P.Naga Shiva Chaitanya
2303A51945 B - 27

Experiment 6: AI-Based Code Completion

Task Description #1: AI-Based Code Completion for Loops

Prompt:

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

AI-Generated Code:

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

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

Input:
N = 10

Output:
2
4
6
8
10

Optimization:

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

Loop Type Used

- **for loop**

Explanation

- `range(1, N+1)` loops from 1 to N
- `% 2 == 0` checks if the number is even
- Even numbers are printed

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 = 0
odd = 0

for num in numbers:
    if num % 2 == 0:
        even += 1
    else:
        odd += 1

print("Even:", even)
print("Odd:", odd)
```

Output:

Even: 3
Odd: 3

Explanation

- Loop goes through each number
- `if` checks even numbers
- `else` counts odd numbers

Task Description #3: Class Attributes Validation

Prompt:

Generate a Python class User that validates age and email.

AI-Generated Code:

```
class User:
    def __init__(self, age, email):
        self.age = age
        self.email = email

    def validate(self):
        if self.age < 18:
            print("Invalid age")
        elif "@" not in self.email:
            print("Invalid email")
        else:
            print("Valid user")
```

INPUT:

u1 = User(20, "test@gmail.com")
u1.validate()

u2 = User(15, "testgmail.com")
u2.validate()

OUTPUT:

Valid user
Invalid age

Issues Identified

- Email validation is too simple
- No return values

Improved Version:

```
class User:
    def __init__(self, age, email):
        self.age = age
        self.email = email

    def validate(self):
        return self.age >= 18 and "@" in self.email
```

Task Description #4: AI-Based Code Completion for Classes

Prompt

> Generate a Python class Student with attributes 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)
```

INPUT:

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

OUTPUT:

```
Total: 245
Average: 81.66
```

Minor Improvement

```
def average_marks(self):
    return round(self.total_marks() / len(self.marks), 2)
```

Task Description #5: AI-Assisted Code Completion Review

Prompt

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

AI-Generated Program

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

    def deposit(self, amount):
        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("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()
    else:
        Break
```

Concepts Used

- **Class** → BankAccount
- **Loop** → while
- **Conditionals** → if / elif

Ethical Use of AI

- AI used as **assistant**, not replacement
- Code reviewed and optimized manually
- Logic verified with test cases