ASSIGMENT-6.1

Name:M Varshith Reddy

HTNO:2303A52087

Batch:40

Task Description #1 (AI-Based Code Completion for Loops)
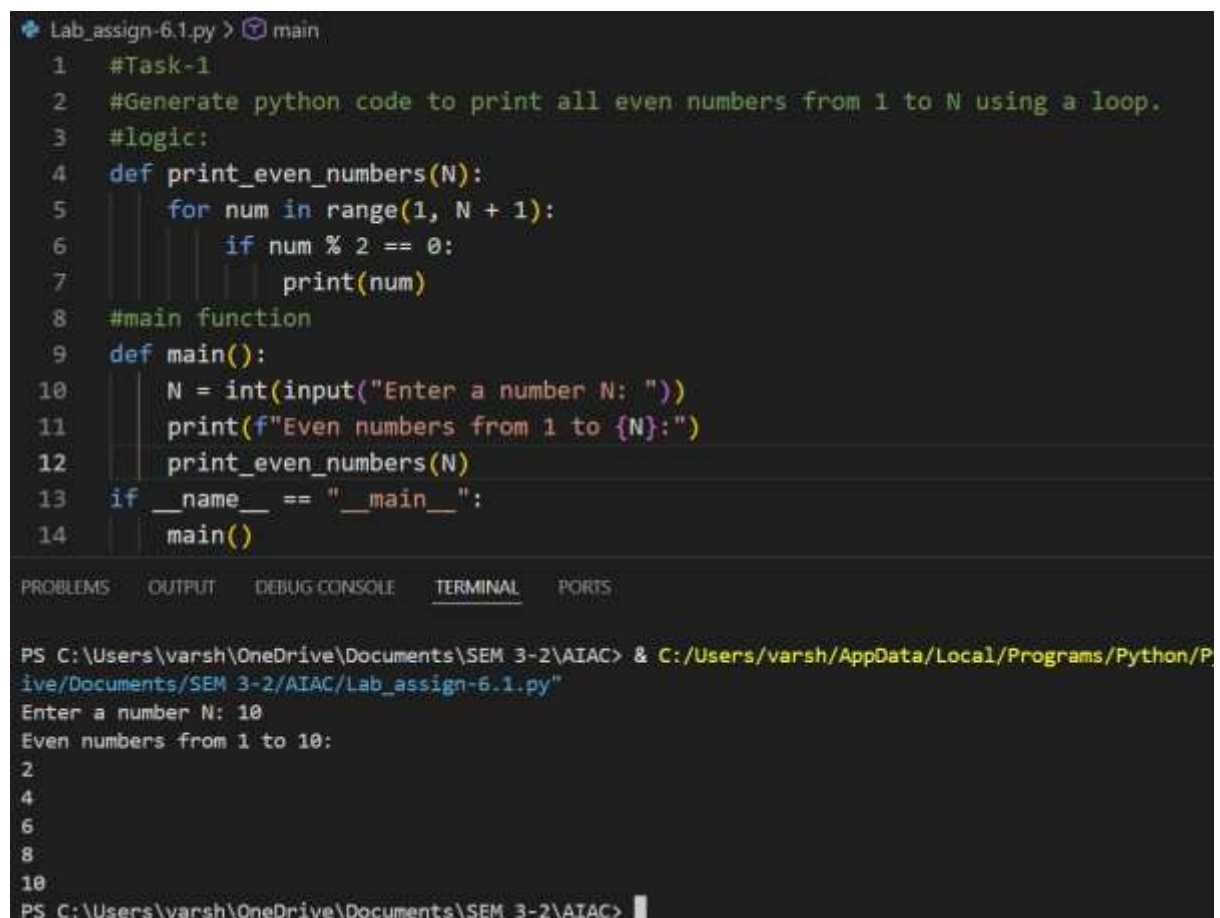
Task: Use an AI code completion tool to generate a loop-based

program.

Prompt:

"Generate Python code to print all even numbers between 1 and N

using a loop."

Expected Output:

• AI-generated loop logic.

• Identification of loop type used (for or while).

• Validation with sample inputs.

```python
#Task-1
#Generate python code to print all even numbers from 1 to N using a loop.
#logic:
def print_even_numbers(N):
    for num in range(1, N + 1):
        if num % 2 == 0:
            print(num)
#main function
def main():
    N = int(input("Enter a number N: "))
    print(f"Even numbers from 1 to {N}:")
    print_even_numbers(N)
if __name__ == "__main__":
    main()
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC> & C:/Users/varsh/AppData/Local/Programs/Python/P
ive/Documents/SEM 3-2/AIAC/Lab_assign-6.1.py"
Enter a number N: 10
Even numbers from 1 to 10:
2
4
6
8
10
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC>
```

Explanation: This task uses an AI code completion tool to generate a Python program based on loops. The AI generates loop logic (using a for or while loop) to print all even numbers between 1 and a given number N.
The loop iterates through the range and checks for even numbers.
Sample input values are used to validate that the output is correct.

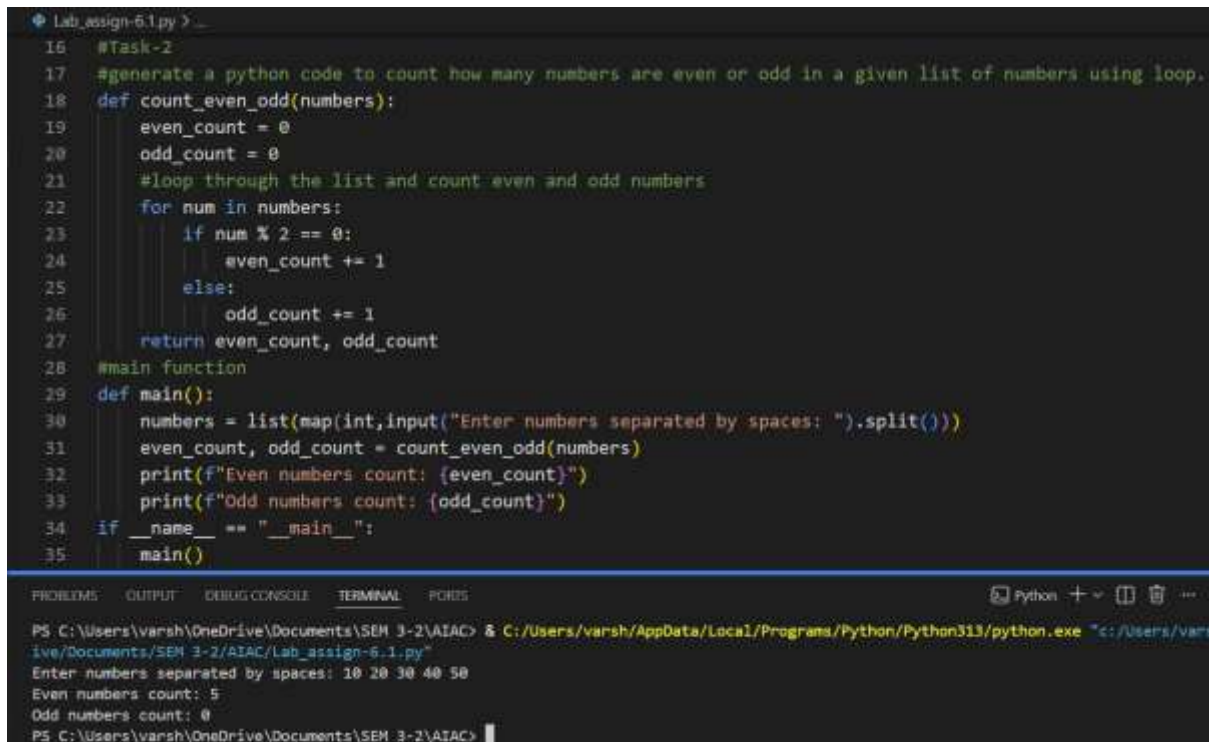Task Description #2 (AI-Based Code Completion for Loop with

Conditionals)

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."

Expected Output:

• AI-generated code using loop and if condition.

• Correct count validation.

• Explanation of logic flow.

```
Lab_assign-6.1.py >
16    #Task-2
17    #generate a python code to count how many numbers are even or odd in a given list of numbers using loop.
18    def count_even_odd(numbers):
19        even_count = 0
20        odd_count = 0
21        #loop through the list and count even and odd numbers
22        for num in numbers:
23            if num % 2 == 0:
24                even_count += 1
25            else:
26                odd_count += 1
27        return even_count, odd_count
28    #main function
29    def main():
30        numbers = list(map(int,input("Enter numbers separated by spaces: ").split()))
31        even_count, odd_count = count_even_odd(numbers)
32        print(f"Even numbers count: {even_count}")
33        print(f"Odd numbers count: {odd_count}")
34    if __name__ == "__main__":
35        main()
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                          Python + ∨ ⊡ 🗑 ⋯

PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC> & C:/Users/varsh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/vars
ive/Documents/SEM 3-2/AIAC/Lab_assign-6.1.py"
Enter numbers separated by spaces: 10 20 30 40 50
Even numbers count: 5
Odd numbers count: 0
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC>
```

Explanation: This task uses AI to generate Python code that combines a loop with conditional statements.
The loop iterates through a list of numbers, and if-else conditions check whether each number is even or odd.

Counters are updated accordingly to keep track of totals.
Sample inputs are used to verify that the logic produces correct counts.

Task Description #3 (AI-Based Code Completion for Class

Attributes Validation)

Task: 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."

Expected Output:

• AI-generated class with validation logic.

• Verification of condition handling.

• Test cases for valid and invalid inputs.

```python
#Task-3
#generate a python class user that validates age and email using conditional statements.
#Intialize
class User:
    def __init__(self, age, email):
        self.age = age
        self.email = email
    #validation methods
    #Age validation between 0 and 120
    def validate_age(self):
        if 0 < self.age < 120:
            return True
        return False
    #Email validation for basic format
    def validate_email(self):
        if "@" in self.email and "." in self.email.split("@")[-1]:
            return True
        return False
#main function
def main():
    age = int(input("Enter your age: "))
    email = input("Enter your email: ")
    user = User(age, email)
    print("Age valid:", user.validate_age())
    print("Email valid:", user.validate_email())
if __name__ == "__main__":
    main()
```

```
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC> & C:/Users/varsh/AppData/Local/Programs/Pytho
ive/Documents/SEM 3-2/AIAC/Lab_assign-6.1.py"
Enter your age: 20
Enter your email: varshith@gmail.com
Age valid: True
Email valid: True
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC>
```

Explanation: In this task, AI generates a Python class that validates user inputs using conditional statements.
The User class checks whether the age is valid and if the email follows a proper format.
Conditions handle both valid and invalid cases gracefully.
Test cases confirm that the validation logic works correctly.

Task Description #4 (AI-Based Code Completion for Classes)

Task: 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."

Expected Output:

• AI-generated class code.

• Verification of correctness and completeness of class

structure.

• Minor manual improvements (if needed) with justification.

```python
Lab_assign-6.1.py > main
64
65    #Task-4
66    #Generate a python class student(name,roll no,marks).
67    class Student:
68        def __init__(self, name, roll_no, marks):
69            self.name = name
70            self.roll_no = roll_no
71            self.marks = marks
72        def display_info(self):
73            print(f"Name: {self.name}")
74            print(f"Roll No: {self.roll_no}")
75            print(f"Marks: {self.marks}")
76        #method to calculate total and average marks.
77        def calculate_total_average(self):
78            total = sum(self.marks)
79            average = total / len(self.marks) if self.marks else 0
80            return total, average
81    #main function
82    def main():
83        name = input("Enter student name: ")
84        roll_no = input("Enter roll number: ")
85        marks = list(map(int, input("Enter marks separated by spaces: ").split()))
86        student = Student(name, roll_no, marks)
87        student.display_info()
88        total, average = student.calculate_total_average()
89        print(f"Total Marks: {total}")
90        print(f"Average Marks: {average:.2f}")
91    if __name__ == "__main__":
92        main()
```

```
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC> & C:/Users/varsh/AppData/Local/
ive/Documents/SEM 3-2/AIAC/Lab_assign-6.1.py"
Enter student name: varshith
Enter roll number: 2087
Enter marks separated by spaces: 10 15 30 50
Name: varshith
Roll No: 2087
Marks: [10, 15, 30, 50]
Total Marks: 105
Average Marks: 26.25
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC>
```

Explanation: This task involves generating a Python class to manage student details using AI assistance.

The Student class includes attributes such as name, roll number, and marks.

Methods are implemented to calculate total and average marks.

Minor improvements may be added to enhance clarity, efficiency, or error handling.

Task Description 5 (AI-Assisted Code Completion Review)

Task: 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."

Expected Output:

• Complete AI-generated program.

• Identification of strengths and limitations of AI suggestions.

• Reflection on how AI assisted coding productivity.

```python
#Task-5
#generate a python code for simple bank account system using loops and conditional statements.
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance
    #method to deposit amount
    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.")
    #method to withdraw amount
    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew: {amount}. New Balance: {self.balance}")
        else:
            print("Insufficient funds or invalid withdrawal amount.")
    #method to display balance
    def display_balance(self):
        print(f"Account Holder: {self.account_holder}, Balance: {self.balance}")
#main function
def main():
    #Welcome message
    print("Welcome to the Simple Bank Account System")
```

```python
    #create bank account
    account_holder = input("Enter account holder name: ")
    account = BankAccount(account_holder)
    #loop for user actions
    while True:
        print("\nOptions: 1. Deposit 2. Withdraw 3. Display Balance 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':
            account.display_balance()
        elif choice == '4':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
if __name__ == "__main__":
    main()
```

```
PS C:\Users\varsh\OneDrive\Documents\SEM 3-2\AIAC> & C:/Users/varsh/AppData/Local/Programs/Python/Python3
ive/Documents/SEM 3-2/AIAC/Lab_assign-6.1.py"
Welcome to the Simple Bank Account System
Enter account holder name: varshith

Options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option (1-4): 1
Enter amount to deposit: 10000
Deposited: 10000.0. New Balance: 10000.0

Options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option (1-4): 2
Enter amount to withdraw: 5000
Withdrew: 5000.0. New Balance: 5000.0

Options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option (1-4): 
```

Explanation: This task uses AI to generate a complete Python program that integrates classes, loops, and conditionals.

The bank account system handles operations like deposit, withdrawal, and balance checking.

The strengths and limitations of AI-generated code are analyzed.

Finally, reflection highlights how AI improves coding speed and productivity.