

# **AIAC LAB 6.3**

**TOPIC: AI-Based Code Completion – Classes,  
Loops, and Conditionals**

**ENROLLMENT NO: 2403A52006**

**NAME: Mohammed Mustafa**

**COURSE : CSE(AI/ML)**

**BATCH: 01**

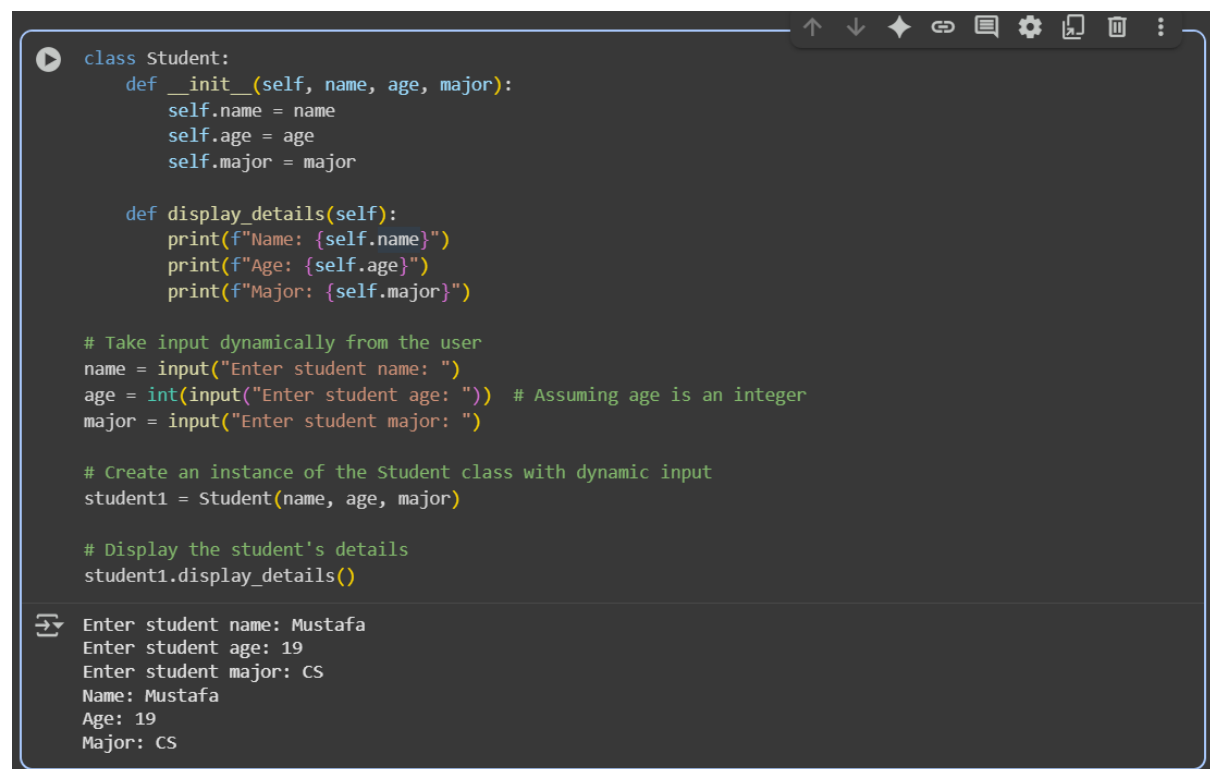
**DATE: 10-09-2025**

## 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

### Used Prompt:

generate a python code to complete a student class with attributes (name,age) and a method (display\_details). take the inputs from the user.



```
class Student:
    def __init__(self, name, age, major):
        self.name = name
        self.age = age
        self.major = major

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Major: {self.major}")

# Take input dynamically from the user
name = input("Enter student name: ")
age = int(input("Enter student age: ")) # Assuming age is an integer
major = input("Enter student major: ")

# Create an instance of the Student class with dynamic input
student1 = Student(name, age, major)

# Display the student's details
student1.display_details()
```

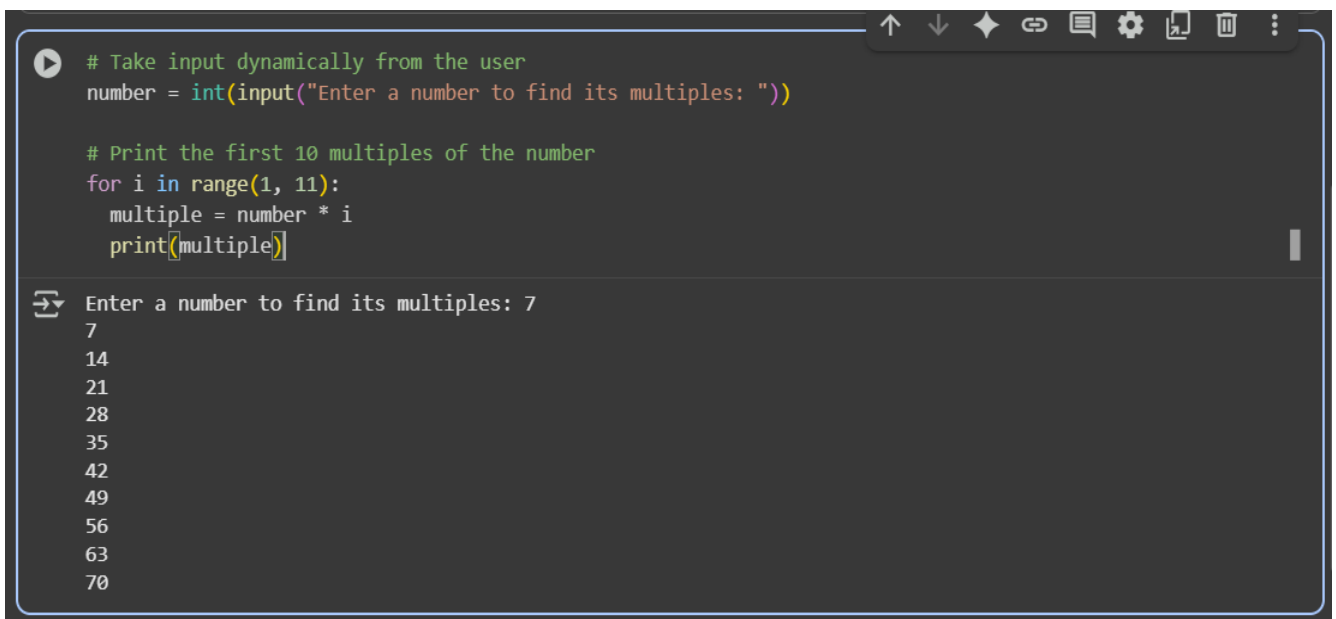
Enter student name: Mustafa  
Enter student age: 19  
Enter student major: CS  
Name: Mustafa  
Age: 19  
Major: CS

## 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

### Used Prompt:

generate a python code to print the first 10 multiples of a number using a loop. take input from the user.



```
# Take input dynamically from the user
number = int(input("Enter a number to find its multiples: "))

# Print the first 10 multiples of the number
for i in range(1, 11):
    multiple = number * i
    print(multiple)
```

Enter a number to find its multiples: 7

7  
14  
21  
28  
35  
42  
49  
56  
63  
70

## 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

### Used Prompt:

generate a python code to classify several age groups using if-else statements (if age:0-2 - infants) and keep making for all the ages up to 150. can take multiple ages from the user and classify them as the age group.

```
# Take multiple ages input dynamically from the user
ages_input = input("Enter ages separated by commas (e.g., 5, 15, 25): ")
ages = [int(age.strip()) for age in ages_input.split(',')]

# Classify age groups for each age
for age in ages:
    if 0 <= age <= 2:
        print(f"{age}: Infant")
    elif 3 <= age <= 12:
        print(f"{age}: Child")
    elif 13 <= age <= 17:
        print(f"{age}: Teenager")
    elif 18 <= age <= 19:
        print(f"{age}: Adult Teenager")
    elif 20 <= age <= 39:
        print(f"{age}: Adult")
    elif 40 <= age <= 59:
        print(f"{age}: Middle-aged Adult")
    elif 60 <= age <= 80:
        print(f"{age}: Senior Adult")
    elif 81 <= age <= 150:
        print(f"{age}: Elderly")
    else:
        print(f"{age}: Age group not classified in this program.")
```

Enter ages separated by commas (e.g., 5, 15, 25): 5,18,66  
5: Child  
18: Adult Teenager  
66: Senior Adult

### CODE EXPLANATION:

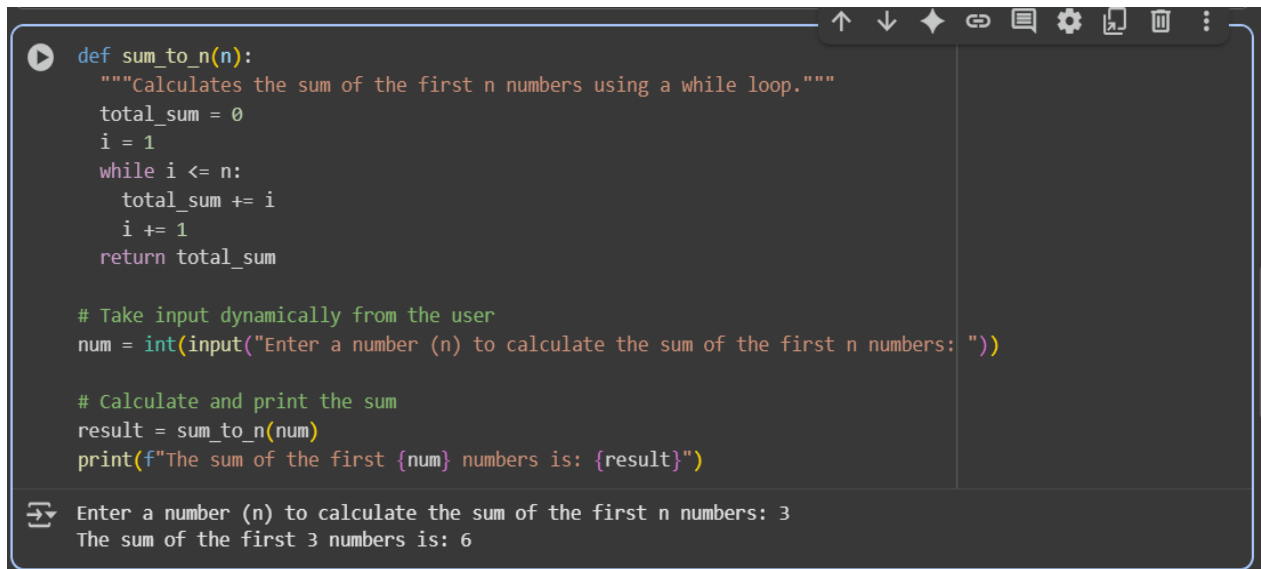
This code takes a list of ages as input, separated by commas. It then checks each age against a set of rules and prints out the corresponding age group classification for each one.

## 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

### Used Prompt:

generate a python function `sum_to_n()` to calculate the sum of first n numbers using a while loop. take the n value from the user.



```
def sum_to_n(n):  
    """Calculates the sum of the first n numbers using a while loop."""  
    total_sum = 0  
    i = 1  
    while i <= n:  
        total_sum += i  
        i += 1  
    return total_sum  
  
# Take input dynamically from the user  
num = int(input("Enter a number (n) to calculate the sum of the first n numbers: "))  
  
# Calculate and print the sum  
result = sum_to_n(num)  
print(f"The sum of the first {num} numbers is: {result}")
```

Enter a number (n) to calculate the sum of the first n numbers: 3  
The sum of the first 3 numbers is: 6

### CODE EXPLANATION:

This code defines a function called `sum_to_n` that calculates the sum of all numbers from 1 up to a number you provide. It then asks you to enter that number and prints the calculated sum.

## 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

### Used Prompt:

generate a python code to build a bank account class with deposit, withdraw and balance methods. take inputs from the user.

```
class BankAccount:
    def __init__(self, account_holder, initial_balance=0):
        self.account_holder = account_holder
        self.balance = initial_balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount}. New balance: ${self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > 0:
            if self.balance >= amount:
                self.balance -= amount
                print(f"Withdrew ${amount}. New balance: ${self.balance}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def get_balance(self):
        return self.balance

# Take input dynamically from the user
account_holder_name = input("Enter account holder name: ")
initial_deposit = float(input("Enter initial deposit amount: "))
```

```
# Create a bank account instance
my_account = BankAccount(account_holder_name, initial_deposit)

# Perform some transactions (example)
my_account.deposit(float(input("Enter deposit amount: ")))
my_account.withdraw(float(input("Enter withdrawal amount: ")))

# Display the final balance
print(f"Final balance: ${my_account.get_balance()}")
```

```
Enter account holder name: Mustafa
Enter initial deposit amount: 1000
Enter deposit amount: 2500
Deposited $2500.0. New balance: $3500.0
Enter withdrawal amount: 3000
Withdrew $3000.0. New balance: $500.0
Final balance: $500.0
```

## CODE EXPLANATION:

This code creates a Bank Account with a holder's name and an initial balance. It then lets you deposit and withdraw money from the account and finally shows you the updated balance.