LAB ASSIGNMENT – 7.2

NAME : V.CHANDANA

ROLL NO : 2403A54103

COURSE : ASSISTED CODING

BATCH : 01

QUESTION

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

Task Description#2 (Loops)

- Task #2 – Loop Off-By-One Error.

```python
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

Task Description#3

- Error: AttributeError

```
class User:
```

Expected Output#2
- AI fixes increment/decrement error

Task Description#3
- Error: AttributeError

```python
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.

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.

Task Description#5

- Conditional Logic Error in Grading System

```python
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.

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

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

Expected Output#4

- Detect missing self and initialize attributes properly.

Task Description#5

- Conditional Logic Error in Grading System

```python
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.

TASK 1

PROMPT

Write a Python program that assigns a value of 10 to the variable a, then checks if a is equal to 10. If the condition is true, the program should print the word 'equal'

EXPLAINATION

1. a = 10: This line assigns the integer value 10 to the variable named a.

2. if a == 10:: This is a conditional statement. It checks if the value of the variable a is equal to 10.

3. print('equal'): This line will only execute if the condition in the if statement is true. In this case, since a is 10, the condition is true, and the word 'equal' is printed to the output.

Since a was assigned the value 10, the condition a == 10 is true, and the program outputs equal.
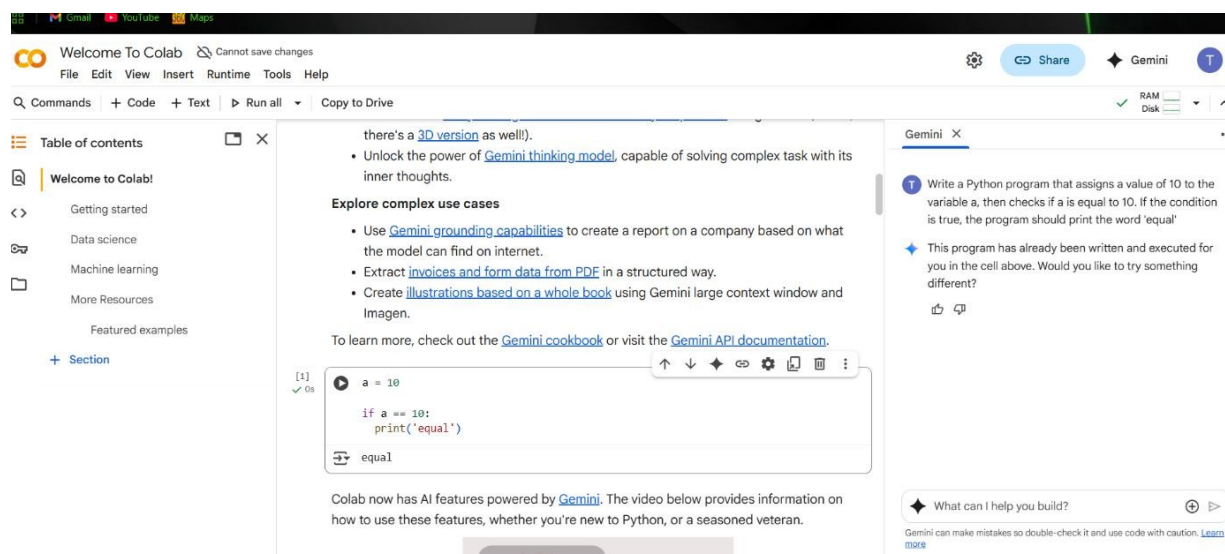

TASK 2

PROMPT

Write a Python function named sum_upto_n(n) that calculates and returns the sum of all natural numbers from 1 up to but not including n.

EXPLAINATION

def sum_upto_n(n): This line defines a function named sum_upto_n that accepts one argument, n.

The text within the triple quotes """...""" is a docstring, which explains what the function does, its arguments (Args), and what it returns (Returns).

total_sum = 0: This line initializes a variable called total_sum to 0.
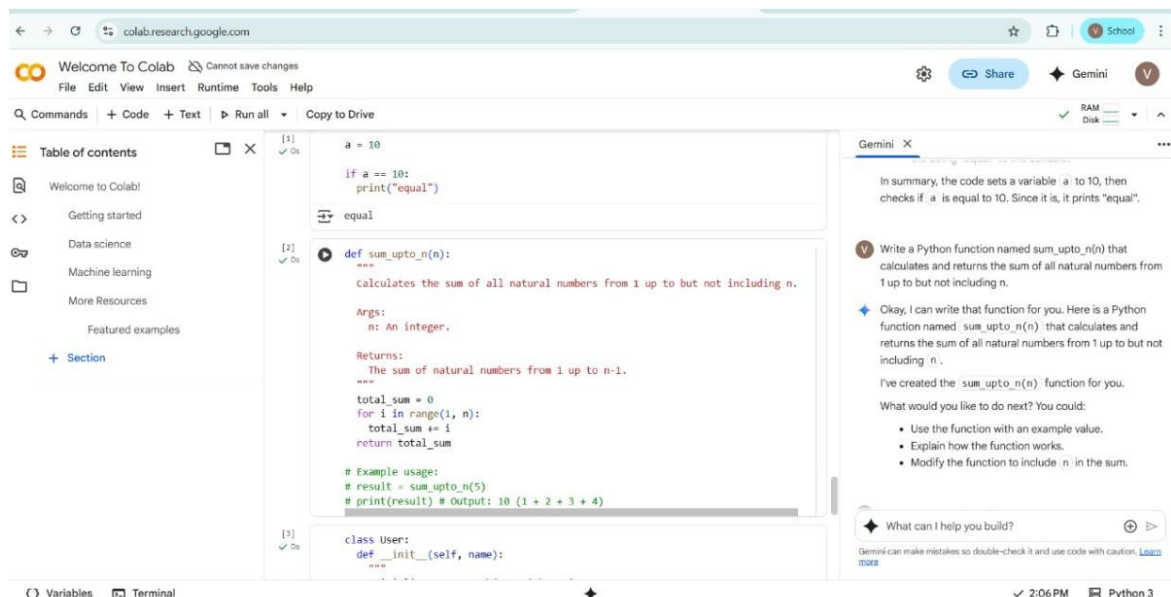
TASK 3

PROMPT:

Write a Python class called User with an __init__ method that takes a name as a parameter and stores it. Add a method getName that returns the user's name. Create an instance of the class with the name "Alice" and print the result of calling getName().

EXPLAINATION
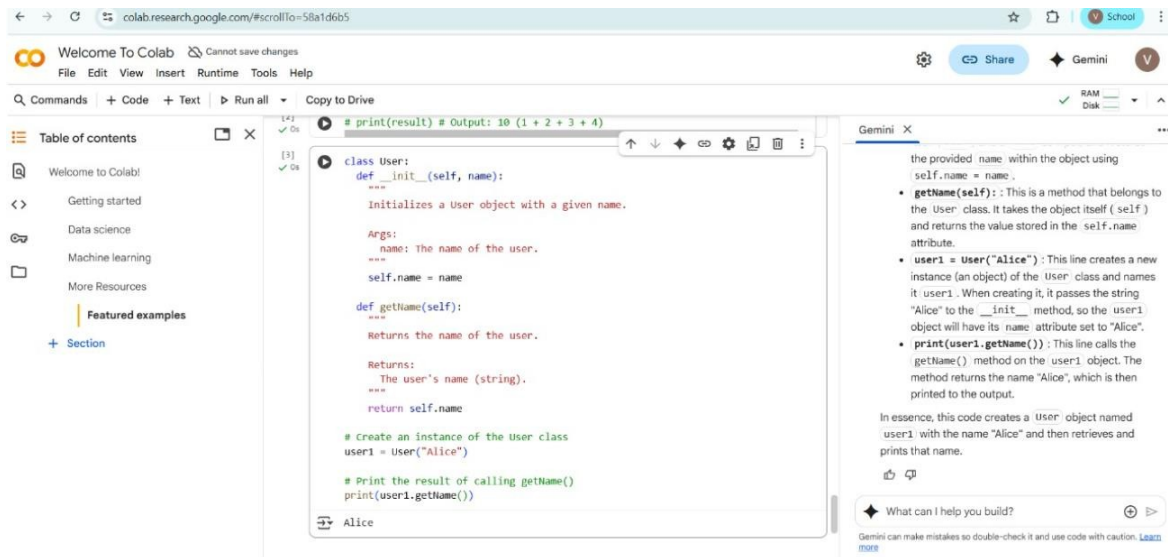
class User:: This line declares a new class named User.

__init__(self, name):: This is a special method called the constructor. It runs whenever you create a new User object. It takes the object itself (self) and a name as input, and it stores the provided name within the object using self.name = name.

getName(self):: This is a method that belongs to the User class. It takes the object itself (self) and returns the value stored in the self.name attribute.

user1 = User("Alice"): This line creates a new instance (an object) of the User class and names it user1. When creating it, it passes the string "Alice" to the __init__ method, so the user1 object will have its name attribute set to "Alice".

print(user1.getName()): This line calls the getName() method on the user1 object. The method returns the name "Alice", which is then printed to the output.

TASK 4

PROMPT

Write a Python class called Car that has a method start which prints "car started". Then, create an instance of the Car class and call the start method.

EXPLAINATON :

class Car:: This line defines a new class named Car. Think of this as a blueprint for creating car objects.

def start(self):: This defines a method within the Car class called start. Methods are functions that belong to a class. The self parameter refers to the instance of the class itself, allowing the method to access and modify the object's attributes (though in this case, it doesn't modify any).

The text within the triple quotes """...""" is a docstring, which explains what the start method does.

print("car started"): This line is inside the start method. When the start method is called, it will print the string "car started" to the console.

my_car = Car(): This line creates a new instance of the Car class and assigns it to the variable my_car. This is like building a car based on the Car blueprint.

my_car.start(): This line calls the start method on the my_car object. This executes the code inside the start method, which in this case prints "car started".
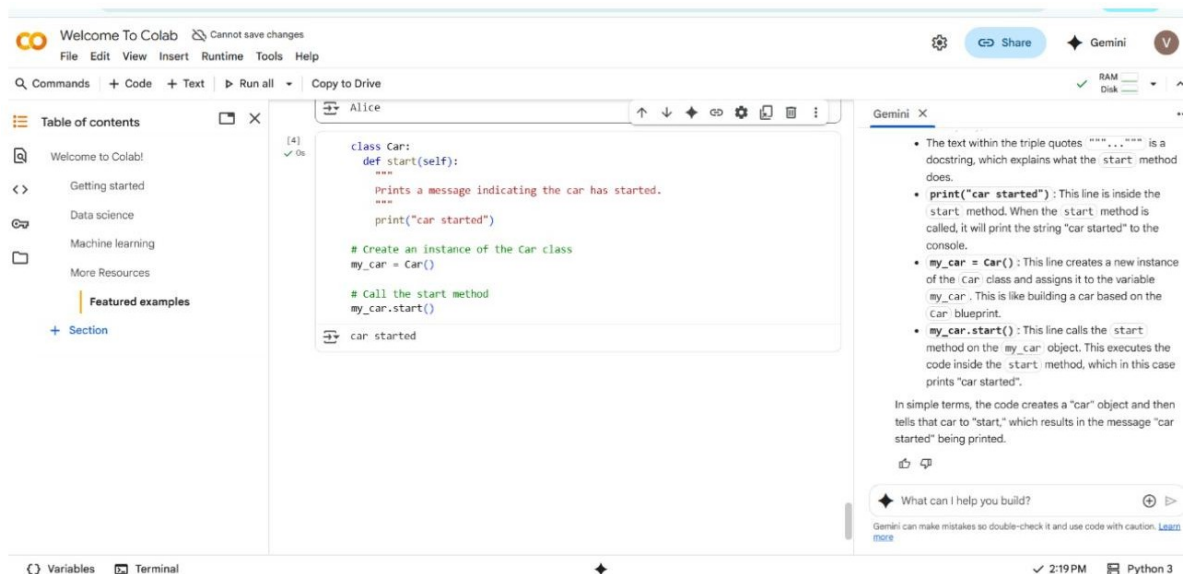
TASK 5

PROMPT

Write a Python function grade_student(score) that takes a student's score and returns a grade based on

the following conditions: Return "A" if the score is less than 40 Return "B" if the score is less than 70 Return "C" otherwise

EXPLAINATION

def grade_student(score):: This line defines the function grade_student and indicates that it accepts one argument, score.

The text within the triple quotes """...""" is a docstring explaining the function's purpose, arguments, and return value.

if score < 40:: This is the first condition. If the score is less than 40, the code inside this if block is executed.

return "A": If the score is less than 40, the function immediately stops and returns the string "A".

elif score < 70:: If the first condition (score < 40) is false, the code checks this elif (else if) condition. If the score is less than 70 (but not less than 40), the code inside this elif block is executed.

CO **Welcome To Colab** ⬡ Cannot save changes

File Edit View Insert Runtime Tools Help

⚙ GD Share ◆ Gemini

Q Commands + Code + Text ▷ Run all ▾ Copy to Drive

RAM ___
Disk ___

↑ ↓ ◆ GD ⚙ ⬚ ⬚ ⋮

Alice

[6]
✓ 0s

```python
def grade_student(score):
    """
    Assigns a letter grade based on the score using updated criteria.

    Args:
        score: An integer representing the student's score.

    Returns:
        A string representing the letter grade ("A", "B", or "C").
    """
    if score < 40:
        return "A"
    elif score < 70:
        return "B"
    else:
        return "C"

# Example usage:
# print(grade_student(30)) # Output: A
# print(grade_student(55)) # Output: B
# print(grade_student(85)) # Output: C
```

[4]
✓ 0s

```python
class Car:
    def start(self):
        """
        Prints a message indicating the car has started.
        """
```

Gemini X

- **elif score < 70:** : If the first condition (
  < 40) is false, the code checks this elif
  condition. If the score is less than 70 (but
  than 40), the code inside this elif block i
  executed.
- **return "B"** : If the score is less than 70
  less than 40, the function immediately stop
  returns the string "B".
- **else:** : If neither of the above conditions (
  < 40 or score < 70) is true, the code ins
  else block is executed. This means the sc
  or greater.
- **return "C"** : If the score is 70 or greater, t
  function immediately stops and returns the
  "C".

The commented-out lines at the end show exampl
how to call the function with different scores and t
expected output for each.

👍 👎

◆ What can I help you build?

Gemini can make mistakes so double-check it and use code with caut
more

{} Variables ⬚ Terminal ◆ ✓ 2:27 PM ⬚ Py

🌡 30°C ENG