

Assignment_6.1

T.Rakshitha

2303a51172

Batch:18

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.

Prompt:

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

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files like "ord reversed.py", "#count vowels from a sentence.py", "lab.txt", "1 to N using a loop.py", "even and odd.py", and "armstrong.py".
- Code Editor:** Displays the following Python code:

```
#Generate Python code to print all even numbers between 1 and N using a loop.
N = int(input("Enter a number N: "))
for i in range(2, N + 1, 2):
    print(i)
```
- Terminal:** Shows the command "python.exe "c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/1 to N using a loop.py"" and the output:

```
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & C:/Users/raksh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/1 to N using a loop.py"
Enter a number N: 6
2
4
6
```
- Right Sidebar:** Shows a file tree with several "Python" entries.

Analysis:

Correct selection of a for loop with step size optimization

Logic is clear and efficient

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.

Prompt:

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

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows files like "ord reversed.py", "#count vowels from a sentence.py", "#read a list of numbers.py", "#read a number and check if it is prime co...", "#read a number and check if it is prime co...", "#read a.txt file and count number of lines in...", "#read input from user number to check if it ...", "#read tuple from user and print sum of eve...", "#take input from user and check even or od...", "#take input from user and check if it is a pe...", "#take input from user and check if it is arms...", "#take input from user and check if it is leap ...", "#take input from user and convert centimet... ", "#take input from user and print factorial of ...", "#take input from user as string and print w...", "# 1 to N using a loop.py", "armstrong.py", "even and odd.py", and "lab.txt".
- Editor:** Displays the "even and odd.py" file with the following code:

```
1 #Generate Python code to count how many numbers in a list are even and odd.
2 nums = list(map(int, input("Enter numbers: ").split()))
3
4 even = 0
5 odd = 0
6
7 for n in nums:
8     if n % 2 == 0:
9         even += 1
10    else:
11        odd += 1
12
13 print("Even count:", even)
14 print("Odd count:", odd)
15
```
- Terminal:** Shows the command line output:

```
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & C:/Users/raksh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/even and odd.py"
Enter numbers: 1 2 3 4 5 6 7 8
Even count: 4
Odd count: 4
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT>
```
- Bottom Status Bar:** Shows system information including weather (30°C, Sunny), battery level (0%), and system time (15:42, 02-02-2026).

Analysis:

Accurate counting logic

Easy to extend for other classifications

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.

Prompt:

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

The screenshot shows the Visual Studio Code interface with the AI ASSISTANT extension active. The code editor displays the following Python code:

```
#Generate a Python class User that validate age and email using conditional statements
class User:
    def __init__(self, age, email):
        if age <= 0:
            raise ValueError("Age must be positive")
        if "@" not in email or "." not in email:
            raise ValueError("Invalid email format")
        self.age = age
        self.email = email
# Test cases
try:
    u1 = User(25, "user@example.com")
    print("Valid user created")
except ValueError as e:
    print(e)
try:
    u2 = User(-5, "user@example.com")
    print("Valid user created")
except ValueError as e:
    print(e)
try:
    u3 = User(30, "userexample.com")
    print("Valid user created")
except ValueError as e:
    print(e)
```

The terminal at the bottom shows the command run: AI ASSISTANT/Generate a Python class User that validate age and email using conditional statements and the resulting output: Valid user created, Age must be positive, Invalid email format.

Analysis:

Correct use of constructor for validation

Clear error handling via exceptions

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.

Prompt:

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

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a folder named "AI ASSISTANT" containing several Python files like "count vowels from a sentence.py", "generate a Python class User that valid ag...", "read a list of numbers.py", etc., and a file "calculate total and avg marks.py" which is currently selected.
- Code Editor:** Displays the generated Python code for a "Student" class. The code includes an __init__ method to initialize name, roll_no, and marks, and two methods to calculate total and average marks. A test case at the bottom creates a Student object and prints its total and average marks.

```
#Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and avg marks.py
class Student:
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks # list of marks

    def total_marks(self):
        return sum(self.marks)

    def average_marks(self):
        return self.total_marks() / len(self.marks)

# Test case
s1 = Student("Ravi", 101, [80, 75, 90])
print("Total:", s1.total_marks())
print("Average:", s1.average_marks())
```

- Terminal:** Shows the command "python.exe "c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/calculate total and avg marks.py"" being run, with output showing "Total: 245" and "Average: 81.66666666666667".
- Status Bar:** Shows the current branch is "master", the file is "calculate total and avg marks.py", the line is "Ln 18, Col 38", and the column is "Spaces: 4". It also shows the Python version is 3.13.2 and the date is 02-02-2026.

Analysis:

Logical separation of data and behavior

Reusability improved by method reuse

Code is readable and maintainable

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:

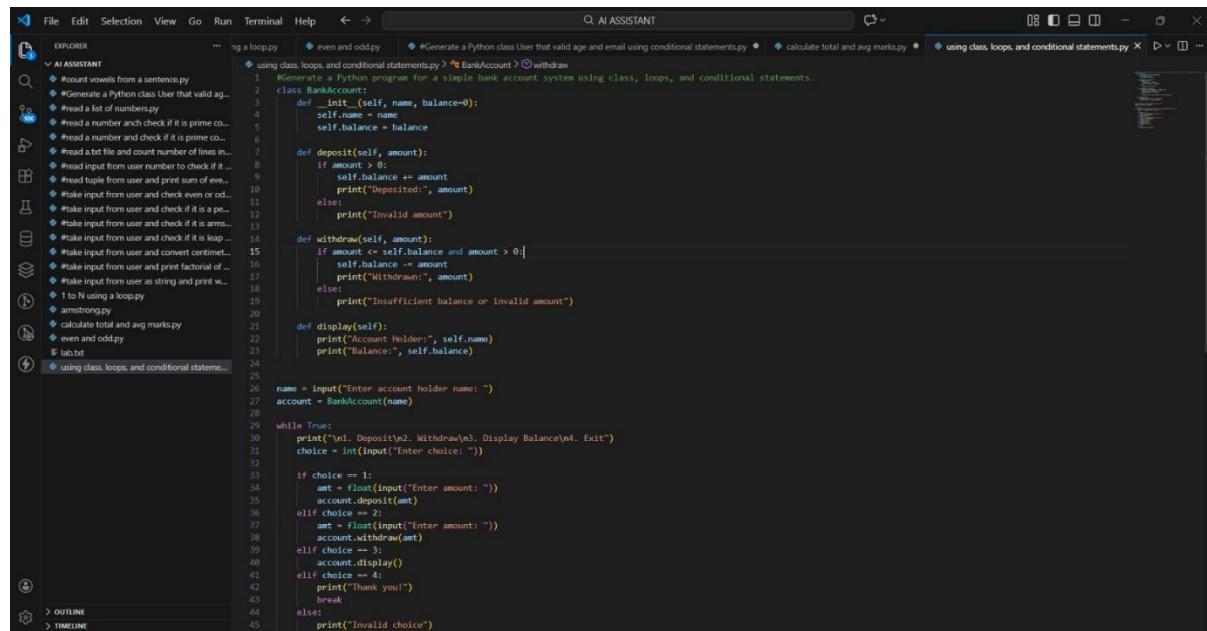
“Generate a Python program for a simple bank account system using class, loops, and conditional statements.”

Expected Output:

- Complete AI-generated program.
- Identification of strengths and limitations of AI suggestions.
- Reflection on how AI assisted coding productivity.

Prompt:

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



The screenshot shows a code editor interface with the title bar "Q AI ASSISTANT". The left sidebar is labeled "EXPLORER" and lists several Python files: "using a loop.py", "even and odd.py", "#Generate a Python class User that valid age and email using conditional statements.py", "withdraw", "#count vowels from a sentence.py", "#read a lot of numbers.py", "#read a number and check if it is prime co...", "#read a number and check if it is prime co...", "#read a file and count number of lines in...", "#read input from user number to check if it ...", "#read tuple from user and print sum of even...", "#take input from user and check even or ou...", "#take input from user and check if it is a pe...", "#take input from user and check if it is armstr...", "#take input from user and check if it is leap ...", "#take input from user and convert centime...", "#take input from user and print factorial of ...", "#take input from user as string and print w...", "1 to N using a loop.py", "armstrong.py", "calculate total and avg marks.py", "even and odd.py", "lab.txt", and "using class, loops, and conditional stateme...". The main code area contains the following Python code:

```
1 #Generate a Python program for a simple bank account system using class, loops, and conditional statements.
2
3 class BankAccount:
4     def __init__(self, name, balance=0):
5         self.name = name
6         self.balance = balance
7
8     def deposit(self, amount):
9         if amount > 0:
10             self.balance += amount
11             print("Deposited:", amount)
12         else:
13             print("Invalid amount")
14
15     def withdraw(self, amount):
16         if amount < self.balance and amount > 0:
17             self.balance -= amount
18             print("Withdrawn:", amount)
19         else:
20             print("Insufficient balance or invalid amount")
21
22     def display(self):
23         print("Account Holder:", self.name)
24         print("Balance:", self.balance)
25
26 name = input("Enter account holder name: ")
27 account = BankAccount(name)
28
29 while True:
30     print("\n1. Deposit\n2. Withdraw\n3. Display Balance\n4. Exit")
31     choice = int(input("Enter choice: "))
32
33     if choice == 1:
34         amt = float(input("Enter amount: "))
35         account.deposit(amt)
36     elif choice == 2:
37         amt = float(input("Enter amount: "))
38         account.withdraw(amt)
39     elif choice == 3:
40         account.display()
41     elif choice == 4:
42         print("Thank you!")
43         break
44     else:
45         print("Invalid choice")
```

Output:

```
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> <input type="text" value="C:\Users\raksh\AppData\Local\Programs\Python\Python311\python.exe c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/using class, loops, and conditional statements.py" data-bbox="117 110 729 373" style="width: 100%; height: 100%; border: none; background-color: transparent; font-family: inherit; font-size: inherit; margin: 0; padding: 0;"/>
```

Enter account holder name: Rakshitha

1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 1
Enter amount: 5000
Deposited: 5000.0

1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 2
Enter amount: 1000
Withdrawn: 1000.0

1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 3
Account holder: Rakshitha
Balance: 4000.0

1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 4
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 4
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 4
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Enter choice: 4
Thank you!

```
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> Thank you!  
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> []
```

Analysis:

Real-world inspired example

Demonstrates program flow clearly