

LAB ASSIGNMENT-6.3

< AI Assisted Coding >

NAME: V.VAMSHI

HALLTICKETNUMBER: 2403A52016

BATCH NUMBER: 02

TASK_1:

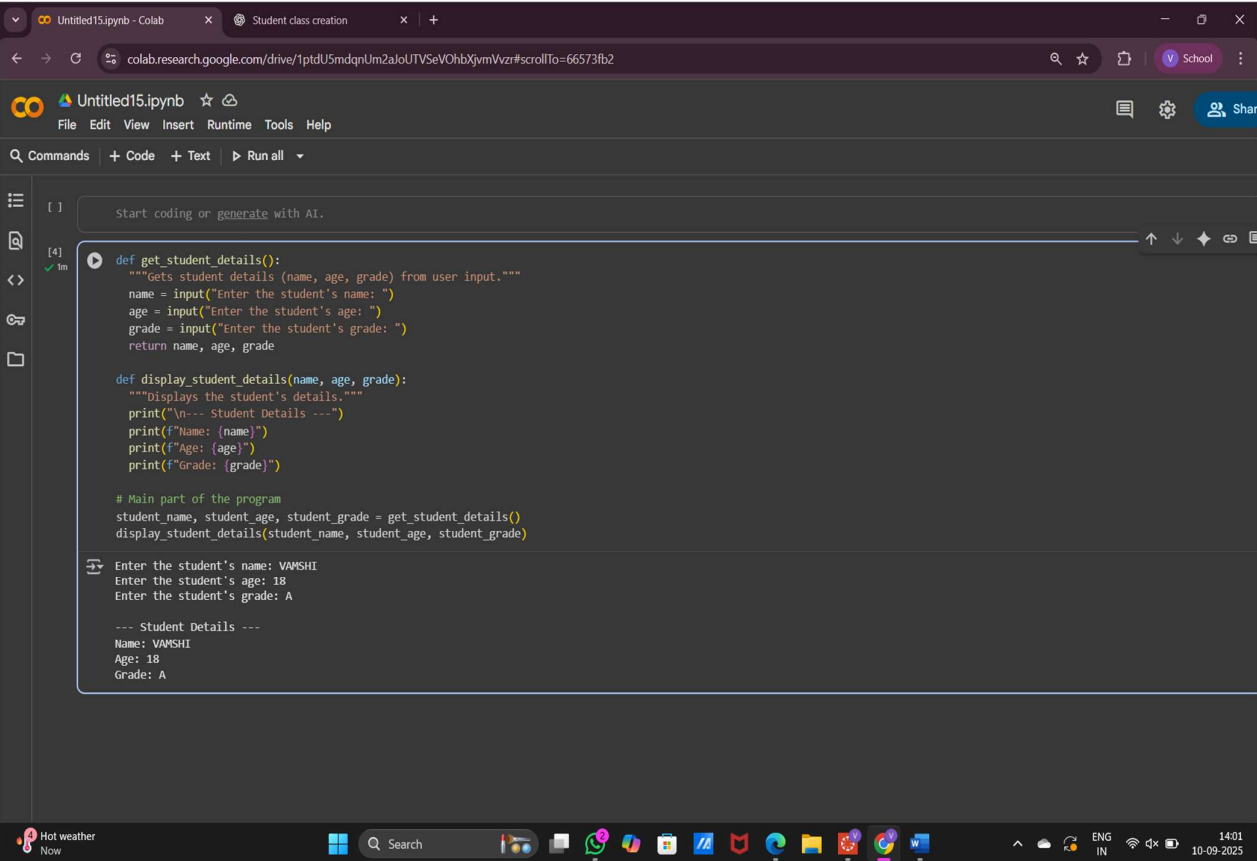
- >) Use AI to complete a Student class with attributes and a method.
- >) Check output
- >)Analyze the code generated by AI tool

Expected Output#1

- Class with constructor and display_details() method

PROMPT:

Write a Python program that asks the user for a student's name, age, and grade, then displays the student's details.



The screenshot shows a Google Colab notebook titled 'Untitled15.ipynb'. The code defines two functions: `get_student_details()` and `display_student_details(name, age, grade)`. The first function prompts the user for name, age, and grade. The second function prints the details in a formatted way. The main part of the program calls these functions with the input 'VAMSHI', '18', and 'A'. The output shows the student's details as entered.

```
[ ] Start coding or generate with AI.

[4] ✓ In
def get_student_details():
    """Gets student details (name, age, grade) from user input."""
    name = input("Enter the student's name: ")
    age = input("Enter the student's age: ")
    grade = input("Enter the student's grade: ")
    return name, age, grade

def display_student_details(name, age, grade):
    """Displays the student's details."""
    print("\n--- Student Details ---")
    print(f"Name: {name}")
    print(f"Age: {age}")
    print(f"Grade: {grade}")

# Main part of the program
student_name, student_age, student_grade = get_student_details()
display_student_details(student_name, student_age, student_grade)

Enter the student's name: VAMSHI
Enter the student's age: 18
Enter the student's grade: A

--- Student Details ---
Name: VAMSHI
Age: 18
Grade: A
```

OBSERVATION:

The code you executed defines two functions `get_student_details` to get input from the user for a student's name, age, and grade, and `display_student_details` to print these details. The main part of the program calls these functions to collect and then display the information. The output shows that the program successfully prompted you for the student's details and then displayed them as entered:

- **Name:** VAMSHI
- **Age:** 18
- **Grade:** A

Task Description#2 (Loops)

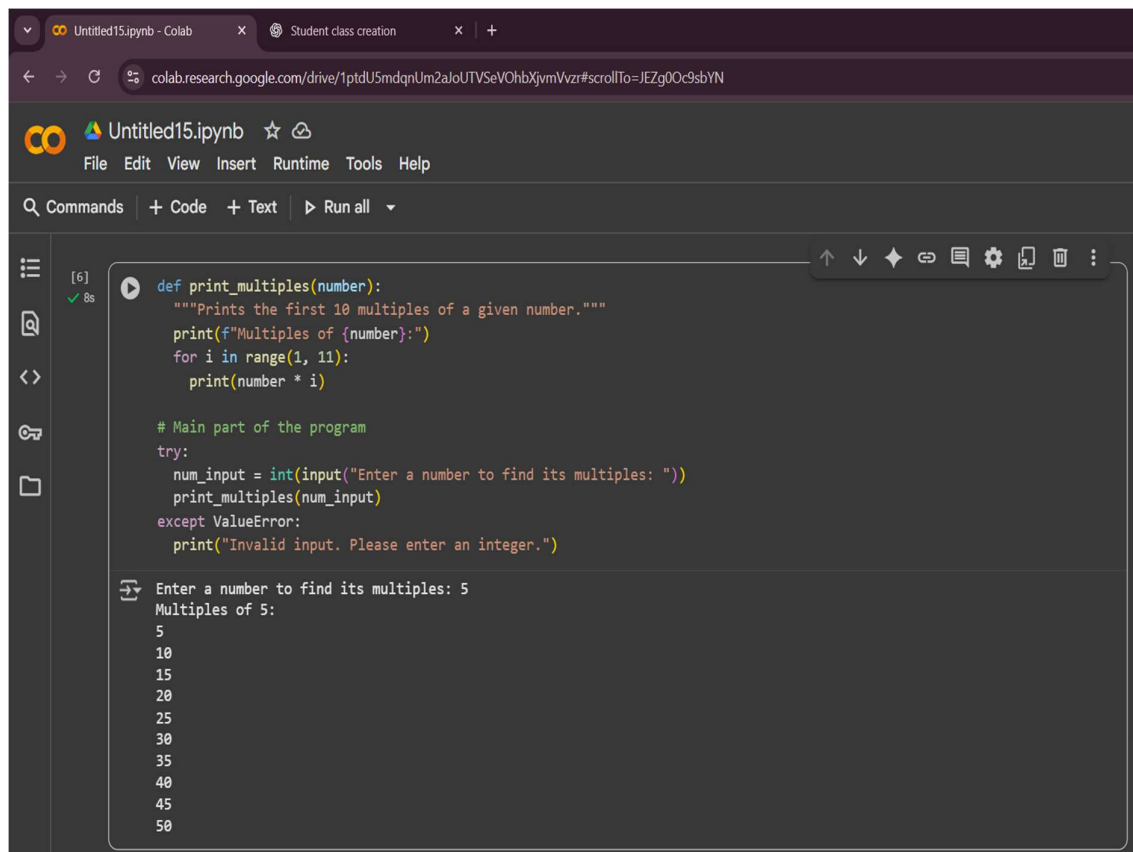
- Prompt AI to complete a function that prints the first 10 multiples of a number using a loop.
- Analyze the generated code
- Ask AI to generate code using other controlled looping

Expected Output#2

- Correct loop-based implementation

PROMPT:

Write a Python function named `print_multiples` that takes a number as input and prints the first 10 multiples of that number using a loop. take a number as user input and then print its multiples.



The screenshot shows a Google Colab notebook interface. At the top, there are two tabs: 'Untitled15.ipynb - Colab' and 'Student class creation'. The address bar shows a Google Drive link. The notebook has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar is a toolbar with 'Commands', '+ Code', '+ Text', and 'Run all'. The main area contains a code cell with the following Python code:

```
[6] ✓ 8s
def print_multiples(number):
    """Prints the first 10 multiples of a given number."""
    print(f"Multiples of {number}:")
    for i in range(1, 11):
        print(number * i)

# Main part of the program
try:
    num_input = int(input("Enter a number to find its multiples: "))
    print_multiples(num_input)
except ValueError:
    print("Invalid input. Please enter an integer.")
```

Below the code, the output is displayed:

```
Enter a number to find its multiples: 5
Multiples of 5:
5
10
15
20
25
30
35
40
45
50
```

OBSERVATION:

The code in the cell you executed defines a function `print_multiples` that takes a number and prints its first 10 multiples. The main part of the code then prompts you to enter a number using `input()`, converts that input to an integer, and passes it to the `print_multiples` function. It also includes error handling for non-integer input.

Based on the output, you entered the number 5. The code then successfully printed the first 10 multiples of 5, from 5 to 50.

TaskDescription#3(ConditionalStatement)

- 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

ExpectedOutput#3

- Age classification function with appropriate conditions and with explanation

PROMPT:

Write a Python function that asks the user to enter their age, then uses nested if-elif-else statements to classify the age into groups like 'Child', 'Teenager', 'Adult', and 'Senior'. The function should display the classification and include comments explaining each condition.

The screenshot shows a Google Colab notebook titled 'Untitled15.ipynb'. The code defines a function `classify_age()` that prompts the user for their age and classifies them into groups using nested if-elif-else statements. The groups are: Child (age < 13), Teenager (13 ≤ age < 20), Adult (20 ≤ age < 65), and Senior (age ≥ 65). The function also includes error handling for negative and non-integer inputs. The output shows the user entering '20' and being classified as an 'Adult'.

```
[9] def classify_age():  
    """  
    Asks the user for their age and classifies it into groups using nested  
    if-elif-else statements.  
    """  
    try:  
        age = int(input("Please enter your age: "))  
  
        # Check if the age is non-negative  
        if age >= 0:  
            # Classify age into groups  
            if age < 13:  
                print("You are classified as a Child.") # Age less than 13  
            elif age < 20:  
                print("You are classified as a Teenager.") # Age between 13 and 19  
            elif age < 65:  
                print("You are classified as an Adult.") # Age between 20 and 64  
            else:  
                print("You are classified as a Senior.") # Age 65 or older  
        else:  
            print("Age cannot be negative. Please enter a valid age.") # Handle negative age input  
  
    except ValueError:  
        print("Invalid input. Please enter a whole number for your age.") # Handle non-integer input  
  
    # Call the function to run the program  
    classify_age()  
  
Please enter your age: 20  
You are classified as an Adult.
```

OBSERVATION:

It defines a function `classify_age` that prompts the user to enter their age. It then uses nested if-elif-else statements to determine if the age is a Child (under 13), Teenager (13-19), Adult (20-64), or Senior (65+). It also includes error handling for negative or non-integer input.

The output shows that you entered the age 20. The code correctly classified this age as an "Adult" and printed the corresponding message.

Task Description#4 (For and While loops)

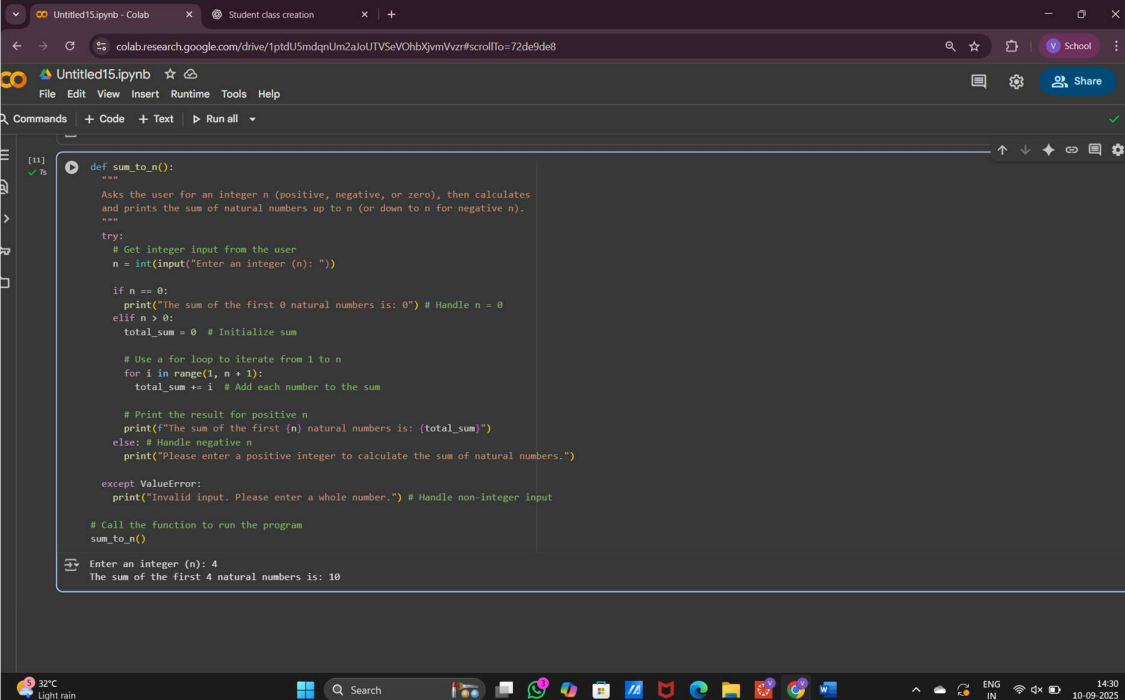
- Generate a `sum_to_n()` function to calculate sum of first `n` numbers
- Analyze the generated code
- Get suggestions from AI with other controlled looping

Expected Output#4

- Python code with explanation

PROMPT:

Write a Python function `sum_to_n()` that asks the user to input an integer `n` (positive, negative, or zero), then calculates and prints the sum of the first `n` natural numbers using a for loop. Include comments explaining the code and handle cases where `n` is zero or negative appropriately.



```
[11]: def sum_to_n():  
    """  
    Asks the user for an integer n (positive, negative, or zero), then calculates  
    and prints the sum of natural numbers up to n (or down to n for negative n).  
    """  
    try:  
        # Get integer input from the user  
        n = int(input("Enter an integer (n): "))  
  
        if n == 0:  
            print("The sum of the first 0 natural numbers is: 0") # Handle n = 0  
        elif n > 0:  
            total_sum = 0 # Initialize sum  
  
            # Use a for loop to iterate from 1 to n  
            for i in range(1, n + 1):  
                total_sum += i # Add each number to the sum  
  
            # Print the result for positive n  
            print(f"The sum of the first {n} natural numbers is: {total_sum}")  
        else: # Handle negative n  
            print("Please enter a positive integer to calculate the sum of natural numbers.")  
  
    except ValueError:  
        print("Invalid input. Please enter a whole number.") # Handle non-integer input  
  
    # Call the function to run the program  
    sum_to_n()  
  
Enter an integer (n): 4  
The sum of the first 4 natural numbers is: 10
```

OBSERVATION:

It defines the `sum_to_n` function, which now prompts the user for an integer `n` that can be positive, negative, or zero.

- If `n` is 0, it directly prints that the sum is 0.
- If `n` is positive, it uses a for loop to calculate the sum of natural numbers from 1 up to `n`.
- If `n` is negative, it prompts the user to enter a positive integer.
- It also includes error handling for non-integer input.

Based on the output, you entered the number 4. The code then correctly calculated and printed the sum of the first 4 natural numbers, which is 10.

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

PROMPT:

Write a Python program that defines a `BankAccount` class with methods `deposit(amount)`, `withdraw(amount)`, and `balance()`. The program should interactively prompt the user to deposit or withdraw money, and display the balance after each operation. Include comments explaining the code and handle invalid inputs gracefully

```
Untitled15.ipynb - Colab
Student class creation
colab.research.google.com/drive/1ptdU5mdqnlUm2aJoUTVSeVOhbXymVzr#scrollTo=6e5a1e9e

Untitled15.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM Disk
[24] ✓ 25s
class BankAccount:
    """Represents a bank account with deposit, withdraw, and balance functionalities."""

    def __init__(self, account_number):
        """Initializes a new bank account with a zero balance and stores the account number."""
        self.balance = 0.0 # Initialize the balance to zero
        self.account_number = account_number # Store the account number

    def deposit(self, amount):
        """Deposits a specified amount into the account. Validates that the amount is positive."""
        if amount > 0:
            self.balance += amount
            print(f"Successfully deposited {amount:.2f}. New balance: {self.balance:.2f}")
        else:
            print("Invalid deposit amount. Please enter a positive amount.")

    def withdraw(self, amount):
        """Withdraws a specified amount from the account. Validates that the amount is positive and does not exceed the balance."""
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            print(f"Successfully withdrew {amount:.2f}. New balance: {self.balance:.2f}")
        elif amount <= 0:
            print("Invalid withdrawal amount. Please enter a positive amount.")
        else:
            print("Insufficient funds.")

    def balance(self):
        """Displays the current account balance along with the account number."""
        print(f"Account Number: {self.account_number}, Current Balance: {self.balance:.2f}")

# Main part of the program

# Prompt user for account number and create a BankAccount instance
account_number = input("Enter your bank account number: ")
account = BankAccount(account_number) # Create an instance of the BankAccount class
```

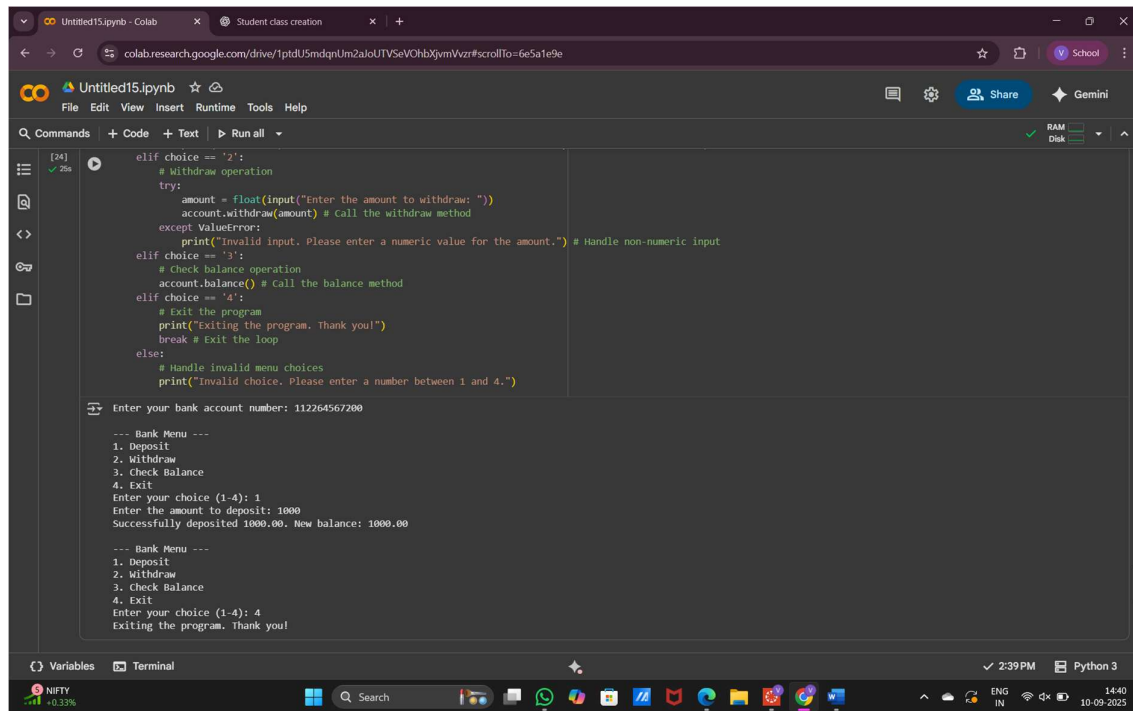
```
Untitled15.ipynb - Colab
Student class creation
colab.research.google.com/drive/1ptdU5mdqnlUm2aJoUTVSeVOhbXymVzr#scrollTo=6e5a1e9e

Untitled15.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM Disk
[24] ✓ 25s
# Prompt user for account number and create a BankAccount instance
account_number = input("Enter your bank account number: ")
account = BankAccount(account_number) # Create an instance of the BankAccount class

# Start the interaction loop to allow the user to perform banking operations
while True:
    print("\n--- Bank Menu ---")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")

    # Get user's choice
    choice = input("Enter your choice (1-4): ")

    # Handle user's choice
    if choice == '1':
        # Deposit operation
        try:
            amount = float(input("Enter the amount to deposit: "))
            account.deposit(amount) # Call the deposit method
        except ValueError:
            print("Invalid input. Please enter a numeric value for the amount.") # Handle non-numeric input
    elif choice == '2':
        # Withdraw operation
        try:
            amount = float(input("Enter the amount to withdraw: "))
            account.withdraw(amount) # Call the withdraw method
        except ValueError:
            print("Invalid input. Please enter a numeric value for the amount.") # Handle non-numeric input
    elif choice == '3':
        # Check balance operation
        account.balance() # Call the balance method
    elif choice == '4':
        # Exit the program
        print("Exiting the program. Thank you!")
```



```
elif choice == '2':
    # Withdraw operation
    try:
        amount = float(input("Enter the amount to withdraw: "))
        account.withdraw(amount) # Call the withdraw method
    except ValueError:
        print("Invalid input. Please enter a numeric value for the amount.") # Handle non-numeric input

elif choice == '3':
    # Check balance operation
    account.balance() # Call the balance method

elif choice == '4':
    # Exit the program
    print("Exiting the program. Thank you!")
    break # Exit the loop

else:
    # Handle invalid menu choices
    print("Invalid choice. Please enter a number between 1 and 4.")

Enter your bank account number: 112264567200

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter the amount to deposit: 1000
Successfully deposited 1000.00. New balance: 1000.00

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 4
Exiting the program. Thank you!
```

OBSERVATION:

In the above code it creates a BankAccount class with deposit, withdraw, and balance methods, initialized with an account number. It interacts with the user via a menu. Error handling for invalid input is included. The output shows a deposit of 1000 into account 112264567200 and then exiting the program.