

AI ASSISTED CODING

ASSIGNMENT - 6.1

ROLL NO. 2303A510J0

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:

Write a Python program that takes an integer N as input and displays all even numbers from 1 up to N using a loop.

Code:

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

i = 2
while i <= N:
    print(i)
    i += 2
```

Input/Output:

```
Enter a number: 15
2
4
6
8
10
12
14
```

Loop Type Used

- **While loop**
- The loop starts from the first even number (2) and increments by 2 until N is reached.

Validation

- The program correctly prints all even numbers between 1 and N.
- Using a different loop type and logic ensures originality.

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:

Create a Python program that takes a list of integers and counts how many elements are even and how many are odd using a loop and conditional statements.

Code:

```
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)
```

Output:

```
Enter numbers separated by space: 3 6 9 12 15 20
Even numbers: 3
Odd numbers: 3
```

Logic Flow Explanation

1. The program first takes a list of integers as input.
2. Two counters are initialized to store the number of even and odd elements.
3. A `for` loop iterates through each number in the list.
4. The `if` condition checks whether the current number is divisible by 2.
5. If true, the even counter is incremented; otherwise, the odd counter is incremented.
6. Finally, the total count of even and odd numbers is displayed.

Validation

- The output correctly reflects the number of even and odd values in the input list.
 - The program works for any list of integers.
-

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:

Create a Python program that takes a list of integers and counts how many elements are even and how many are odd using a loop and conditional statements.

Code:

```
# Create a Python class named User that accepts age and email as input and
# validates them using conditional statements.
class User:
    def __init__(self, age, email):
        self.age = age
        self.email = email
        self.validate_age()
        self.validate_email()

    def validate_age(self):
        if not isinstance(self.age, int) or self.age < 0:
            raise ValueError("Age must be a non-negative integer.")

    def validate_email(self):
        if "@" not in self.email or "." not in self.email.split("@")[-1]:
            raise ValueError("Invalid email format.")

# Example usage:
try:
    user1 = User(25, "user@example.com")
    print("User1 created successfully.")
except ValueError as e:
    print(f"Error creating user1: {e}")

try:
    user2 = User(-5, "invalidemail.com")
    print("User2 created successfully.")
except ValueError as e:
    print(f"Error creating user2: {e}")
```

Output:

```
User1 created successfully.  
Error creating user2: Age must be a non-negative  
integer.
```

Condition Handling Verification

- Age must be between **0** and **120**
- Email must contain '@' and '.'

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, and marks, and include methods to calculate total and average marks.

Code:

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

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):

        if not self.marks:

            return 0

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

```
# Example usage:

student = Student("Alice", 101, [85, 90, 78, 92])

print("Total Marks:", student.total_marks()) # Output: Total Marks: 345

print("Average Marks:", student.average_marks()) # Output: Average Marks:
86.25
```

Output:

```
Total Marks: 345
Average Marks: 86.25
```

Verification of Class Structure

- Attributes: `name`, `roll_no`, `marks`
- Methods:
 - `total_marks()` → calculates sum
 - `average_marks()` → calculates average

Conclusion:

The code gives correct output based on the inputs as shown above.

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:

Design a Python program for a basic bank account system using a class, a looping menu, and conditional statements for user actions.

Code:

```
# Design a Python program for a basic bank account system using a class, a
looping menu, and conditional statements for user actions.

class BankAccount:

    def __init__(self, account_holder, initial_balance=0):

        self.account_holder = account_holder

        self.balance = initial_balance

    def deposit(self, amount):
```

```
        if amount > 0:

            self.balance += amount

            print(f"Deposited: ${amount:.2f}")

        else:

            print("Deposit amount must be positive.")

def withdraw(self, amount):

    if amount > 0:

        if amount <= self.balance:

            self.balance -= amount

            print(f"Withdrew: ${amount:.2f}")

        else:

            print("Insufficient funds.")

    else:

        print("Withdrawal amount must be positive.")

def check_balance(self):

    print(f"Current balance: ${self.balance:.2f}")

def main():

    print("Welcome to the Basic Bank Account System")

    account_holder = input("Enter account holder name: ")

    account = BankAccount(account_holder)
```

```
while True:

    print("\nMenu:")

    print("1. Deposit")

    print("2. Withdraw")

    print("3. Check Balance")

    print("4. Exit")

    choice = input("Choose an option (1-4): ")

    if choice == '1':

        amount = float(input("Enter amount to deposit: "))

        account.deposit(amount)

    elif choice == '2':

        amount = float(input("Enter amount to withdraw: "))

        account.withdraw(amount)

    elif choice == '3':

        account.check_balance()

    elif choice == '4':

        print("Exiting the system. Goodbye!")

        break

    else:

        print("Invalid choice. Please select a valid option.")

if __name__ == "__main__":

    main()
```

Output:

```
Welcome to the Basic Bank Account System
Enter account holder name: Rohit
```

```
Menu:
```

1. Deposit
2. Withdraw
3. Check Balance
4. Exit

```
Choose an option (1-4): 1
```

```
Enter amount to deposit: 10000
```

```
Deposited: $10000.00
```

```
Menu:
```

1. Deposit
2. Withdraw
3. Check Balance
4. Exit

```
Choose an option (1-4): 3
```

```
Current balance: $10000.00
```

```
Menu:
```

1. Deposit
2. Withdraw
3. Check Balance
4. Exit

```
Choose an option (1-4): 2
```

```
Enter amount to withdraw: 400
```

```
Withdrew: $400.00
```

Menu:

1. Deposit
2. Withdraw
3. Check Balance
4. Exit

Choose an option (1-4): 3

Current balance: \$9600.00

Menu:

1. Deposit
2. Withdraw
3. Check Balance
4. Exit

Choose an option (1-4): 4

Exiting the system. Goodbye!

Strengths

- Correct use of class, loop, and conditionals
- Clear and simple structure
- Saves coding time

Limitations

- No exception handling
- Basic validation only

Reflection

AI-assisted coding improves productivity by quickly generating a working program structure, allowing the programmer to focus on understanding logic and making improvements.