AI Assisted Coding

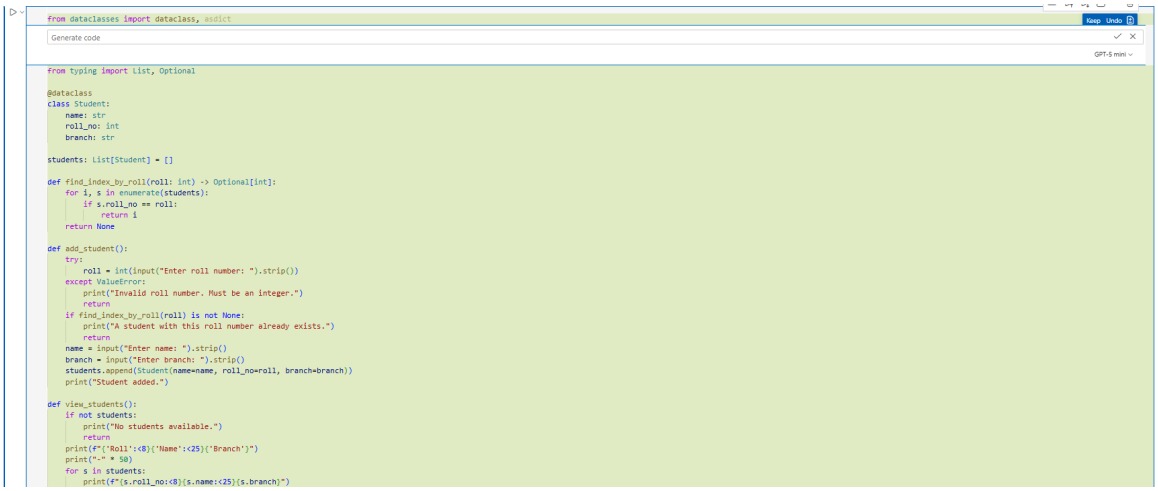
Name : Vineeth-Chidurala **Date** : 04-02-2026

Ht.No. : 2303A52447

Task - 1 : Classes (Student Class)
Scenario
You are developing a simple student information management module.
• Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
• The class should include attributes such as name, roll number, and branch.
• Add a method display_details() to print student information.
• Execute the code and verify the output.
• Analyze the code generated by the AI tool for correctness and clarity.

Prompt : Generate a simple python code for student information management module with details like name,roll no and branch with user input with brief analysis of the generated code.

Screenshots :



Output :

```
    print("\nStudent Information:")
    for student in students:
        student.display_info()

if __name__ == "__main__":
    main()
analysis = """
1. The code defines a `Student` class with an initializer to set the name, roll number, and branch of a student.
2. The `display_info` method in the `Student` class prints the student's details in a formatted string.
3. The `main` function handles user input to create multiple `Student` objects based on the number of students specified by the user.
4. It stores the created `Student` objects in a list and then iterates through the list to display each student's information.
5. The use of a class encapsulates student-related data and behavior, promoting code organization and reusability.
6. The program is interactive, allowing users to input data dynamically, making it flexible for different numbers of students.
"""
print(analysis)
```

[2] ✓ 28.2s

```
Student Information:
Name: vineeth, Roll No: 1, Branch: cse
Name: avinash, Roll No: 2, Branch: cse
Name: snehith, Roll No: 3, Branch: cse

1. The code defines a `Student` class with an initializer to set the name, roll number, and branch of a student.
2. The `display_info` method in the `Student` class prints the student's details in a formatted string.
3. The `main` function handles user input to create multiple `Student` objects based on the number of students specified by the user.
4. It stores the created `Student` objects in a list and then iterates through the list to display each student's information.
5. The use of a class encapsulates student-related data and behavior, promoting code organization and reusability.
6. The program is interactive, allowing users to input data dynamically, making it flexible for different numbers of students.
```

Task – 2 : Loops (Multiples of a Number)

You are writing a utility function to display multiples of a given number.

• Prompt the AI tool to generate a function that prints the first 10 multiples of a given number

using a loop.

• Analyze the generated loop logic.

• Ask the AI to generate the same functionality using another controlled looping structure (e.g.,

while instead of for).

Prompt : **1.**Write a Python code that accepts a number as dynamic input and prints the first 10 multiples of that number using a for loop.

Ensure the code is readable, correct, and displays the output clearly.

2.Generate a Python function that prints the first 10 multiples of a given number using a while loop.

Use dynamic input and ensure correct loop termination.

ScreenShots :

```
# Write a Python code that accepts a number as dynamic input and prints the first 10 multiples of that n
number = int(input("Enter a number: "))

print("\nFirst 10 multiples:\n")

for i in range(1, 11):
    result = number * i
    print(number, "x", i, "=", result)
```

[3] ✓ 2.7s

```python
# Generate a Python function that prints the first 10 multiples of a given number using a while loop
def print_multiples(num):
    i = 1
    while i <= 10:
        print(num, "x", i, "=", num * i)
        i += 1

number = int(input("Enter a number: "))
print("\nFirst 10 multiples:\n")
print_multiples(number)
```

Input : 10

Output :

```
    print(number, "x", i, "=", result)
```

[3] ✓ 2.7s

```
First 10 multiples:

10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

```python
print_multiples(number)
```

[5] ✓ 1.5s

```
First 10 multiples:

10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

Task-3 : Conditional Statements (Age Classification)

You are building a basic classification system based on age.

• Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups

(e.g., child, teenager, adult, senior).

• Analyze the generated conditions and logic.

• Ask the AI to generate the same classification using alternative conditional structures (e.g.,

simplified conditions or dictionary-based logic).

Prompt : **1.**Write a Python Code that takes age as dynamic user input and classifies the person into age groups such as child, teenager, adult, or senior using nested if-elif-else statements.

Ensure the logic is correct, readable, and handles valid age ranges.

2.Generate a Python Code age classification program using simplified conditional logic or a dictionary-based approach instead of multiple elif statements.

Use dynamic input and ensure clarity and correctness.

ScreenShots :

```python
# Write a Python Code that takes age as dynamic user input and classifies the person into age groups s
age = int(input("Enter your age: "))

if age >= 0:
    if age <= 12:
        print("Age Group: Child")
    elif age <= 19:
        print("Age Group: Teenager")
    elif age <= 59:
        print("Age Group: Adult")
    else:
        print("Age Group: Senior")
else:
    print("Invalid age entered")
```

```python
# Generate a Python Code age classification program using simplified conditional logic or a dictiona
#Use dynamic input and ensure clarity and correctness.

age = int(input("Enter your age: "))

groups = {
    "Child": range(0,13),
    "Teenager": range(13,20),
    "Adult": range(20,60),
    "Senior": range(60,200)
}

if age < 0:
    print("Invalid age")
else:
    for k,v in groups.items():
        if age in v:
            print("Age Group:", k)
            break
```

Input : 22

10

Output :

```python
    elif age <= 19:
        print("Age Group: Teenager")
    elif age <= 59:
        print("Age Group: Adult")
    else:
        print("Age Group: Senior")
else:
    print("Invalid age entered")
```

[6] ✓ 4.1s

```
Age Group: Adult
```

```python
if age < 0:
    print("Invalid age")
else:
    for k,v in groups.items():
        if age in v:
            print("Age Group:", k)
            break
```

[9] ✓ 1.8s

```
Age Group: Teenager
```

Task-4 : For and While Loops (Sum of First n Numbers)

You need to calculate the sum of the first n natural numbers.

• Use AI assistance to generate a sum_to_n() function using a for loop.

• Analyze the generated code.

• Ask the AI to suggest an alternative implementation using a while loop or a mathematical

formula.

Prompt : 1. Write a Python Code for sum_to_n() that takes a positive integer n as dynamic user input and calculates the sum of the first n natural numbers using a for loop.

Ensure the code is readable, correct, and handles basic input validation**.**

2.Generate an alternative implementation of the sum_to_n() function using a while loop or a mathematical formula.

Compare the logic and efficiency with the for loop version.

Screenshots :

```python
#Write a Python Code for sum_to_n() that takes a positive integer n as dynamic user input and calculates t
#Ensure the code is readable, correct, and handles basic input validation.
n = int(input("Enter a positive integer: "))

if n <= 0:
    print("Please enter a positive number.")
else:
    total = 0
    for i in range(1, n + 1):
        total += i

    print("Sum of first", n, "natural numbers is:", total)
```

```python
# Generate an alternative implementation of the sum_to_n() function us
#Compare the logic and efficiency with the for loop version.
def sum_to_n_while(n):
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total
```

[12] ✓ 0.0s

Input : 10

Output :

```
    for i in range(1, n + 1):
        total += i

    print("Sum of first", n, "natural numbers is:", total)
```

[10] ✓ 1.3s

```
Sum of first 10 natural numbers is: 55
```

Task-5 : Classes (Bank Account Class) You are designing a basic banking application.

Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().

Analyze the AI-generated class structure and logic.

Add meaningful comments and explain the working of the code.

Prompt : Create a Python Code-class named BankAccount.
The class should have attributes for account holder name and balance.
Implement methods deposit(), withdraw(), and check_balance().
Use dynamic user input, ensure input validation, and add meaningful comments.
Demonstrate deposit and withdrawal operations with updated balance output.

Screenshots :

```python
#Create a Python Code-class named BankAccount.
#The class should have attributes for account holder name and balance.
#Implement methods deposit(), withdraw(), and check_balance().
#Use dynamic user input, ensure input validation, and add meaningful comments.
#Demonstrate deposit and withdrawal operations with updated balance output.
class BankAccount:
    def __init__(self, name, balance):
        # Initialize account holder name and starting balance
        self.name = name
        self.balance = balance

    def deposit(self, amount):
        # Deposit only positive amounts
        if amount > 0:
            self.balance += amount
            print("Deposited:", amount)
        else:
            print("Invalid deposit amount")

    def withdraw(self, amount):
        # Withdraw only if amount is valid and balance is sufficient
        if amount <= 0:
            print("Invalid withdrawal amount")
        elif amount > self.balance:
            print("Insufficient balance")
        else:
            self.balance -= amount
```

Input : Vineeth
10000
5000
15000
2000
13000

Output :

```python
    account.deposit(dep)
    account.check_balance()

    wd = float(input("\nEnter amount to withdraw: "))
    account.withdraw(wd)
    account.check_balance()
```

[13] ✓ 19.0s

```
Welcome, vineeth
Current Balance: 10000.0
Deposited: 5000.0
Current Balance: 15000.0
Withdrawn: 2000.0
Current Balance: 13000.0
```