AI-Based Code Completion

2303A51892

Batch-28

Task Description #1: Loop-Based Program

Prompt

Generate Python code to print all even numbers between 1 and N using a loop.

AI-Generated Code

```
n = int(input("Enter a number: "))
for i in range(1, n + 1):
    if i % 2 == 0:
        print(i)
```

Loop Type Used

- **for loop**

Explanation

- range(1, n+1) iterates from 1 to N.
- i % 2 == 0 checks if a number is even.
- Even numbers are printed.

Sample Input

```
Enter a number: 10
```

Sample Output

```
2
4
6
```

```
8
10
```

Optimization (Manual Improvement)

```
for i in range(2, n + 1, 2):
    print(i)
```

Task Description #2: Loop with Conditionals

Prompt

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

AI-Generated Code

```
numbers = [1, 2, 3, 4, 5, 6]

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)
```

Explanation of Logic Flow

- Loop iterates through each element.
- if checks even condition.

- Counters increment accordingly.

Validation

For list [1, 2, 3, 4, 5, 6]

Output

```
Even numbers: 3

Odd numbers: 3
```

Task Description #3: Class Attributes Validation

Prompt

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

AI-Generated Code

```
class User:
    def __init__(self, age, email):
        self.age = age
        self.email = email
    def validate_age(self):
        return self.age >= 18
    def validate_email(self):
        return "@" in self.email and "." in self.email
```

Condition Handling Verification

- Age must be **18 or above**

- Email must contain @ and .

Test Cases

```
user1 = User(20, "test@gmail.com")

print(user1.validate_age())    # True

print(user1.validate_email())  # True


user2 = User(15, "invalidmail")

print(user2.validate_age())    # False

print(user2.validate_email())  # False
```

Works for both valid and invalid inputs

Task Description #4: Student Class

Prompt

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

AI-Generated Code

```
class Student:

    def __init__(self, name, roll_no, marks):

        self.name = name

        self.roll_no = roll_no

        self.marks = marks

    def total_marks(self):
        return sum(self.marks)
    def average_marks(self):
        return self.total_marks() / len(self.marks)
```

Verification

- Uses proper class structure
- Methods work correctly

Test Example

```
s1 = Student("Sai", 101, [80, 75, 90])
print("Total:", s1.total_marks())
print("Average:", s1.average_marks())
```

Minor Manual Improvement

Add marks validation:

```
if not self.marks:
    return 0
```

Prevents division by zero

Task Description #5: AI-Assisted Code Completion Review

Prompt

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

AI-Generated Program

```
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance
    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")
    def show_balance(self):
        print("Current balance:", self.balance)
account = BankAccount()
while True:
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Balance")
    print("4. Exit")
    choice = int(input("Enter choice: "))
    if choice == 1:
        amt = int(input("Enter amount: "))
        account.deposit(amt)
    elif choice == 2:
        amt = int(input("Enter amount: "))
        account.withdraw(amt)
    elif choice == 3:
        account.show_balance()
    elif choice == 4:
        break
    else:
        print("Invalid choice")
```

Strengths

Uses class, loops, and conditionals
Simple and readable
Interactive menu system

Limitations

- No input validation for negative amounts
- No user authentication

Reflection (AI Ethics & Productivity)

- AI speeds up development and reduces boilerplate code.
- Human review is necessary to catch logical flaws.
- Responsible use includes validating AI output before deployment.