ASSIGNMENT- 6.3

Name: Rishitha Kunchala

HT.No: 2303A51333

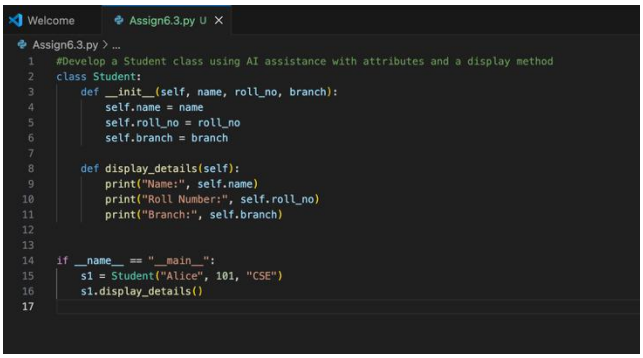
Batch: 20

Task 1: Classes – Student Class

Develop a Student class using AI assistance with attributes and a display method

.**Prompt: #Generate a Python Student class with name, roll number, and branch. Include a method to display student details..**

Code:



Result:

Observation:
The AI-generated class structure is clear and logically organized. The constructor correctly initializes attributes, and the display method outputs student details in a readable format. The code is simple, correct, and suitable for beginner-level object-oriented programming.

Task 2: Loops – Multiples of a Number. Generate code to print the first 10 multiples of a given number using different loop constructs.

Prompt: #Generate Python code to print the first

10 multiples of a number using a loop.

Code:

Result:

Observation:
Both loop implementations correctly generate the required output. The for-loop version is more concise and readable, while the while-loop version provides better insight into loop control and

iteration. AI suggestions for both approaches are correct and efficient.

Task 3: Conditional Statements – Age Classification. Classify a person's age into categories using conditional statements.

Prompt: # Generate Python code to classify age into child, teenager, adult, and senior using if-elif-else..

Code:

Result:

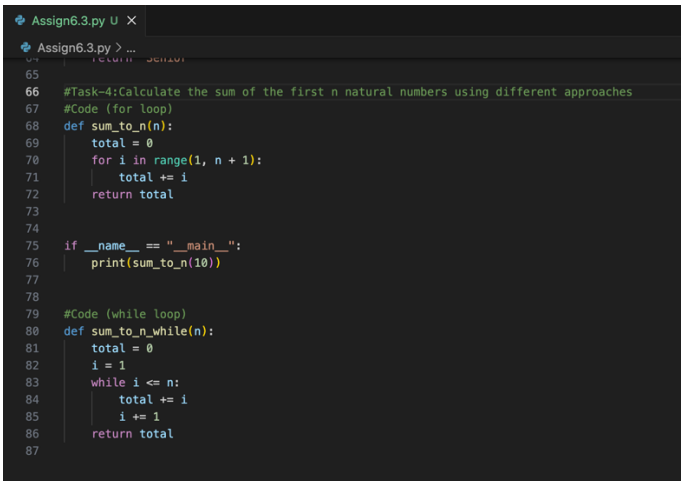
Observation:
The AI-generated conditions correctly classify age groups. The if-elif-else structure is clear and readable, while the simplified version reduces nesting and improves clarity. Both approaches are logically sound.

Task 4: For and While Loops – Sum of First n Numbers. Calculate the sum of the first n natural numbers using different approaches.

Prompt: #Generate Python code to find the sum of the first n natural numbers using

loops.

Code:



Result:

Observation

Both loop-based solutions produce the correct result. The for-loop version is more concise, while the while-loop version offers explicit control over iteration. AI-generated logic is correct and easy to understand

Task 5: Classes – Bank Account Class

Create a Bank Account class with deposit, withdraw, and balance checking functionality.

Prompt: #Generate a Python Bank Account class with deposit, withdraw, and check balance methods.

Code:

```python
#Task-5:Create a Bank Account class with deposit, withdraw, and balance checking functionality.
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")

    def check_balance(self):
        print("Current Balance:", self.balance)


if __name__ == "__main__":
    acc = BankAccount(5000)
    acc.deposit(1000)
    acc.withdraw(2000)
    acc.check_balance()
```

Result:

```
using while loop: 15
using while loop: 20
using while loop: 25
using while loop: 30
using while loop: 35
using while loop: 40
using while loop: 45
using while loop: 50
Adult
55
Deposited: 1000
Withdrawn: 2000
Current Balance: 4000
```

Observation:
The AI-generated class structure is well organized and logically correct. Methods perform expected operations, and balance updates are accurate. The code is readable, maintainable, and suitable for a basic banking application.