

ASSIGNMENT-6.1

A.KARTHIK

2303A51132

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

Code:

```
# Print all even numbers between 1 and N
N = int(input("Enter N: "))
for i in range(2, N + 1, 2):
    print(i)
```



```
PS C:\Users\smini\OneDrive\Desktop\ai> & 'C:\Users\smini\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\smini\.vscode\extensions\ms-python.debugpy-2025.18.0\bundled\libs\debugpy\launcher' '52334' '--'
Enter N: 15
2
4
6
8
10
12
14
```

Explanation Steps

1. Take input value **N** from the user.
2. Start loop from **2** (first even number).
3. Increase by **2 each time** using range(2, N+1, 2).
4. Print each number inside the loop.
5. Loop stops when value exceeds **N**.

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.

CODE:

```
# Count even and odd numbers in a list
numbers = list(map(int, input("Enter
numbers separated by space: ").split()))
even_count = 0
odd_count = 0
for
num in numbers:
    if num % 2 == 0:
        even_count += 1
    else:
        odd_count += 1
print("Even numbers:", even_count)
print("Odd numbers:", odd_count)
```

```
● PS C:\Users\srini\OneDrive\Desktop\ai> c;; cd 'c:\Users\srini\OneDrive\Desktop\ai'; & 'C:\Users\srini\AppData\Local\Programs\Python\Python
18.0\bundled\libs\debugpy\launcher' '54964' '--' 'c:\Users\srini\OneDrive\Desktop\6.1.2.py'
Enter numbers separated by space: 2 3 4
Even numbers: 2
Odd numbers: 1
● PS C:\Users\srini\OneDrive\Desktop\ai> c;; cd 'c:\Users\srini\OneDrive\Desktop\ai'; & 'C:\Users\srini\AppData\Local\Programs\Python\Python
18.0\bundled\libs\debugpy\launcher' '54981' '--' 'c:\Users\srini\OneDrive\Desktop\6.1.2.py'
Enter numbers separated by space: 2 3 4 5 6 7
Even numbers: 3
Odd numbers: 3
```

Explanation Steps

1. Take list input from the user.
2. Initialize two counters → even_count = 0, odd_count = 0.
3. Use **for loop** to check each number in the list.
4. If number % 2 == 0, increment even counter.
5. Else, increment odd counter.

6. Print final 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.

CODE:

```
class User:  
  
    def __init__(self, age, email):  
        if 1 <= age <= 120:  
            print("Valid age")  
        else:  
            print("Invalid age")  
            if  
                "@" in email and "." in email:  
                    print("Valid email")  
                else:  
                    print("Invalid email")  
  
age = int(input("Enter age: ")) email  
= input("Enter email: ") u =  
User(age, email)
```

```
PS C:\Users\srini\OneDrive\Desktop\ai> c:; cd 'c:\Users\srini\OneDrive\Desktop\ai'; & 'C:\Users\srini\OneDrive\Desktop\18.0\bundled\libs\debugpy\launcher' '55650' '--' 'c:\Users\srini\OneDrive\Desktop\6.1.3.py'  
Enter age: 23  
Enter email: user@gmai.com  
Valid age  
Valid email  
PS C:\Users\srini\OneDrive\Desktop\ai> c:; cd 'c:\Users\srini\OneDrive\Desktop\ai'; & 'C:\Users\srini\OneDrive\Desktop\18.0\bundled\libs\debugpy\launcher' '55650' '--' 'c:\Users\srini\OneDrive\Desktop\6.1.3.py'
```

Explanation Steps

1. Define a class **User**.
2. Constructor `__init__()` receives **age** and **email**.
3. Check age using condition `1 <= age <= 120`.
4. Print **Valid age** or **Invalid age**.
5. Check email contains '`@`' and '`.`'.
6. Print **Valid email** or **Invalid email**.
7. Take input from user and create User object.

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

CODE:

```
class Student:  
    def __init__(self,  
                 name, roll, marks):  
        self.name = name  
        self.roll = roll  
        self.marks = marks  
  
    def total(self):  
        return sum(self.marks)  
  
    def average(self):  
        return sum(self.marks) / len(self.marks)
```

```

name = input("Enter name: ") roll = input("Enter roll number: ") marks
= list(map(int, input("Enter marks separated by space: ").split())) s =
Student(name, roll, marks) print("Total Marks:", s.total())
print("Average Marks:", s.average())

```

```

● PS C:\Users\srini\OneDrive\Desktop\ai> cd 'c:\Users\srini\OneDrive\Desktop\ai'; & 'C:\U
18.0\bundled\libs\debugpy\launcher' '53112' '--' 'c:\Users\srini\OneDrive\Desktop\6.1.4.py'
Enter name: hello
Enter roll number: 34
Enter marks separated by space: 56 78 89
Total Marks: 223
Average Marks: 74.33333333333333
○ PS C:\Users\srini\OneDrive\Desktop\ai> cd 'c:\Users\srini\OneDrive\Desktop\ai'; & 'C:\U

```

Explanation Steps

1. Define class **Student** with attributes → name, roll, marks.
2. Store values using constructor `__init__()`.
3. `total()` method calculates sum of marks using `sum()`.
4. `average()` method calculates average using `total / number of subjects`.
5. Take input from user and create Student object.
6. Call methods to display total and average.

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."

CODE:

```

class BankAccount: def __init__(self,
balance=0): self.balance = balance def
deposit(self, amount): self.balance +=
amount print("Amount deposited
successfully") def withdraw(self, amount):
if amount <= self.balance: self.balance -

```

```

= amount      print("Amount withdrawn
successfully")

else:
    print("Insufficient balance")  def
check_balance(self):      print("Current
balance:", self.balance) account =
BankAccount() while True:
    print("\n1.Deposit 2.Withdraw 3.Check Balance 4.Exit")
choice = int(input("Enter choice: "))  if choice == 1:
    amt = float(input("Enter amount: "))
account.deposit(amt)  elif choice == 2:
    amt = float(input("Enter amount: "))
account.withdraw(amt)

elif choice == 3:
    account.check_balance()

elif choice == 4:
print("Thank you!")

break  else:
    print("Invalid choice")

```

```

PS C:\Users\sriini\OneDrive\Desktop\ai> c:; cd 'c:\Users\sriini\OneDrive\Desktop\ai'; & 'C:\Users\sriini\OneDrive\Desktop\Python\venv\Scripts\python.exe' 'c:\Users\sriini\OneDrive\Desktop\ai\6.1.5.py'
1.Deposit 2.Withdraw 3.Check Balance 4.Exit
Enter choice: 2
Enter amount: 1245
Insufficient balance

1.Deposit 2.Withdraw 3.Check Balance 4.Exit
Enter choice: 1
Enter amount: 1234
Amount deposited successfully

1.Deposit 2.Withdraw 3.Check Balance 4.Exit
Enter choice: []

```

Explanation Steps

1. Create class **BankAccount** with attribute **balance**.
2. **deposit()** adds money to **balance**.
3. **withdraw()** checks **balance** using **if condition** and subtracts money.

4. `check_balance()` displays current balance.
5. Use **while loop** to repeatedly show menu options.
6. Use **if–elif–else** to perform selected operation.
7. Loop stops when user chooses Exit.