2303A510A1 - Bhanu

Batch - 14

Assignment - 6.1

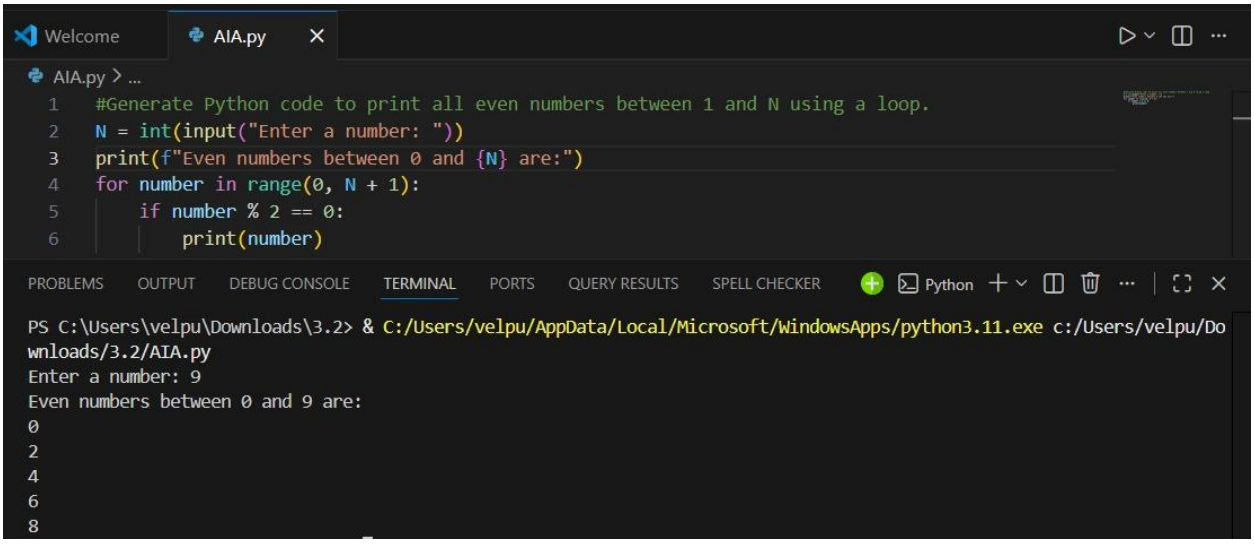
Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

Task 1 : Use an AI code completion tool to generate a loop-based program.

Prompt:

Generate Python code to print all even numbers between 1 and N using a loop.

Program and Output:

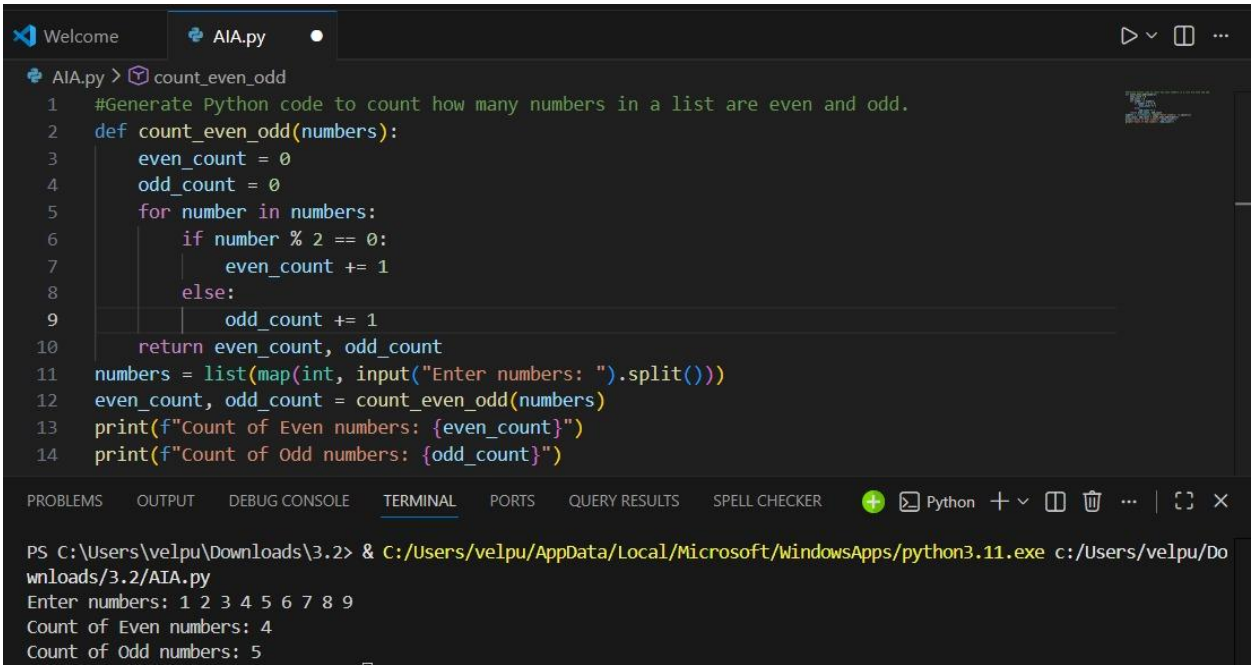


Task: Use an AI code completion tool to combine loops and conditionals.

Prompt:

Generate Python code to count how many numbers in a list are even and odd.

Program and Output:



Explanation of logic flow:

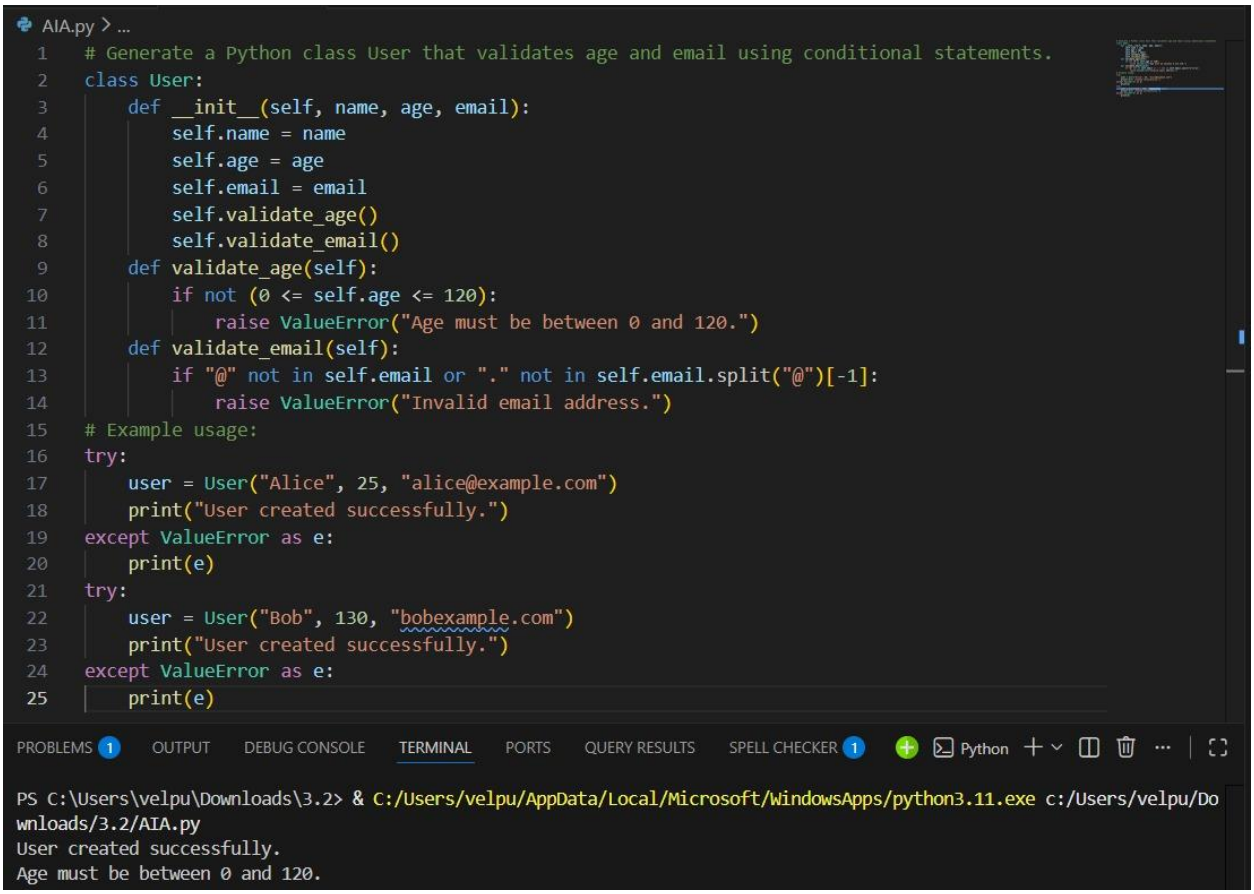
- This code defines a function 'count_even_odd' that takes a list of numbers as input.
- It initializes two counters, 'even_count' and 'odd_count', to zero.
- It then iterates through each number in the list, checking if it is even or odd using the modulus operator (%).
- If the number is even (i.e., divisible by 2), it increments the ' even_count' by 1; otherwise, it increments the 'odd_count' by 1.
- Finally, the function returns the counts of even and odd numbers.
- The user is prompted to enter a list of numbers, which are then converted to integers and stored in the 'numbers' list.
- The function is called with this list, and the results are printed to the console.

Task 3: Use an AI tool to complete a Python class that validates user input.

Prompt:

Generate a Python class User that validates age and email using conditional statements.

Program and Output:

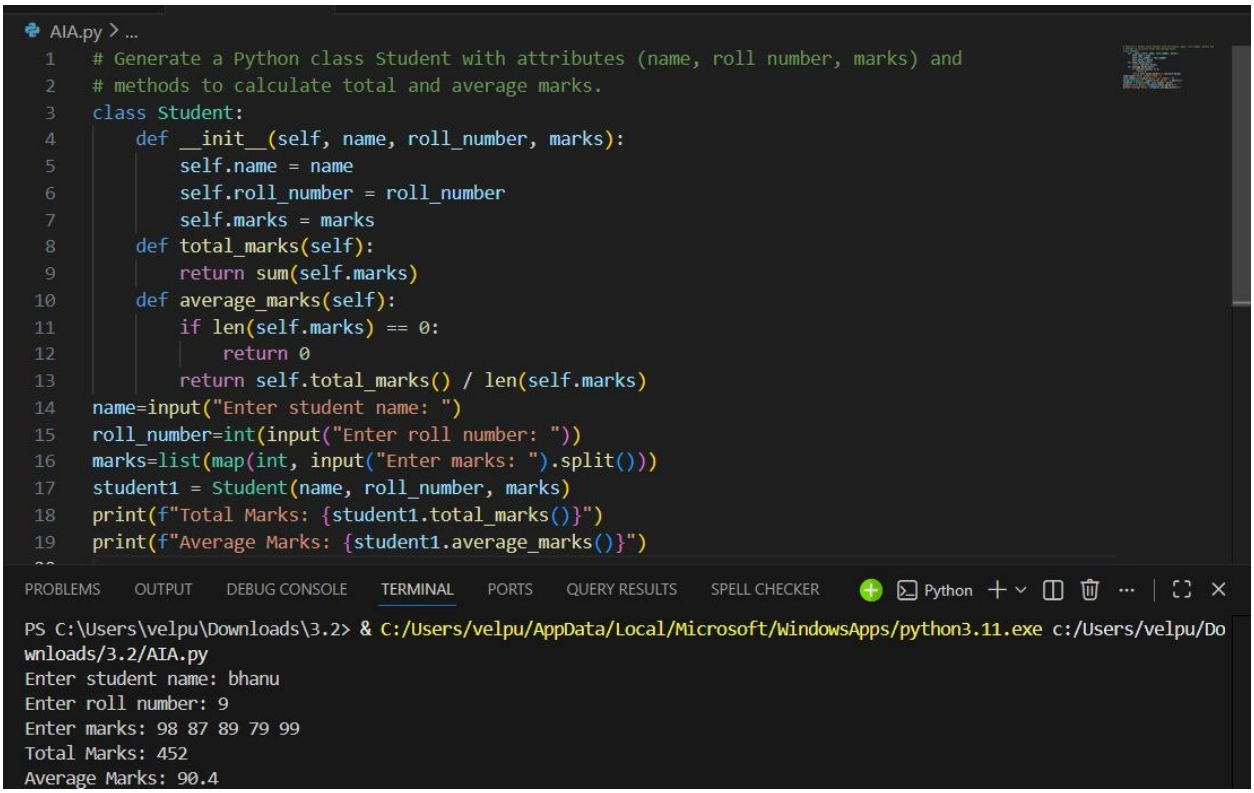


Task 4: Use an AI code completion tool to generate a Python class for managing student details.

Prompt:

Generate a Python class Student with attributes (name, roll number,marks) and methods to calculate total and average marks.

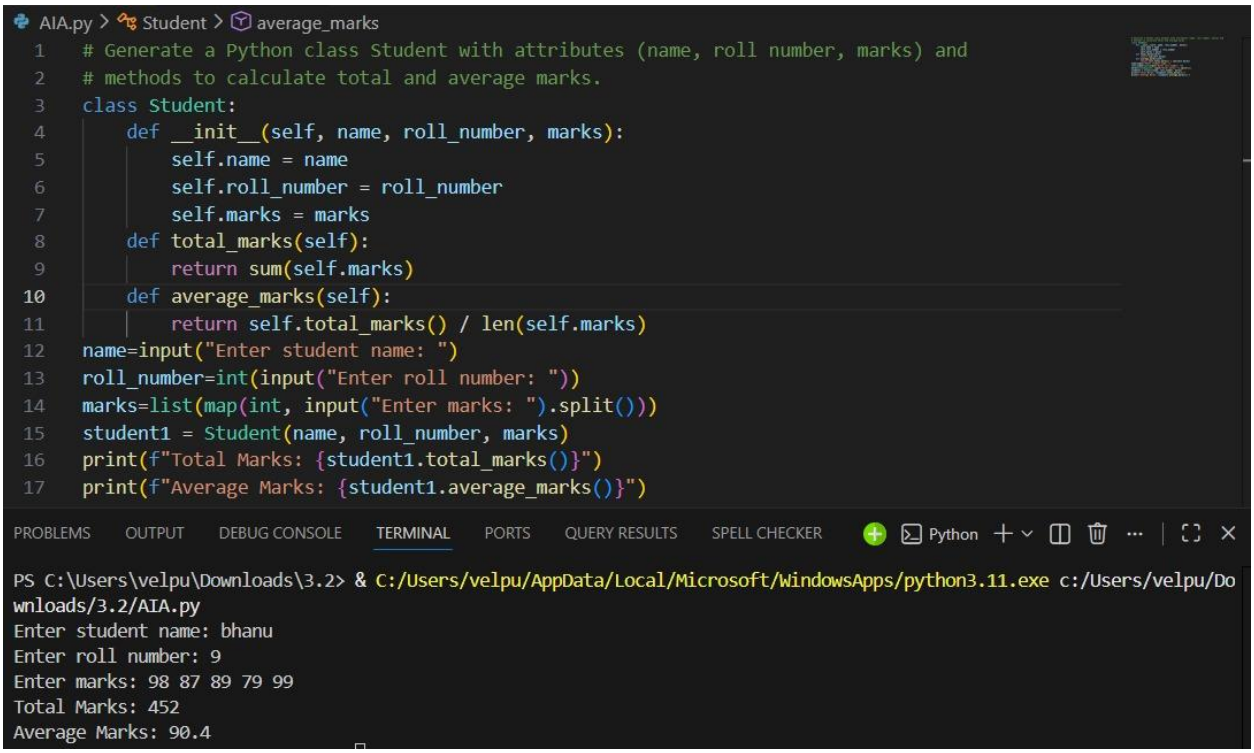
Program and Output:(Manual Improvement)



Justification:

- The manual improvement adds a conditional check in the average_marks() method to handle the case where the marks list is empty.
- Without this check, the original AI-generated code may cause a runtime error (division by zero) if no marks are entered.
- By returning 0 when the marks list is empty, the improved version makes the program more robust, error-free, and reliable.
- This modification reflects responsible use of AI-generated code by reviewing, testing, and improving it to handle edge cases safely.

Program and Output:(AI generated)



Justification:

- The AI-generated code correctly defines a Student class with appropriate attributes (name, roll_number, marks) and methods to calculate total marks and average marks.
- It efficiently uses built-in Python functions like sum() and len() to perform calculations, making the code simple, readable, and easy to understand.
- The logic works correctly when valid input (non-empty marks list) is provided, demonstrating proper use of classes, loops (implicit in sum()), and object-oriented programming concepts.

Conclusion

The AI-generated code provided a correct base implementation, while the manual improvement enhanced its stability and correctness, showing the importance of human oversight in AI-assisted coding.

Task 5: Use an AI tool to generate a complete Python program using classes, loops, and conditionals together.

Prompt:

Generate a Python program for a simple bank account system using class, loops, and conditional statements.

Program :

AIA.py > _

```python
# Generate a Python program for a simple bank account system using class, loops,
# and conditional statements.
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:.2f}")
        else:
            print("Deposit amount must be positive.")
    def withdraw(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew: ${amount:.2f}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")
    def get_balance(self):
        return self.balance
def main():
    print("Welcome to the Simple Bank Account System")
    account_holder = input("Enter account holder name: ")
    account = BankAccount(account_holder)
    while True:
        print("\nMenu:")
        print("1. Deposit")
        print("2. Withdraw")
        print("3. Check Balance")
        print("4. Exit")
        choice = input("Choose an option (1-4): ")
        if choice == '1':
            amount = float(input("Enter amount to deposit: "))
            account.deposit(amount)
        elif choice == '2':
            amount = float(input("Enter amount to withdraw: "))
            account.withdraw(amount)
        elif choice == '3':
            balance = account.get_balance()
            print(f"Current balance: ${balance:.2f}")
        elif choice == '4':
            print("Thank you for using the Simple Bank Account System. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
if __name__ == "__main__":
    main()
```

Output:

```
Welcome to the Simple Bank Account System
Enter account holder name: sai pallavi

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 1
Enter amount to deposit: 1000
Deposited: $1000.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 2
Enter amount to withdraw: 200
Withdrew: $200.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance

Menu:
1. Deposit
2. Withdraw
3. Check Balance
Menu:
1. Deposit
2. Withdraw
3. Check Balance
1. Deposit
2. Withdraw
3. Check Balance
3. Check Balance
4. Exit
Choose an option (1-4): 3
Current balance: $800.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 4
Thank you for using the Simple Bank Account System. Goodbye!
```