

ASSESSMENT-6

NAME:B.Shivamani

ROLLNO:2303A52079

Question: 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.

#INPUT:

```
#write a program on Generate Python code to print all even numbers between 1 and N using a loop
n = int(input("Enter a number N: "))
print(f"Even numbers between 1 and {n} are:")
for i in range(2, n + 1, 2):
    print(i)
```

#OUTPUT:

```
PS C:\Users\SHIVA MANI\OneDrive\Documents\AIAC> & 'c:\Miniforge\python.exe' 'c:\Users\SHIVA MANI\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53889' '--' 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC\Ass-6.py'
Enter a number N: 4
Even numbers between 1 and 4 are:
2
4
```

Analysis:

- AI generated correct Python code using a loop.
- A for loop was used to iterate from 1 to N.
- Conditional logic identified even numbers.

- Output was correct for sample inputs.
- Code was simple and easy to understand.

Question: 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.

#INPUT:

```
#write a program Generate Python code to count how many numbers in a list are even and odd code
numbers = input("Enter a list of numbers separated by spaces: ")
num_list = list(map(int, numbers.split()))
even_count = 0
odd_count = 0
for num in num_list:
    if num % 2 == 0:
        even_count += 1
    else:
        odd_count += 1
print(f"Total even numbers: {even_count}")
print(f"Total odd numbers: {odd_count}")
```

#OUTPUT

```
● PS C:\Users\SHIVA MANI\OneDrive\Documents\AIAC> cd 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC'; & 'c:\MiniForge\python.exe' 'c:\Users\SHIVA MANI\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52971' '--' 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC\Ass-6.py'
Enter a list of numbers separated by spaces: 10
Total even numbers: 1
Total odd numbers: 0
```

Analysis:

- AI generated code using a loop and if-else.
- The list was iterated element by element.
- Even and odd numbers were counted accurately.
- Output matched expected results.
- Logic flow was clear and efficient.

Question: 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.

#INPUT:

```

#write a program Generate a Python class User that validates age and email using conditional statements
class User:
    def __init__(self, name, age, email):
        self.name = name
        self.age = age
        self.email = email

    def is_valid_age(self):
        return 0 < self.age < 120

    def is_valid_email(self):
        return "@" in self.email and "." in self.email.split("@")[-1]

    def validate_user(self):
        if not self.is_valid_age():
            return f"Invalid age: {self.age}. Age must be between 1 and 119."
        if not self.is_valid_email():
            return f"Invalid email: {self.email}. Email must contain '@' and a domain."
        return f"User {self.name} is valid."

# Test cases
user1 = User("Alice", 30, "alice@example.com")
user2 = User("Bob", 150, "bob@example.com")
user3 = User("Charlie", 25, "charlieexample.com")

print(user1.validate_user())
print(user2.validate_user())
print(user3.validate_user())

```

#OUTPUT:

```

● PS C:\Users\SHIVA MANI\OneDrive\Documents\AIAC> c;; cd 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC'; &
  'c:\Miniforge\python.exe' 'c:\Users\SHIVA MANI\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64
  \bundled\libs\debugpy\launcher' '54866' '--' 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC\Ass-6.py'
User Alice is valid.
Invalid age: 150. Age must be between 1 and 119.
○ Invalid email: charlieexample.com. Email must contain '@' and a domain.

```

Analysis:

- AI generated a User class with attributes.
- Conditional statements validated age and email.
- Valid and invalid inputs were handled correctly.
- Methods returned appropriate validation messages.
- Code structure was clean and readable.

Question: 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.

#INPUT:

```
#write a program Generate a Python class Student with attributes (name, roll number,marks) and m
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

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

    def average_marks(self):
        return self.total_marks() / len(self.marks) if self.marks else 0

# Example usage
student1 = Student("Alice", 101, [85, 90, 78, 92])
print(f"Total marks for {student1.name}: {student1.total_marks()}")
print(f"Average marks for {student1.name}: {student1.average_marks()}")
```

#OUTPUT:

```
PS C:\Users\SHIVA MANI\OneDrive\Documents\AIAC> c:; cd 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC'; &
'c:\MiniForge\python.exe' 'c:\Users\SHIVA MANI\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64
\bundled\libs\debugpy\launcher' '65497' '--' 'c:\Users\SHIVA MANI\OneDrive\Documents\AIAC\Ass-6.py'
Total marks for Alice: 345
Average marks for Alice: 86.25
```

Analysis:

- AI generated a Student class with required attributes.
- Methods calculated total and average marks.
- Class structure was correct and complete.

- Minor manual improvement avoided division errors.
- Code was reusable and well-organized.

Question: 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.

#INPUT:

```
#write a program Generate a Python program for a simple bank account system using class, loops,
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    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.")

    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.")

    def display_balance(self):
        print(f"Account holder: {self.account_holder}, Balance: ${self.balance}")
```

#OUTPUT:

```
PS C:\Users\SHIVA MANI\OneDrive\Documents\AIAC> cd 'c:\c:\MiniForge\python.exe' 'c:\Users\SHIVA MANI\OneDrive\Doc
Select an action:
1. Show account details
2. Deposit
3. Withdraw
4. Show balance
5. Exit

Enter choice (1-5): 2
Enter deposit amount: 20000
Deposited 20000.00.

Select an action:
1. Show account details
2. Deposit
3. Withdraw
4. Show balance
5. Exit

Enter choice (1-5): 4
Current balance: 20000.00

Select an action:
1. Show account details
2. Deposit
3. Withdraw
4. Show balance
5. Exit

Enter choice (1-5): 1
Owner: asmitha
4. Show balance
5. Exit
```

Analysis:

- AI generated a complete class-based program.
- Loops enabled continuous user interaction.
- Conditionals controlled deposits and withdrawals.
- Program produced correct balance updates.
- AI improved speed but human review was necessary.