LAB ASSIGNMENT

NAME: B.Srinithi

ROLL NO : 2403A51413

COURSE : AIML

BATCH:01

QUESTION :

Duration	2 Hours	Applicableto Batches	

AssignmentNumber:**6.3**(Present assignment number)/**24**(Total number of assignments)

Q.No.	Question	ExpectedTime to complete
1	Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals **Lab Objectives:** • To explore AI-powered auto-completion features for core Python constructs. • To analyze how AI suggests logic for class definitions, loops, and conditionals. • To evaluate the completeness and correctness of code generated by AI assistants. **Lab Outcomes (LOs):**	Week3 - Wednesday

After completing this lab, students will be able to:

- Use AI tools to generate and complete class definitions and methods.
- Understand and assess AI-suggested loops for iterative tasks.
- Generate conditional statements through prompt-driven suggestions.
- Critically evaluate AI-assisted code for correctness and clarity.

Task Description#1 (Classes)
- Use AI to complete a Student class with attributes and a method.
- Check output
- Analyze the code generated by AI tool

Expected Output#1
- Class with constructor and display_details() method

Task Description#2 (Loops)
- Prompt AI to complete a function that prints the first 10 multiples of a number using a loop.
- Analyze the generated code
- Ask AI to generate code using other controlled looping

Expected Output#2
- Correct loop-based implementation

Task Description#3 (Conditional Statements)
- Ask AI to write nested if-elif-else conditionals to classify age groups.
- Analyze the generated code
- Ask AI to generate code using other conditional statements

Expected Output#3
- Age classification function with appropriate conditions and with explanation

Task Description#4 (For and While loops)
- Generate a sum_to_n() function to calculate sum of first n numbers
- Analyze the generated code
- Get suggestions from AI with other controlled looping

- Analyze the generated code
- Ask AI to generate code using other conditional statements

Expected Output#3
- Age classification function with appropriate conditions and with explanation

Task Description#4 (For and While loops)
- Generate a sum_to_n() function to calculate sum of first n numbers
- Analyze the generated code
- Get suggestions from AI with other controlled looping

Expected Output#4
- Python code with explanation

Task Description#5 (Class)
- Use AI to build a BankAccount class with deposit, withdraw, and balance methods.
- Analyze the generated code
- Add comments and explain code

Expected Output#5
- Python code with explanation

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
Class	1.0

TASK 1

PROMPT :

Use AI to write a Python Student class with some attributes (like name, age, and grade) and one method to display student details. Show the output and explain the code in an easy way.""Write a Python class named Student with a constructor that initializes attributes like name, age, and grade. Include a display_details() method that prints the student's information. Show the output and explain the code simply
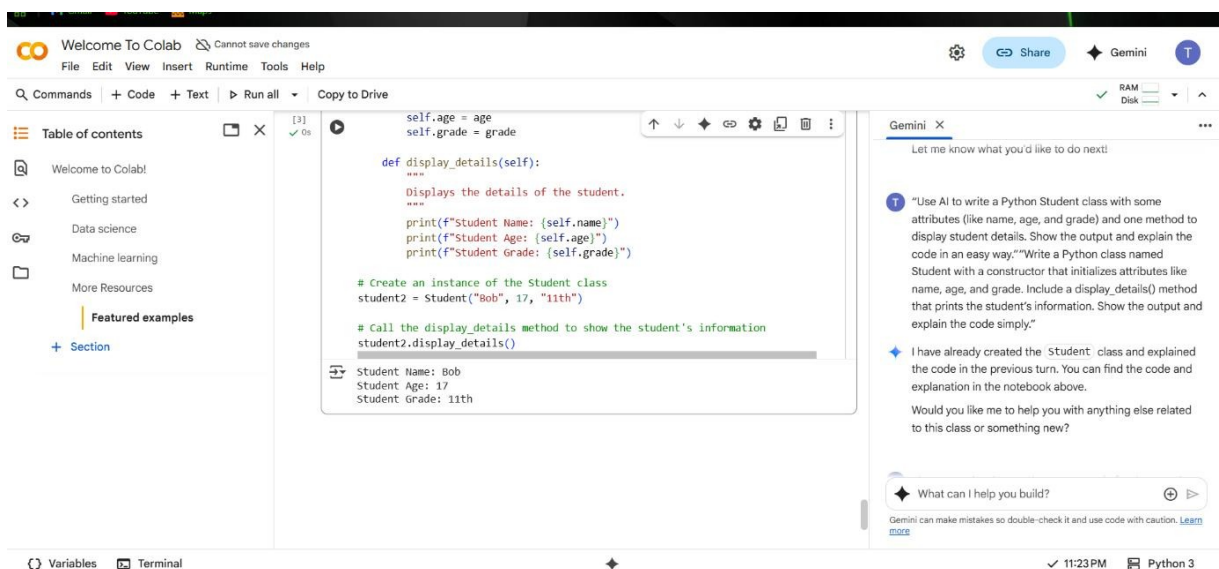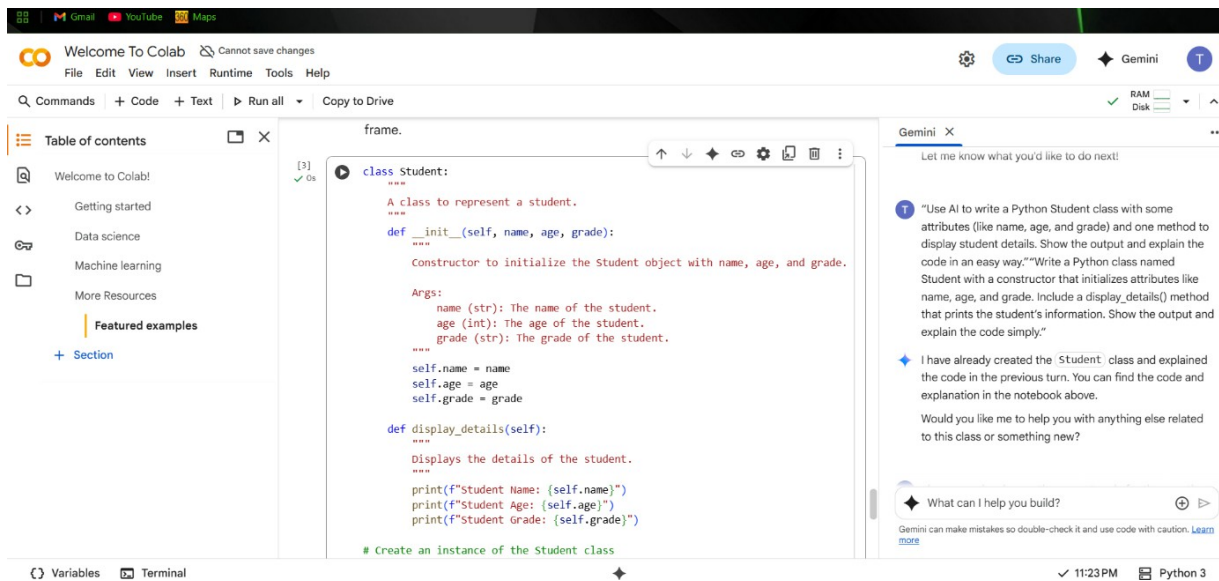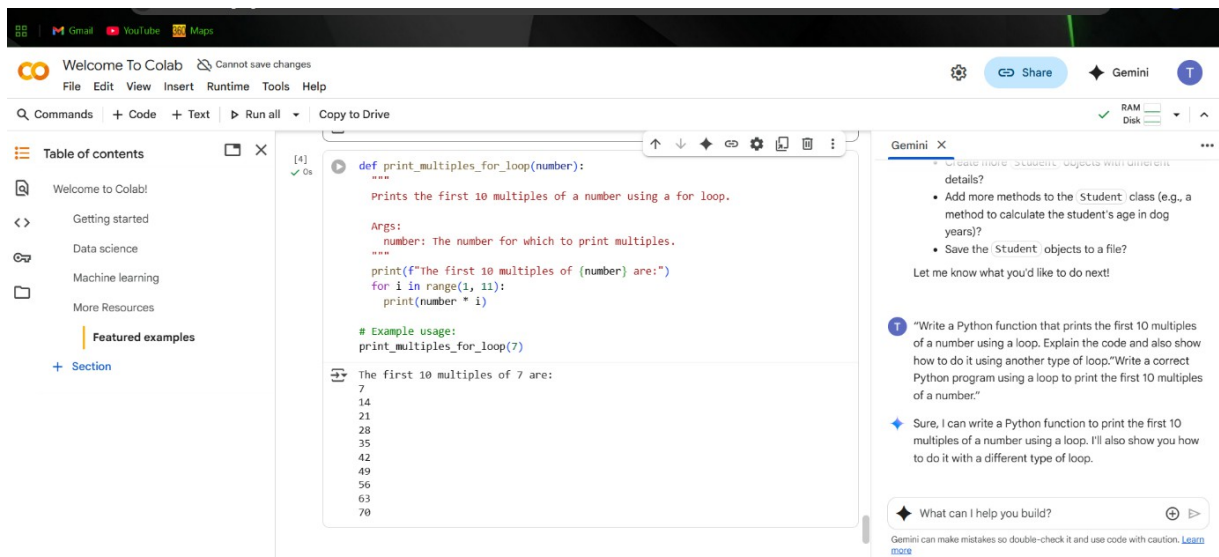
TASK 2

Prompt

Write a Python function that prints the first 10 multiples of a number using a loop. Explain the code and also show how to do it using another type of loop."Write a correct Python program using a loop to print the first 10 multiples of a number

Output

Task 3

Prompt :

Write a Python program using nested if-elif-else statements to classify age groups. Explain the code and also show how to do it using other types of conditionals.""Write a Python function to classify age groups with proper conditions and explain how the code works in a simple way.

OUTPUT:

Explaination :

- **Define the function**: Create a Python function that takes age as input.

- **Implement nested if-elif-else**: Use nested if-elif-else statements within the function to classify age groups (e.g., Child, Teenager, Adult, Senior).

- **Add explanation**: Provide a simple explanation of the code and how the nested if-elif-else statements work.

- **Implement alternative conditional**: Show an alternative way to classify age groups using a different conditional structure (e.g., a dictionary or a series of simple if-elif-else statements without nesting).
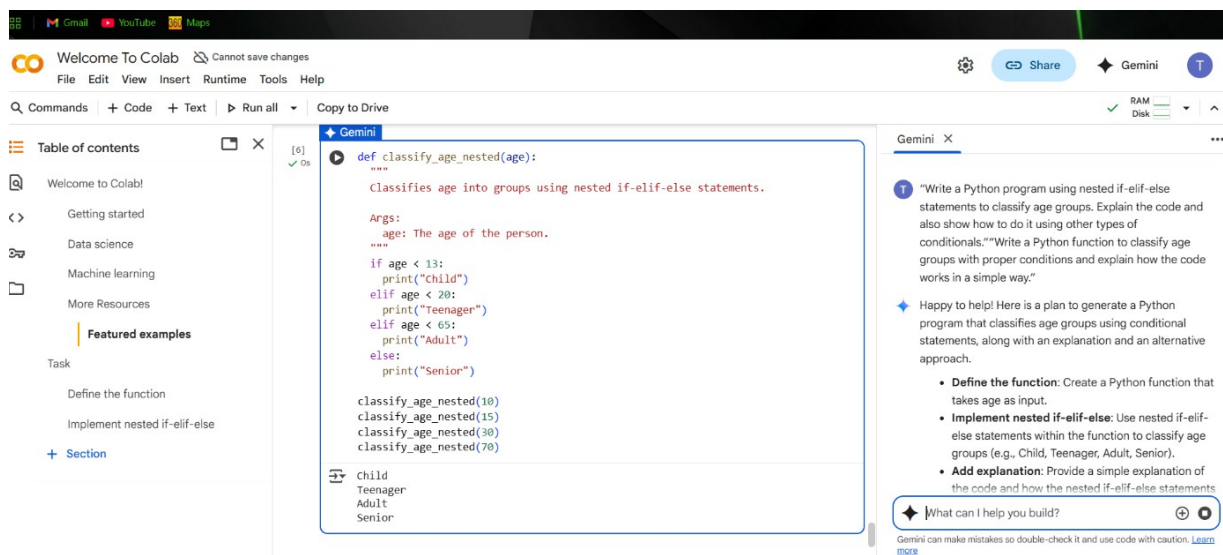
TASK 4

PROMPT :

Write a Python function sum_to_n() to calculate the sum of the first n numbers. Explain the code and show ho

w to do it using other loops.Python code with explanation

Explaination :

- **Define the function using a for loop**: Create a Python function sum_to_n_for() that takes an integer n as input and calculates the sum of the first n numbers using a for loop.

- **Explain the for loop code**: Add a markdown cell explaining the sum_to_n_for() function and how the for loop is used to calculate the sum.

- **Define the function using a while loop**: Create a Python function sum_to_n_while() that takes an integer n as input and calculates the sum of the first n numbers using a while loop.

- **Explain the while loop code**: Add a markdown cell explaining the sum_to_n_while() function and how the while loop is used.

OUTPUT:

TASK 5 :

PROMPT :

Create a Python class called BankAccount that has methods for depositing money, withdrawing money, and checking the balance. Include error checks for negative amounts and insufficient funds. Add comments and explain each method clearly."

OUTPUT :

Explanation :

- **Define the bankaccount class**: Create a Python class named BankAccount with a constructor to initialize the account balance.

- **Implement deposit method**: Add a deposit method to the BankAccount class that takes an amount as input and adds it to the balance. Include comments.

- **Implement withdraw method**: Add a withdraw method to the BankAccount class that takes an amount as input and subtracts it from the balance, ensuring there are sufficient funds. Include comments.

- **Implement balance method**: Add a balance method to the BankAccount class that returns the current account balance. Include comments.

- **Add example usage**: Create instances of the BankAccount class and demonstrate the usage of the deposit, withdraw, and balance methods.

- **Explain the code**: Provide a detailed explanation of the BankAccount class, its methods, and how they work together.

- **Explain the output**: Explain the output generated from the example usage, showing how the balance changes after each operation.

```
                    # Check if the withdrawal amount is valid
                    if amount <= 0:
                        print("Error: Withdrawal amount must be positive.")
                    # Check for sufficient funds
                    elif amount > self.balance:
                        print("Error: Insufficient funds.")
                    else:
                        self.balance -= amount
                        print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

                def check_balance(self):
                    """
                    Displays the current account balance.
                    """
                    print(f"Current balance for {self.account_holder}: ${self.balance:.2f}")

            # Example usage:
            my_account = BankAccount("John Doe", 1000.00)
            my_account.deposit(500.00)
            my_account.withdraw(200.00)
            my_account.check_balance()
            my_account.withdraw(1500.00) # Example of insufficient funds
            my_account.deposit(-100.00) # Example of invalid deposit
```

```
Account created for John Doe with an initial balance of $1000.00
Deposited $500.00. New balance: $1500.00
Withdrew $200.00. New balance: $1300.00
Current balance for John Doe: $1300.00
Error: Insufficient funds.
Error: Deposit amount must be positive.
```

Gemini ✕

Create a Python class called BankAccount that has methods for depositing money, withdrawing money, and checking the balance. Include error checks for negative amounts and insufficient funds. Add comments and explain each method clearly."

Absolutely! I can create a BankAccount class with methods for depositing, withdrawing, and checking the balance, including the error checks you mentioned. I'll also add comments and explanations for each method.

The BankAccount class has been successfully created with the requested methods and error handling, and the example usage demonstrates its functionality.

What can I help you build?

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