

## 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 displays a Windows desktop with a VS Code editor open. The editor has a dark theme and shows a Python file named `1 to N using a loop.py`. The code in the file is as follows:

```
1 #Generate Python code to print all even numbers between 1 and N using a loop.
2 N = int(input("Enter a number N: "))
3 for i in range(2, N + 1, 2):
4     print(i)
5
```

The left sidebar of VS Code shows the Explorer view with a file tree containing:

- AI ASSISTANT
- #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 (selected)
- armstrong.py
- even and odd.py
- lab.txt

The bottom panel of VS Code shows the TERMINAL view. It contains the command prompt output for running the script:

```
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
6
```

The taskbar at the bottom of the screen shows various icons including the Start button, task view, and several open applications like Chrome, File Explorer, and VS Code. The system clock in the bottom right corner indicates the date is 02-02-2024 and the time is 15:42.

**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:**

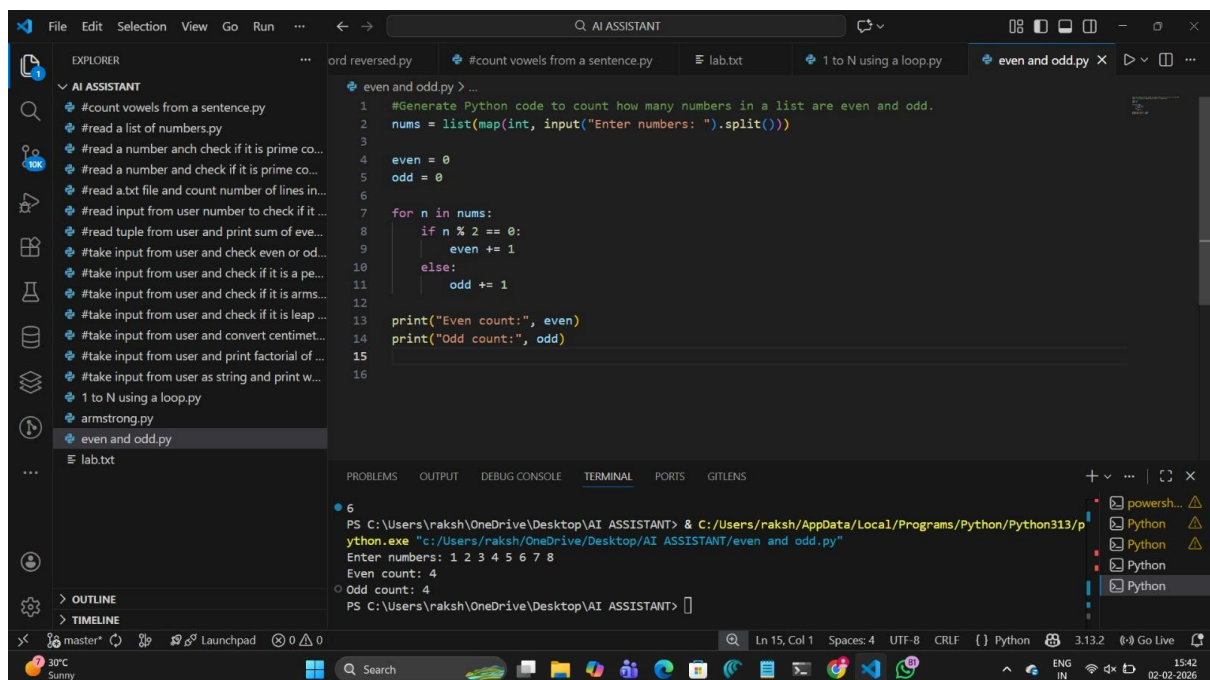
“Generate Python code to count how many numbers in a list are even and odd.”

**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 a VS Code editor with a file named 'even and odd.py'. The code is as follows:

```
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
16
```

The terminal output shows the execution of the code:

```
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & C:/Users/raksh/AppData/Local/Programs/Python/Python313/p
ython.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>
```

**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:**

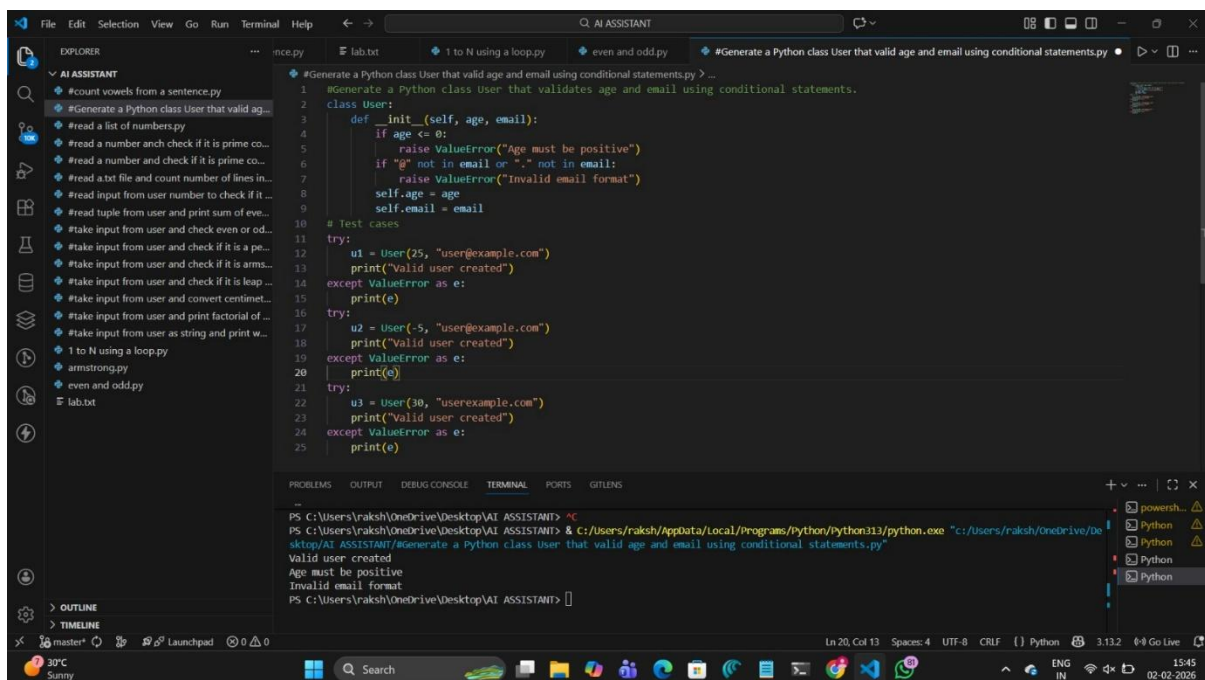
“Generate a Python class User that validates age and email using conditional statements.”

**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.



```
#Generate a Python class User that valid age and email using conditional statements.py
1 #Generate a Python class User that validates age and email using conditional statements.
2 class User:
3     def __init__(self, age, email):
4         if age <= 0:
5             raise ValueError("Age must be positive")
6         if "g" not in email or "." not in email:
7             raise ValueError("Invalid email format")
8         self.age = age
9         self.email = email
10
11 # Test cases
12 try:
13     u1 = User(25, "user@example.com")
14     print("Valid user created")
15 except ValueError as e:
16     print(e)
17
18 try:
19     u2 = User(-5, "user@example.com")
20     print("Valid user created")
21 except ValueError as e:
22     print(e)
23
24 try:
25     u3 = User(30, "userexample.com")
26     print("Valid user created")
27 except ValueError as e:
28     print(e)
```

PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> <C>  
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & C:/Users/raksh/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/raksh/OneDrive/Desktop/AT ASSISTANT/#Generate a Python class User that valid age and email using conditional statements.py"  
Valid user created  
Age must be positive  
Invalid email format  
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT>

**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:**

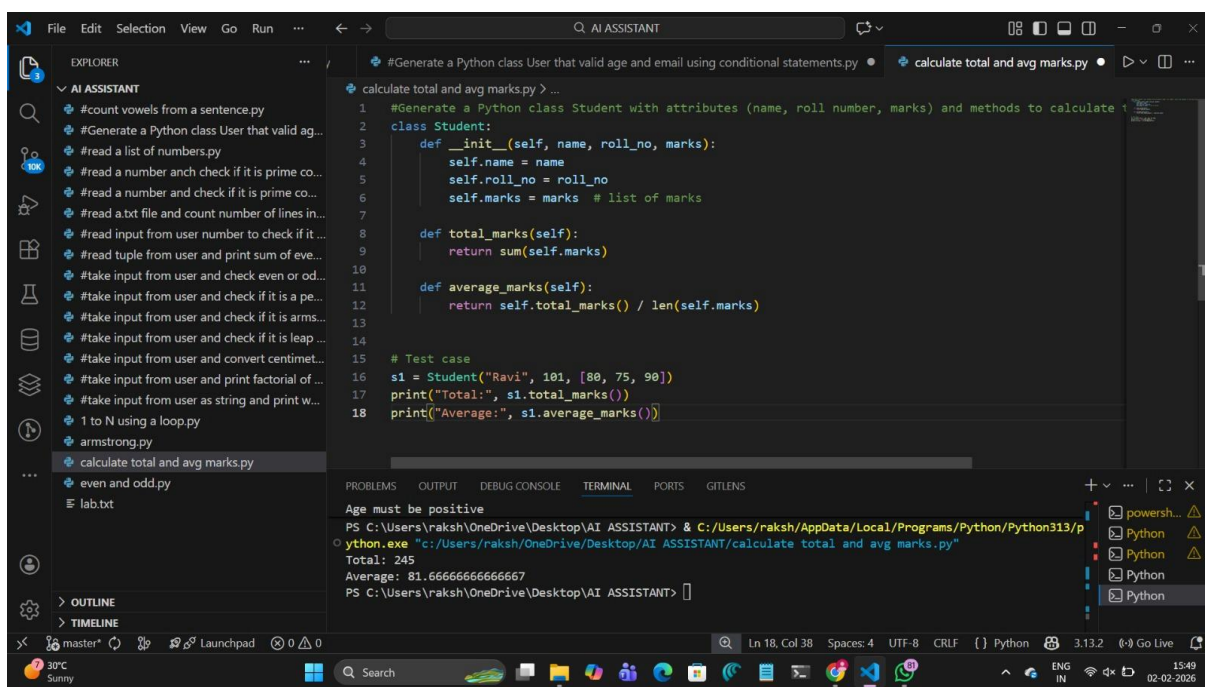
“Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and average marks.”

**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 with a file explorer on the left, a main code editor, and a terminal at the bottom. The file explorer shows a list of files, including 'calculate total and avg marks.py'. The main code editor displays the following Python code:

```
1 #Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and avg marks.py
2 class Student:
3     def __init__(self, name, roll_no, marks):
4         self.name = name
5         self.roll_no = roll_no
6         self.marks = marks # list of marks
7
8     def total_marks(self):
9         return sum(self.marks)
10
11     def average_marks(self):
12         return self.total_marks() / len(self.marks)
13
14 # Test case
15 s1 = Student("Ravi", 101, [80, 75, 90])
16 print("Total:", s1.total_marks())
17 print("Average:", s1.average_marks())
```

The terminal at the bottom shows the output of the code execution:

```
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/calculate total and avg marks.py"
Total: 245
Average: 81.66666666666667
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT>
```

**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:**

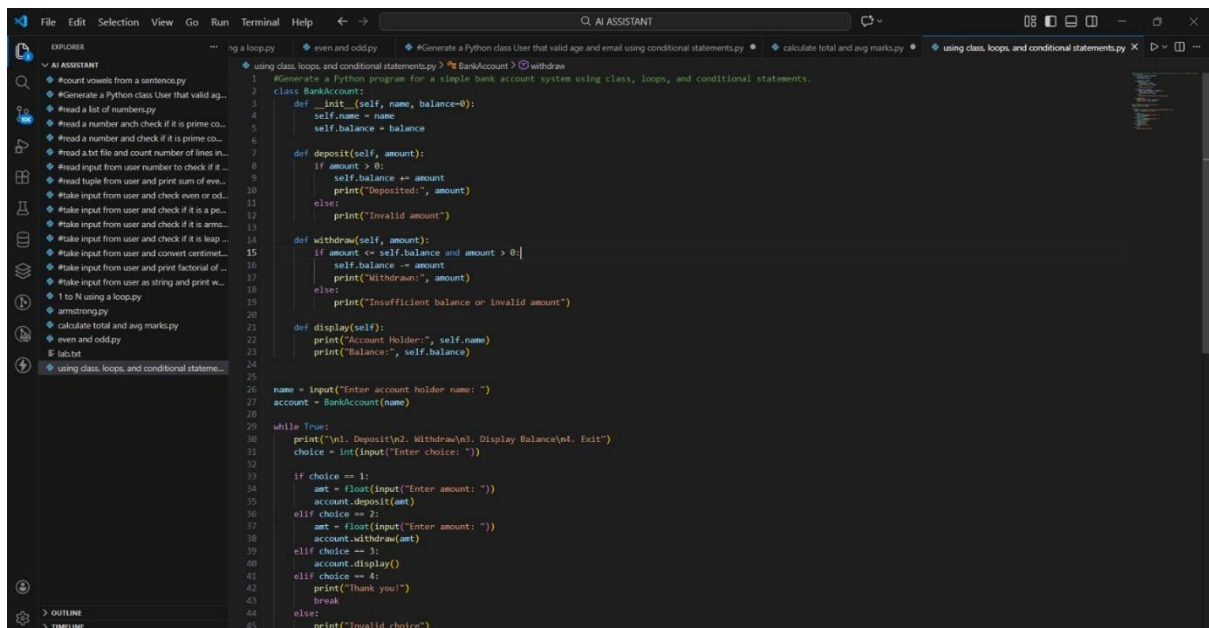
“Generate a Python program for a simple bank account system using class, loops, and conditional statements.”

**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.



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

## Output:

```
PS C:\Users\raksh\Desktop\AI ASSISTANT>
PS C:\Users\raksh\Desktop\AI ASSISTANT> & C:\Users\raksh\AppData\Local\Programs\Python\Python311\python.exe "c:\Users\raksh\Desktop\AI ASSISTANT\using class, loops, and conditional statements.py"
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

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

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

## Analysis:

Real-world inspired example

Demonstrates program flow clearly