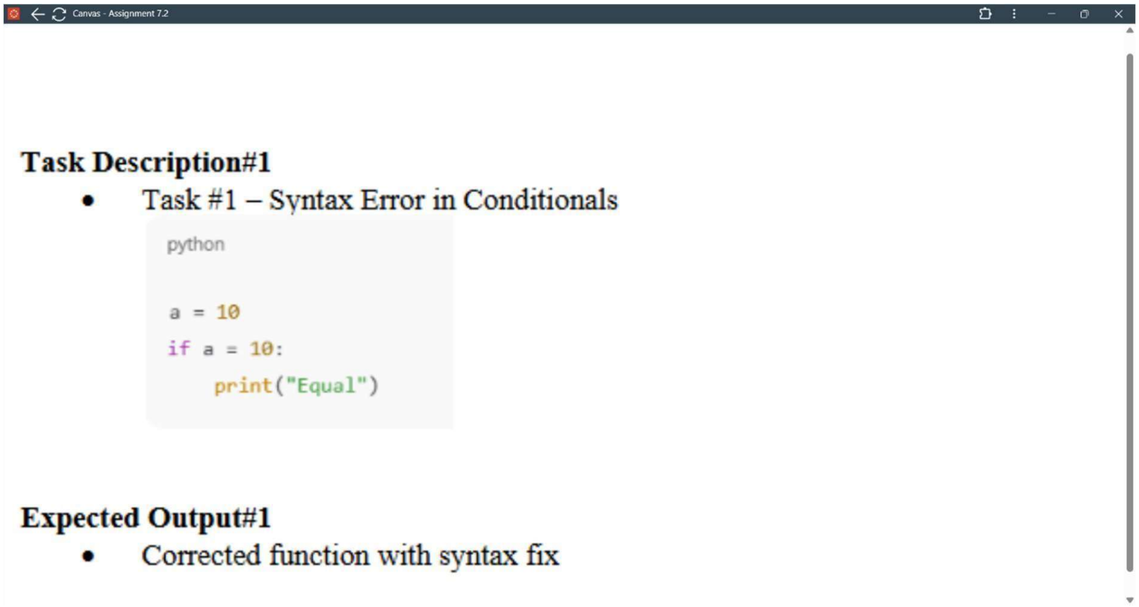
ASSIGNMENT – 7.2

AI ASSISTED CODING

NAME : S RUCHITHA
HALLTICKECT NO : 2403A52316
BATCH NUMBER : 01
COURSE CODE : 24CS002PC215
PROGRAM NAME : B.TECH
YEAR/SEM : 2ND AND 3RD

TASK - 1



Task Description#1

- Task #1 – Syntax Error in Conditionals

```python
a = 10
if a = 10:
    print("Equal")
```

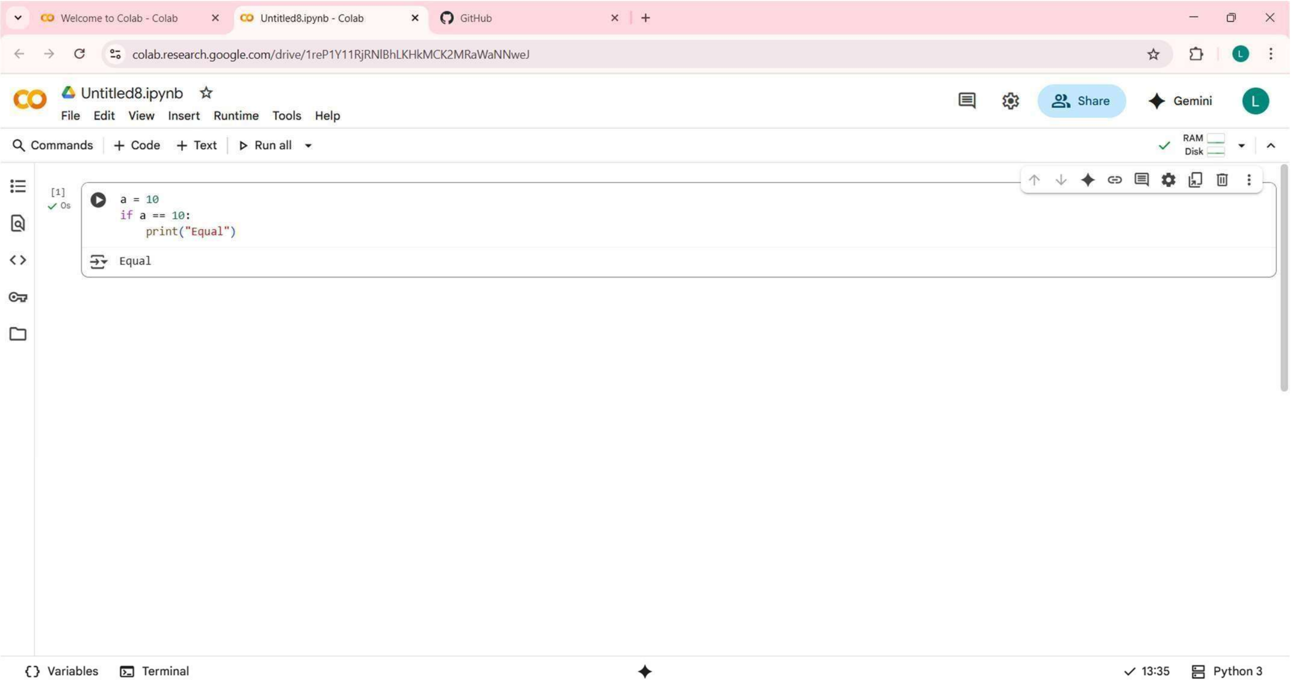
Expected Output#1

- Corrected function with syntax fix

PROMT :

"Write a Python code snippet that assigns the value 10 to a variable a and then prints 'Equal' if a is equal to 10."

CODE :



```
a = 10
if a == 10:
    print("Equal")
```

```
Equal
```

OUTPUT :

“Equal”

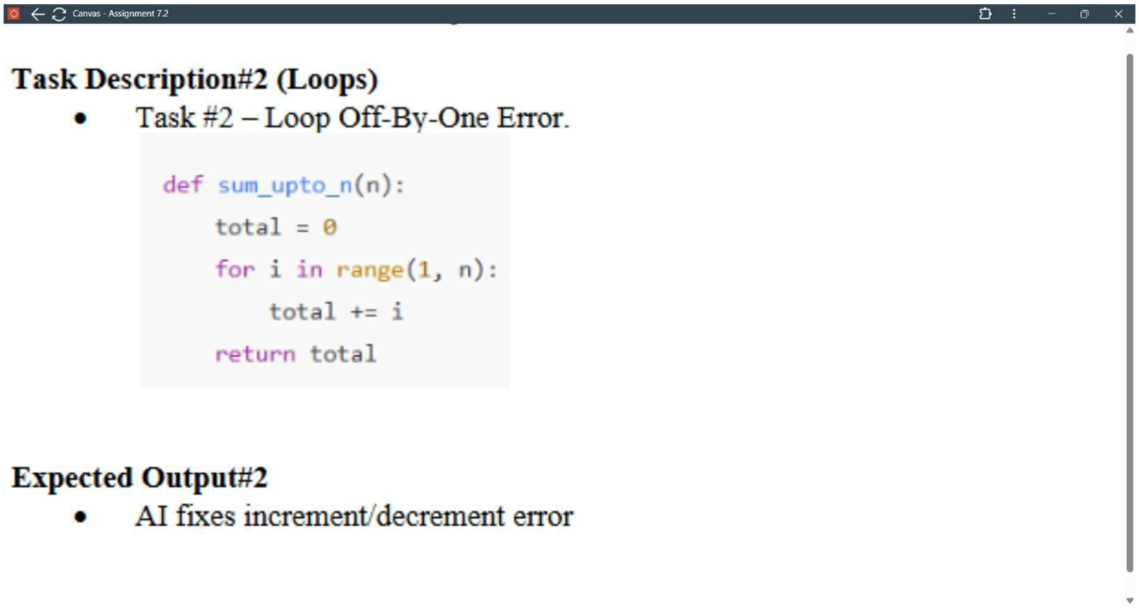
EXPLANATION :

This code assigns the value 10 to a variable named a. It then uses an if statement to check if the value of a is equal to 10. If the condition is true, it prints the word "Equal".

OBSERVATION :

- It demonstrates a basic variable assignment.
- It shows a simple conditional statement (if).
- The code compares the value of a variable to a literal value.
- It uses the equality operator (==).
- The print() function is used for output.
- The code will always print "Equal" because the condition is always true with the given value of a.

TASK – 2



Task Description#2 (Loops)

- Task #2 – Loop Off-By-One Error.

```
def sum_upto_n(n):
    total = 0
    for i in range(1, n):
        total += i
    return total
```

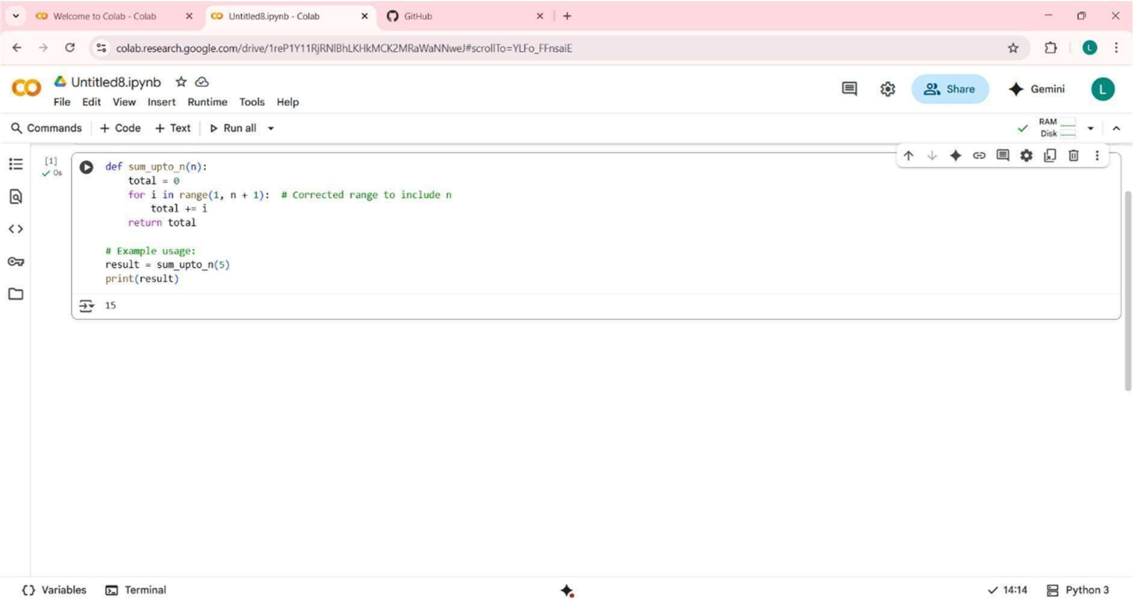
Expected Output#2

- AI fixes increment/decrement error

PROMT :

"Write a Python function called sum_upto_n that takes an integer n as input and returns the sum of all integers from 1 to n. Include an example of how to use the function and print the result for n=5."

CODE :



```
def sum_upto_n(n):
    total = 0
    for i in range(1, n + 1):  # Corrected range to include n
        total += i
    return total

# Example usage:
result = sum_upto_n(5)
print(result)
```

```
15
```

OUTPUT:

```
15
```

PROMT :

"Write a Python function called sum_upto_n that takes an integer n as input and returns the sum of all integers from 1 to n. Include an example of how to use the function and print the result for n=5."

EXPLANATION :

This code defines a Python function called sum_upto_n that calculates the sum of all integers from 1 up to a given number n.

1. def sum_upto_n(n):: This line defines the function named sum_upto_n that accepts one argument, n.
2. total = 0: Inside the function, a variable total is initialized to 0. This variable will accumulate the sum.
3. for i in range(1, n + 1):: This is a for loop that iterates through a sequence of numbers. range(1, n + 1) generates numbers starting

   from 1 and going up to n (inclusive). This is the corrected part to fix the "off-by-one" error, ensuring n is included in the sum.
4. total += i: In each iteration, the current number i is added to the total.

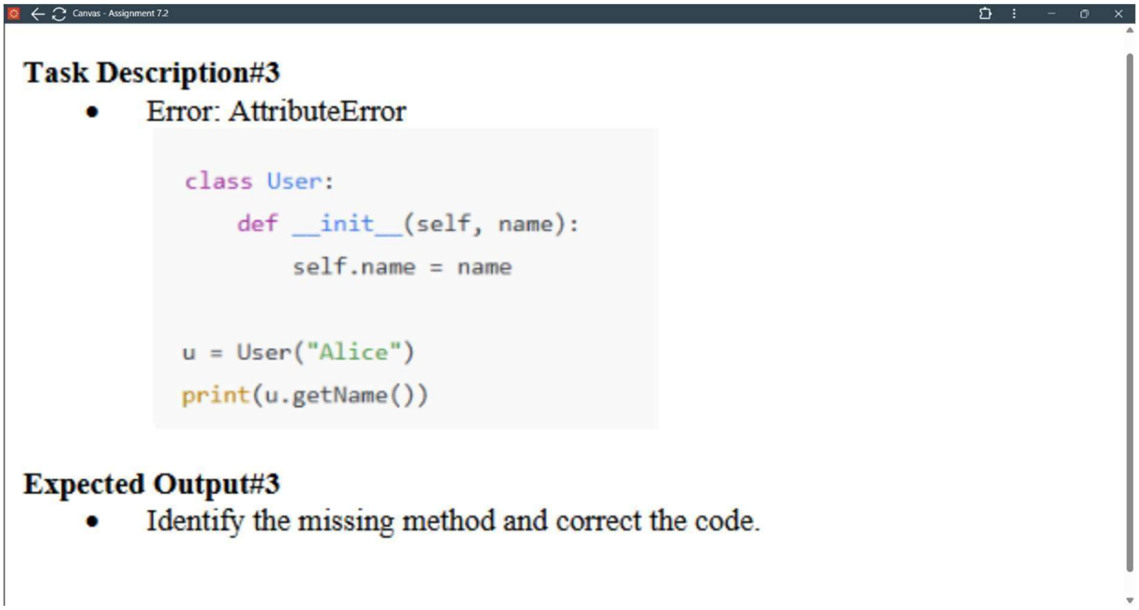
5. return total: After the loop finishes, the function returns the final calculated total.
6. result = sum_upto_n(5): This line calls the sum_upto_n function with 5 as the argument and stores the returned sum in the result variable.
7. print(result): This line prints the value stored in the result variable to the console.

When this code is executed, it will calculate the sum of 1 + 2 + 3 + 4 + 5, which is 15, and print 15.

OBSERVATION :

- It defines a reusable function to calculate a sum.
- It uses a for loop to iterate through a range of numbers.
- The range() function is used to generate the sequence of numbers.
- The + 1 in range(1, n + 1) is crucial for including the number n in the sum, fixing a potential off-by-one error if it were range(1, n).
- The code accumulates the sum in the total variable.
- It demonstrates how to return a value from a function.
- There is an example usage of the function provided.
- The print() function is used to display the final result.
- This function calculates the sum of an arithmetic series.
- It's a common pattern for performing iterative calculations.

TASK -3



Task Description#3

- Error: AttributeError

```
class User:
    def __init__(self, name):
        self.name = name

u = User("Alice")
print(u.getName())
```

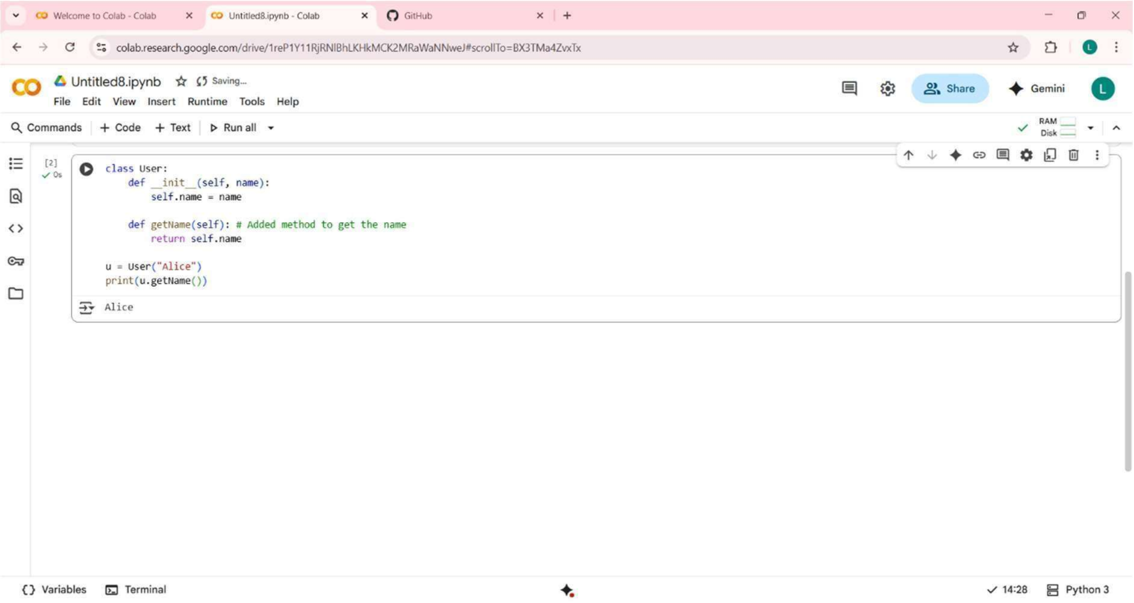
Expected Output#3

- Identify the missing method and correct the code.

PROMT :

"Write a Python class named User with a constructor that takes a name as an argument and stores it as an attribute. Add a method called getName that returns the user's name. Then, create an instance of the User class with the name 'Alice' and print the name using the getName method."

CODE :



```python
class User:
    def __init__(self, name):
        self.name = name

    def getName(self): # Added method to get the name
        return self.name

u = User("Alice")
print(u.getName())
```

```
Alice
```

OUTPUT :

Alice

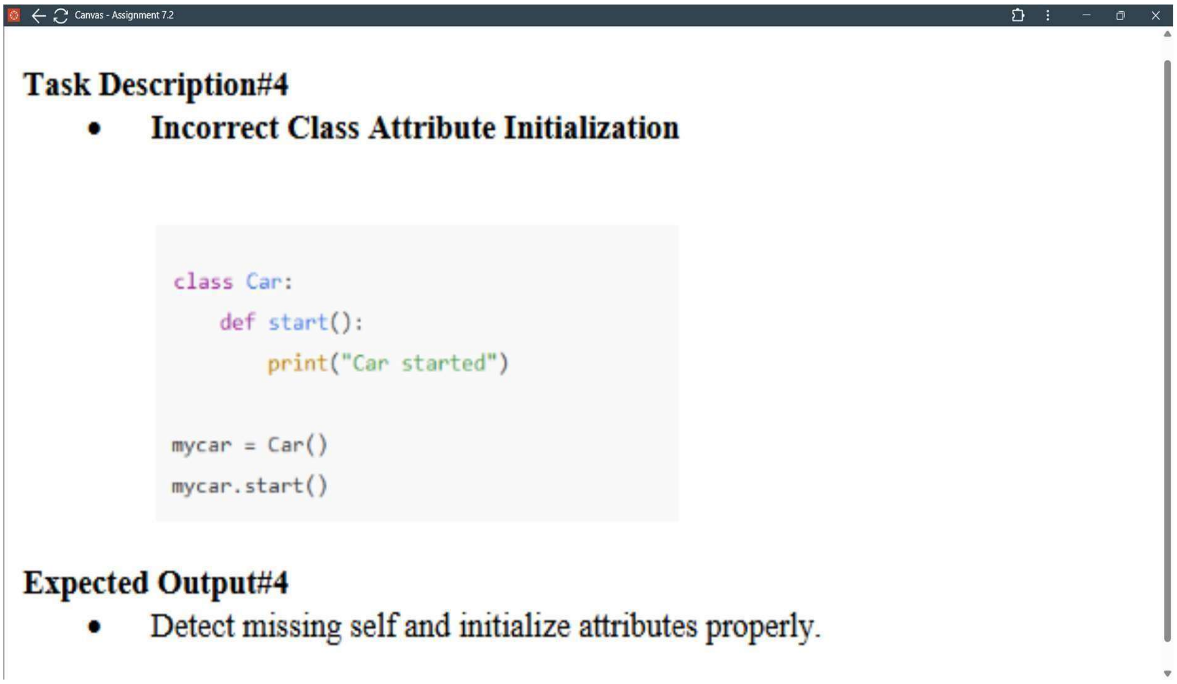
EXPLANATION :

This code defines a User class with a constructor to set a name attribute. It includes a getName method to return the stored name. An instance of User named u is created with "Alice". Finally, it prints the name using u.getName().

OBSERVATION :

- It defines a class, a fundamental concept in object-oriented programming.
- It demonstrates how to pass arguments when creating an object.
- Attributes (self.name) are used to store data within an object.
- Methods (getName) define the behaviors of an object.
- The self parameter is used to refer to the instance of the class.
- Objects are created from classes (u = User("Alice")).
- Methods are called on objects using dot notation (u.getName()).

TASK – 4



Task Description#4

- **Incorrect Class Attribute Initialization**

```python
class Car:
    def start():
        print("Car started")

mycar = Car()
mycar.start()
```

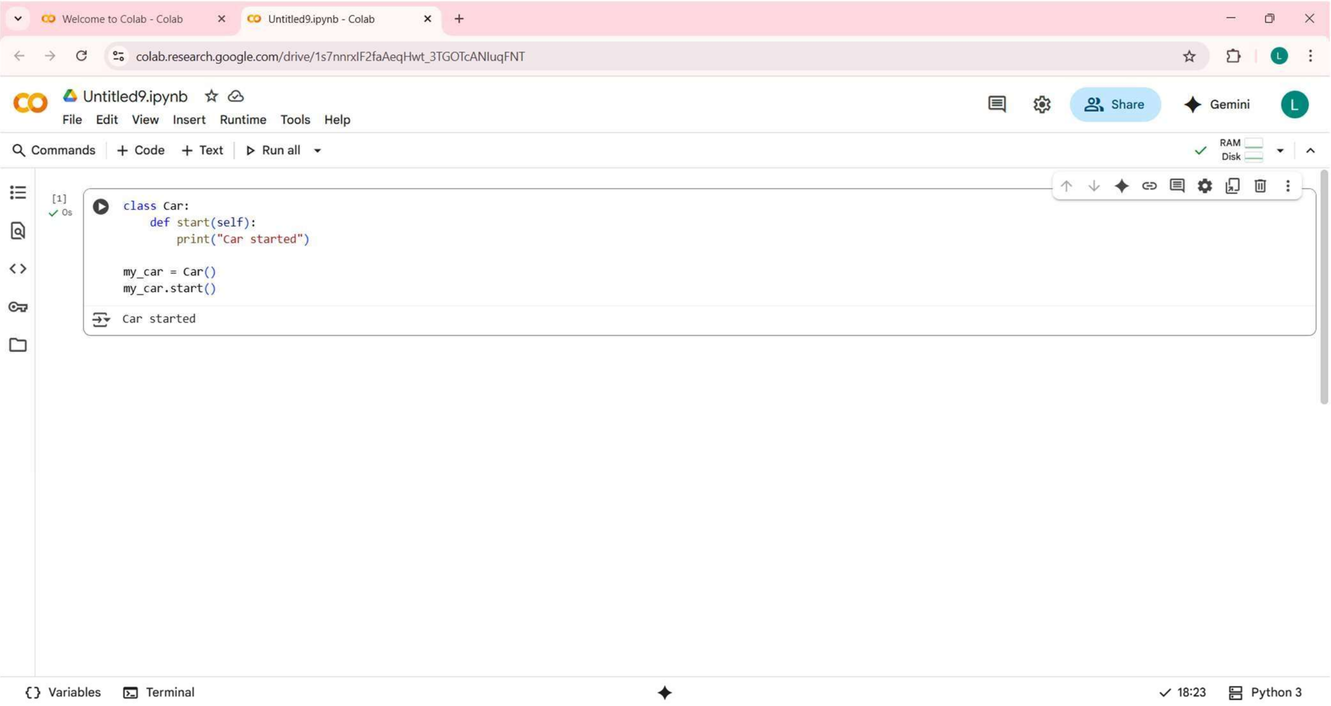
Expected Output#4

- Detect missing self and initialize attributes properly.

PROMT :

"Write Python code for a simple class named Car. The class should have a method called start that prints 'Car started' when called. After defining the class, create an instance of the Car class and call the start method."

CODE :


Welcome to Colab - Colab
Untitled9.ipynb - Colab
colab.research.google.com/drive/1s7nnrxlF2faAeqHwt_3TGOTcANluqFNT
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
[1]
0s
Variables
Terminal
18:23
Python 3

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

my_car = Car()
my_car.start()
```

```
Car started
```

OUTPUT :

CAR STARTED

EXPLANATION :

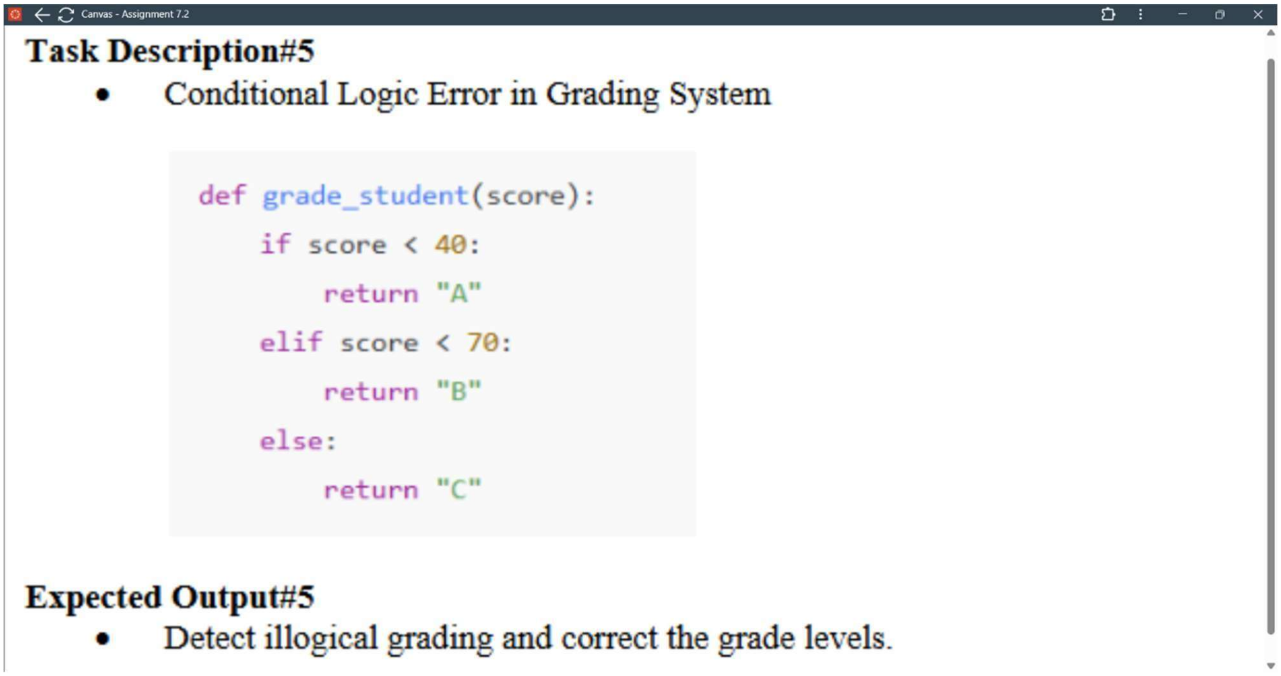
This Python code defines a class called Car. Inside the class, there's a method named start that simply prints "Car started". An object named my_car is created from the Car class. Finally, the start method of the my_car object is called, which outputs "Car started". This is a basic example of object-oriented programming in Python.

OBSERVATION :

Here are some observations about this code:

It defines a basic Python class named Car.The Car class has one method, start. The start method has a self parameter, as is standard for instance methods. Calling start prints the string "Car started" to the console. An object my_car is created from the Car class.

TASK – 5



Task Description#5

- Conditional Logic Error in Grading System

```
def grade_student(score):
    if score < 40:
        return "A"
    elif score < 70:
        return "B"
    else:
        return "C"
```

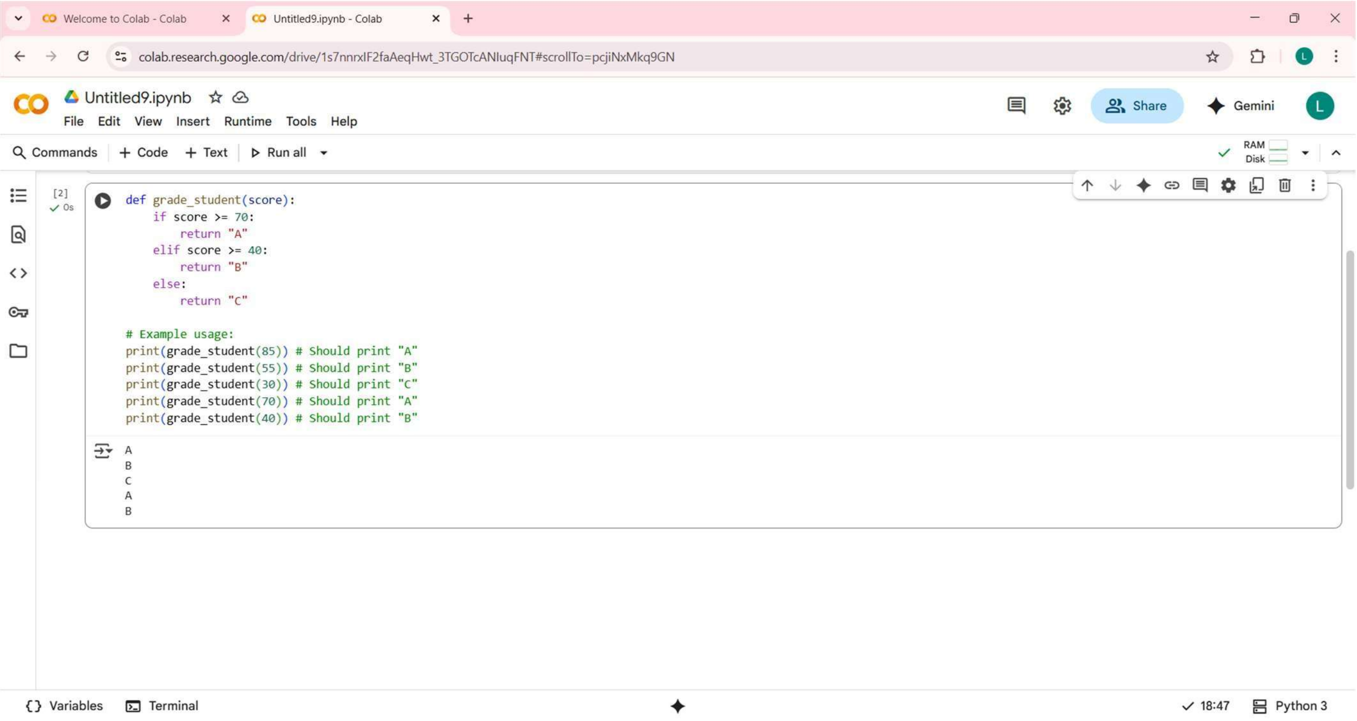
Expected Output#5

- Detect illogical grading and correct the grade levels.

PROMT :

"Write a Python function called grade_student that takes a student's score as input. The function should return 'A' if the score is 70 or above, 'B' if the score is 40 or above but less than 70, and 'C' if the score is below 40. Include example calls to test the function with different scores."

CODE :



```python
def grade_student(score):
    if score >= 70:
        return "A"
    elif score >= 40:
        return "B"
    else:
        return "C"

# Example usage:
print(grade_student(85)) # Should print "A"
print(grade_student(55)) # Should print "B"
print(grade_student(30)) # Should print "C"
print(grade_student(70)) # Should print "A"
print(grade_student(40)) # Should print "B"
```

```
A
B
C
A
B
```

OUTPUT :

A

B

C

A

B

EXPLANATION :

This Python code defines a function grade_student that assigns a letter grade based on a numerical score. It checks if the score is 70 or higher for an 'A', 40 or higher for a 'B', and otherwise assigns a 'C'. The lines below the function demonstrate how to call it with different scores and print the results.

OBSERVATION :

Here are some observations about the grade_student function:

- It uses if, elif, and else statements to implement conditional logic.
- The conditions are checked in order: >= 70, then >= 40.
- The else block handles any score less than 40.
- The function returns a string representing the grade.
- Example calls demonstrate the function's behavior for different score ranges.
- The output shows that the function correctly assigns grades A, B, and C based on the defined criteria.
- It's a clear and concise example of using conditional statements to categorize input.