
Python 3

```
class Car:
    def start(self):
        return "Car started"

    def drive(self):
        return "Car is driving"

my_car = Car()

print("Example 1: Starting the car")
start_result = my_car.start()
print("Result:", start_result)

print("\nExample 2: Driving the car")
drive_result = my_car.drive()
print("Result:", drive_result)

print("\nExample 3: Checking if the object is a Car instance")
is_instance_result = isinstance(my_car, Car)
print("Result:", is_instance_result)
```

```
Example 1: Starting the car
Result: Car started

Example 2: Driving the car
Result: Car is driving

Example 3: Checking if the object is a Car instance
Result: True
```

Gemini

give individual examples and results

Okay, I can modify the code in cell `t-MZUD1UjqNM` to show individual examples of using the `Car` class methods and display their results. This will replace the assert statements with explicit calls and print statements.

Great! I've updated the code in cell `t-MZUD1UjqNM` to show individual examples of starting the car, driving the car, and checking if the object is a `Car` instance, along with their results.

What would you like to do next?

- Would you like to add more methods to the `Car` class (e.g., `stop`, `brake`)?
- Would you like to create more complex examples using the `Car` class?
- Would you like to work with other object-oriented programming concepts?
- Is there anything else you'd like to explore?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

Task Description #4 (AttributeError – Calling a Non-Existent Method)

Task: Give a class where a non-existent method is called (e.g., obj.undefined_method()). Use AI to debug and fix.

```
# Bug: Calling an undefined method
    class Car:
        def start(self):
            return "Car started"
    my_car = Car()
    print(my_car.drive())  # drive() is not defined
```

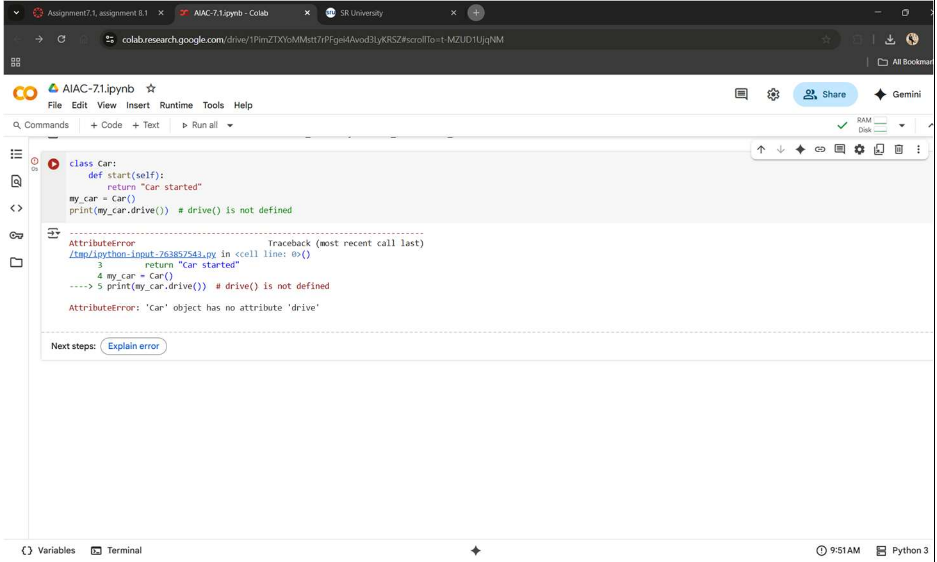
Requirements:

- Students must analyze whether to define the missing method or correct the method call.

- Use 3 assert tests to confirm the corrected class works.

Expected Output #4:

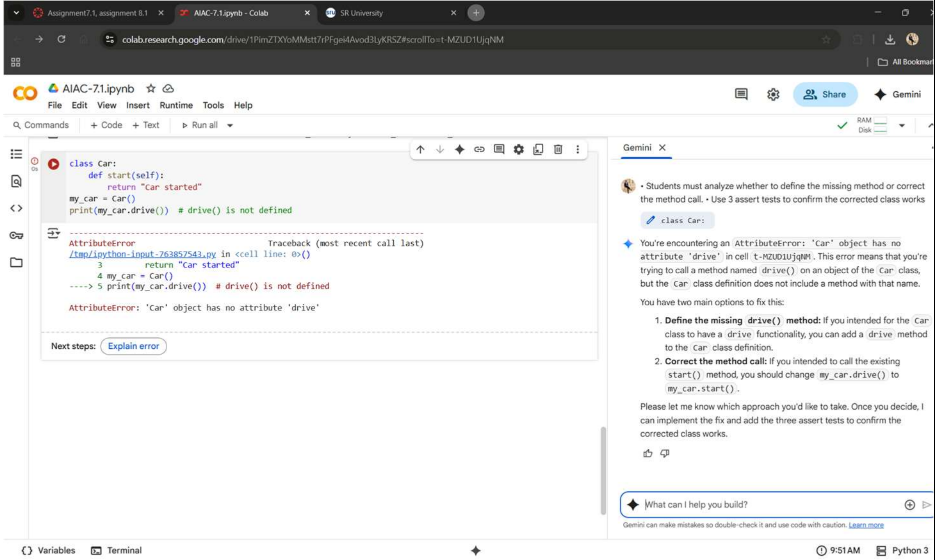
- Corrected class with clear AI explanation.



```
class Car:
    def start(self):
        return "Car started"
my_car = Car()
print(my_car.drive())  # drive() is not defined
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-763857543.py in <cell line: 0>()
      3         return "Car started"
      4 my_car = Car()
----> 5 print(my_car.drive())  # drive() is not defined

AttributeError: 'Car' object has no attribute 'drive'
```



```
class Car:
    def start(self):
        return "Car started"
my_car = Car()
print(my_car.drive())  # drive() is not defined
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipython-input-763857543.py in <cell line: 0>()
      3         return "Car started"
      4 my_car = Car()
----> 5 print(my_car.drive())  # drive() is not defined

AttributeError: 'Car' object has no attribute 'drive'
```

• Students must analyze whether to define the missing method or correct the method call. • Use 3 assert tests to confirm the corrected class works

`class Car:`

You're encountering an `AttributeError: 'Car' object has no attribute 'drive'` in cell `t-MZUD1UjqNM`. This error means that you're trying to call a method named `drive()` on an object of the `Car` class, but the `Car` class definition does not include a method with that name.

You have two main options to fix this:

1. **Define the missing `drive()` method:** If you intended for the `Car` class to have a `drive` functionality, you can add a `drive` method to the `Car` class definition.
2. **Correct the method call:** If you intended to call the existing `start()` method, you should change `my_car.drive()` to `my_car.start()`.

Please let me know which approach you'd like to take. Once you decide, I can implement the fix and add the three assert tests to confirm the corrected class works.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

Task Description #5 (TypeError – Mixing Strings and Integers in Addition)

Task: Provide code that adds an integer and string ("5" + 2) causing a TypeError. Use AI to resolve the bug.

```
# Bug: TypeError due to mixing string and integer
    def add_five(value):
        return value + 5
```

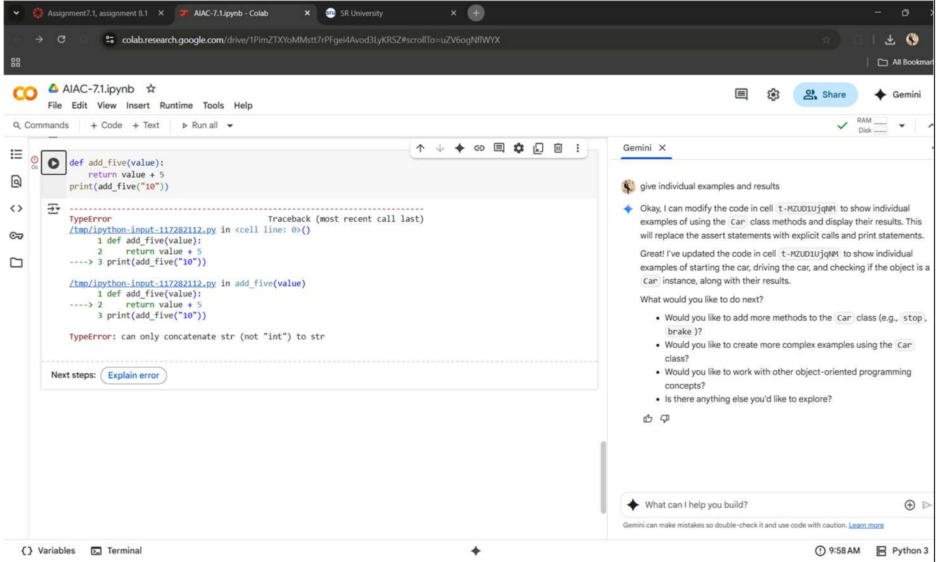
```
print(add_five("10"))
```

Requirements:

- Ask AI for two solutions: type casting and string concatenation.
- Validate with 3 assert test cases.

Expected Output #5:

- Corrected code that runs successfully for multiple inputs.


colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=uZV6ogNflWYX
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Next steps: Explain error
Variables
Terminal
9:58 AM
Python 3

```
def add_five(value):
    return value + 5
print(add_five("10"))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-117282112.py in <cell line: 0>()
      1 def add_five(value):
      2     return value + 5
----> 3 print(add_five("10"))

/tmp/ipython-input-117282112.py in add_five(value)
      1 def add_five(value):
----> 2     return value + 5
      3 print(add_five("10"))

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

give individual examples and results

Okay, I can modify the code in cell `t-MZUD1UjqNM` to show individual examples of using the `Car` class methods and display their results. This will replace the assert statements with explicit calls and print statements.

Great! I've updated the code in cell `t-MZUD1UjqNM` to show individual examples of starting the car, driving the car, and checking if the object is a `Car` instance, along with their results.

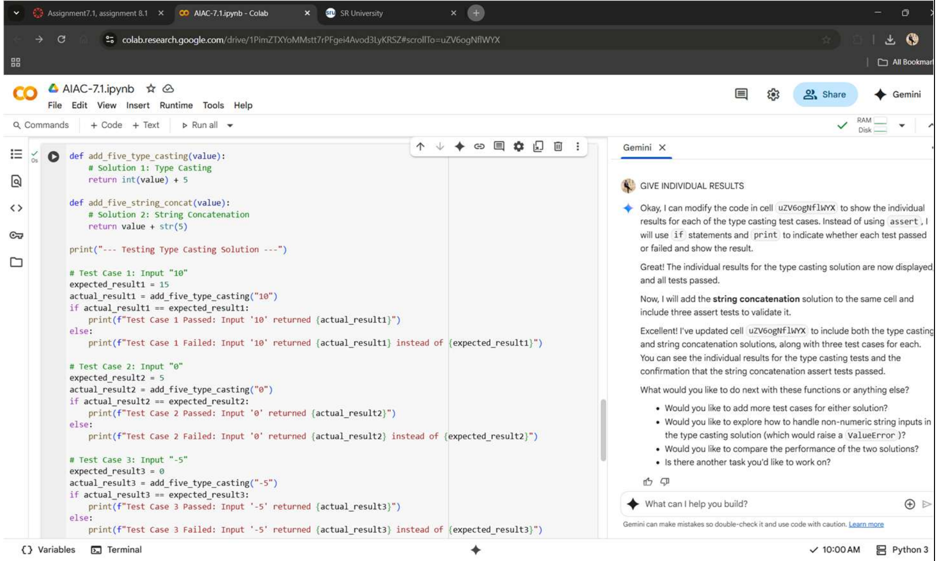
What would you like to do next?

- Would you like to add more methods to the `Car` class (e.g., `stop`, `brake`)?
- Would you like to create more complex examples using the `Car` class?
- Would you like to work with other object-oriented programming concepts?
- Is there anything else you'd like to explore?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

OUTPUT:


colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=uZV6ogNflWYX
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Variables
Terminal
10:00 AM
Python 3

```
def add_five_type_casting(value):
    # Solution 1: Type Casting
    return int(value) + 5

def add_five_string_concat(value):
    # Solution 2: String Concatenation
    return value + str(5)

print("--- Testing Type Casting Solution ---")

# Test Case 1: Input "10"
expected_result1 = 15
actual_result1 = add_five_type_casting("10")
if actual_result1 == expected_result1:
    print(f"Test Case 1 Passed: Input '10' returned {actual_result1}")
else:
    print(f"Test Case 1 Failed: Input '10' returned {actual_result1} instead of {expected_result1}")

# Test Case 2: Input "0"
expected_result2 = 5
actual_result2 = add_five_type_casting("0")
if actual_result2 == expected_result2:
    print(f"Test Case 2 Passed: Input '0' returned {actual_result2}")
else:
    print(f"Test Case 2 Failed: Input '0' returned {actual_result2} instead of {expected_result2}")

# Test Case 3: Input "-5"
expected_result3 = 0
actual_result3 = add_five_type_casting("-5")
if actual_result3 == expected_result3:
    print(f"Test Case 3 Passed: Input '-5' returned {actual_result3}")
else:
    print(f"Test Case 3 Failed: Input '-5' returned {actual_result3} instead of {expected_result3}")
```

GIVE INDIVIDUAL RESULTS

Okay, I can modify the code in cell `uZV6ogNflWYX` to show the individual results for each of the type casting test cases. Instead of using `assert`, I will use `if` statements and `print` to indicate whether each test passed or failed and show the result.

Great! The individual results for the type casting solution are now displayed and all tests passed.

Now, I will add the **string concatenation** solution to the same cell and include three assert tests to validate it.

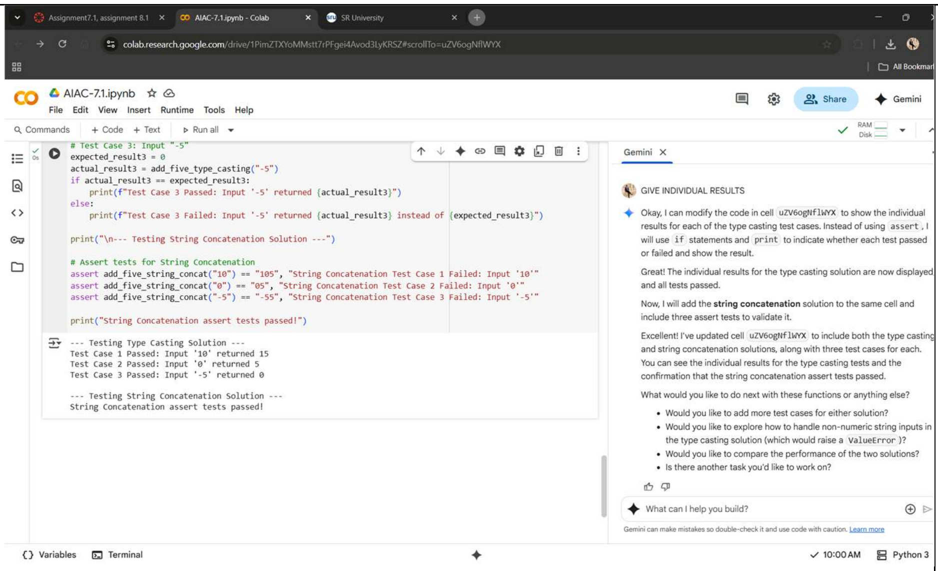
Excellent! I've updated cell `uZV6ogNflWYX` to include both the type casting and string concatenation solutions, along with three test cases for each. You can see the individual results for the type casting tests and the confirmation that the string concatenation assert tests passed.

What would you like to do next with these functions or anything else?

- Would you like to add more test cases for either solution?
- Would you like to explore how to handle non-numeric string inputs in the type casting solution (which would raise a `ValueError`)?
- Would you like to compare the performance of the two solutions?
- Is there another task you'd like to work on?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more


Assignment7.1, assignment 8.1
AIAC-7.1.ipynb - Colab
SR University
colab.research.google.com/drive/1PimZTXYoMMstt7rPFgei4Avod3LyKRSZ#scrollTo=uZV6ogNflWYX
All Bookmar
AIAC-7.1.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Gemini
RAM
Disk
Variables
Terminal
10:00 AM
Python 3

```python
# Test Case 3: Input "-5"
expected_result3 = 0
actual_result3 = add_five_type_casting("-5")
if actual_result3 == expected_result3:
    print(f"Test Case 3 Passed: Input '-5' returned {actual_result3}")
else:
    print(f"Test Case 3 Failed: Input '-5' returned {actual_result3} instead of {expected_result3}")

print("\n--- Testing String Concatenation Solution ---")

# Assert tests for String Concatenation
assert add_five_string_concat("10") == "105", "String Concatenation Test Case 1 Failed: Input '10'"
assert add_five_string_concat("0") == "05", "String Concatenation Test Case 2 Failed: Input '0'"
assert add_five_string_concat("-5") == "-55", "String Concatenation Test Case 3 Failed: Input '-5'"

print("String Concatenation assert tests passed!")
```

```
--- Testing Type Casting Solution ---
Test Case 1 Passed: Input '10' returned 15
Test Case 2 Passed: Input '0' returned 5
Test Case 3 Passed: Input '-5' returned 0

--- Testing String Concatenation Solution ---
String Concatenation assert tests passed!
```

Gemini

GIVE INDIVIDUAL RESULTS

Okay, I can modify the code in cell `uZV6ogNflWYX` to show the individual results for each of the type casting test cases. Instead of using `assert`, I will use `if` statements and `print` to indicate whether each test passed or failed and show the result.

Great! The individual results for the type casting solution are now displayed and all tests passed.

Now, I will add the **string concatenation** solution to the same cell and include three assert tests to validate it.

Excellent! I've updated cell `uZV6ogNflWYX` to include both the type casting and string concatenation solutions, along with three test cases for each. You can see the individual results for the type casting tests and the confirmation that the string concatenation assert tests passed.

What would you like to do next with these functions or anything else?

- Would you like to add more test cases for either solution?
- Would you like to explore how to handle non-numeric string inputs in the type casting solution (which would raise a `ValueError`)?
- Would you like to compare the performance of the two solutions?
- Is there another task you'd like to work on?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Identification of bugs	0.5
Application of AI-suggested fixes	0.5
Explanation and understanding of errors	0.5
Corrected code functionality	0.5
Report structure and reflection	0.5
Total	2.5 Marks