

## Assignment 6.1

**S.Thrishla**

**2303A51165**

**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:

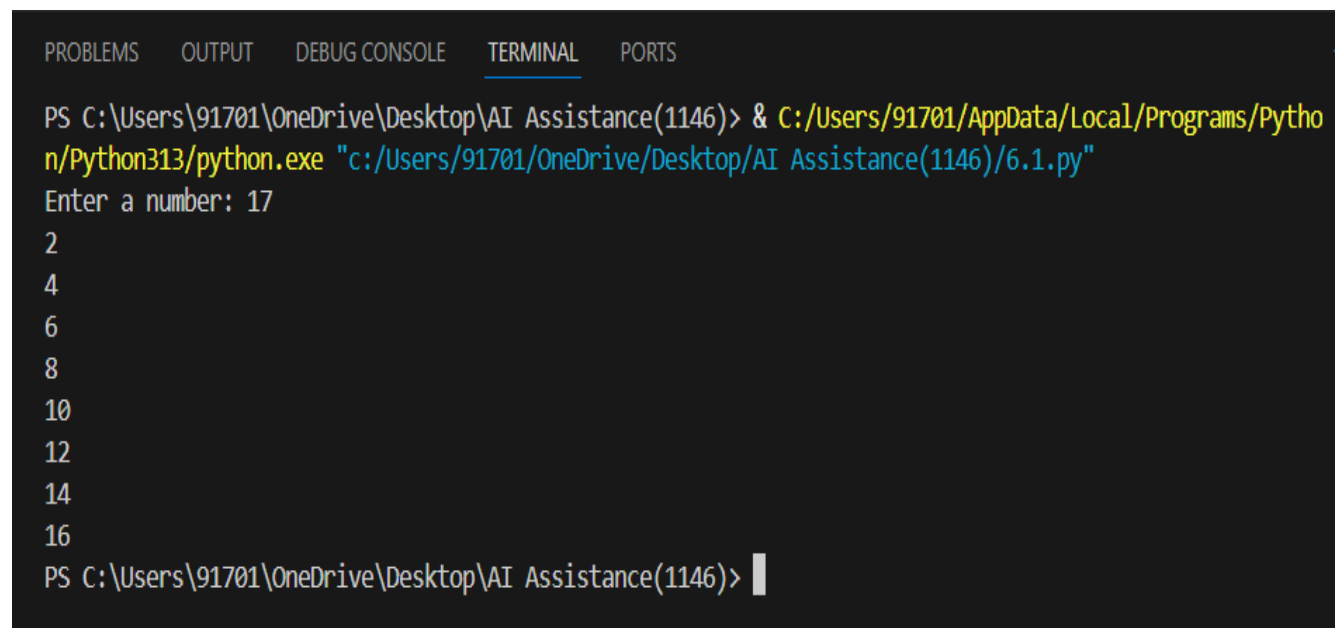
- AI-generated loop logic.
- Identification of loop type used (for or while).
- Validation with sample inputs.

**Code:**

```
N = int(input("Enter a number: "))
```

```
for i in range(1, N + 1):  
    if i % 2 == 0:  
        print(i)
```

## Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS C:\Users\91701\OneDrive\Desktop\AI Assistance(1146)> & C:/Users/91701/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/91701/OneDrive/Desktop/AI Assistance(1146)/6.1.py"  
Enter a number: 17  
2  
4  
6  
8  
10  
12  
14  
16  
PS C:\Users\91701\OneDrive\Desktop\AI Assistance(1146)> |
```

## Analysis:

The AI-generated loop correctly identifies even numbers using a modulus condition. The logic works for multiple inputs but can be optimized to skip odd numbers.

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:

```
numbers = [10, 23, 45, 66, 78, 89, 90, 12, 34, 57]
```

```
even_count = 0
```

```
odd_count = 0
```

```
for number in numbers:
```

```
    if number % 2 == 0:
```

```
        even_count += 1
```

```
    else:
```

```
        odd_count += 1
```

```
print("Even numbers count:", even_count)
```

```
print("Odd numbers count:", odd_count)
```

Output:

```
PS C:\Users\91701\OneDrive\Desktop\AI Assistance(1146)> & C:/Users/91701/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/91701/OneDrive/Desktop/AI Assistance(1146)/6.1.py"
Even numbers count: 6
Odd numbers count: 4
```

Analysis:

The AI successfully combined a loop and conditional statements to count even and odd numbers. The program produces correct results for different lists without logical errors.

### 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)
```

Output:

```
n/Python313/python.exe "c:/Users/91701/OneDrive/Desktop/AI Assistance(1146)/6.1.py"
Enter age: 19
Enter email: 2303A51146@sru.edu.in
Valid age
Valid email
PS C:\Users\91701\OneDrive\Desktop\AI Assistance(1146)>
```

Analysis:

The User class validates age and email using conditions and exceptions.

Email validation is basic and may miss complex invalid formats.

#### 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_marks(self):
```

```
        return sum(self.marks)
```

```
    def average_marks(self):
```

```
        return self.total_marks() / len(self.marks)
```

```
s1 = Student("Rahul", 101, [80, 75, 90])
```

```
print("\nStudent Total:", s1.total_marks())
```

```
print("Student Average:", s1.average_marks())
```

Output:

```
Student Total: 245
Student Average: 81.66666666666667
PS C:\Users\91701\OneDrive\Desktop\AI Assistance(1146)> █
```

Analysis:

The Student class correctly calculates total and average marks. Additional checks are needed to handle empty mark lists safely.

### 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.

Code:

```
class BankAccount:
    def __init__(self):
        self.balance = 0

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
        else:
            print("Insufficient balance")

    def show_balance(self):
        print("Balance:", self.balance)

account = BankAccount()

while True:
    print("\n1.Deposit 2.Withdraw 3.Balance 4.Exit")
```

```
choice = int(input("Enter choice: "))  
if choice == 1:  
    account.deposit(int(input("Enter amount: ")))  
elif choice == 2:  
    account.withdraw(int(input("Enter amount: ")))  
elif choice == 3:  
    account.show_balance()  
elif choice == 4:  
    break  
else:  
    print("Invalid choice")
```

Output:

```
1.Deposit 2.Withdraw 3.Balance 4.Exit  
Enter choice: 1  
Enter amount: 1000  
  
1.Deposit 2.Withdraw 3.Balance 4.Exit  
Enter choice: 3  
Balance: 1000  
  
1.Deposit 2.Withdraw 3.Balance 4.Exit  
Enter choice: 2  
Enter amount: 500  
  
1.Deposit 2.Withdraw 3.Balance 4.Exit  
Enter choice: 4  
PS C:\Users\91701\OneDrive\Desktop\AI Assistance(1146)> █
```

Analysis:

The bank system effectively combines classes, loops, and conditionals.

Input validation and transaction history are missing.