

## AI-Based Code Completion

### **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.**

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

- **Identification of loop type used (for or while).**

- **Loop type:** for loop

- **Why:**

- The for loop iterates through a known range (1 to N)
- It is simple, readable, and ideal when the number of iterations is predetermined

- **Validation with sample inputs.**

**Enter a value for N: 10**

#### **OUTPUT :**

2 4

6 8

10

## 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.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
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)
```

- Correct count validation.

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Explanation of logic flow.

- A <sup>?</sup>list of numbers is defined.

- Two counters (even\_count and odd\_count) are initialized to zero.
- A for loop iterates through each number in the list.
- Inside the loop:

- The if condition checks if the number is divisible by 2.
- If true → the number is even, so even\_count is increased.
- Otherwise → the number is odd, so odd\_count is increased.

💡 After the loop finishes, the final counts are printed.

### 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.

```
class User:  
    def  
    __init__(self, age, email):  
        self.age = age  
        self.email = email  
    def validate_age(self):  
        if  
            isinstance(self.age, int) and self.age >= 18:  
                return True  
        else:  
            return False  
    def  
    validate_email(self):  
        if isinstance(self.email, str) and "@" in self.email and "." in  
        self.email:  
            return True  
        else:  
            return False
```

- Verification of condition handling.

Age Validation

- Checks if:
  - Age is an integer ◦ Age is 18 or above
- Returns:
  - True → valid age ◦ False → invalid age

## Email Validation •

Checks if:

- Email is a string ◦
- Contains "@" and "."
- Returns:
  - True → valid email
  - format ◦ False → invalid email format
- **Test cases for valid and invalid inputs.** # Valid user

```
user1 = User(25, "user@example.com") print("User1  
Age Valid:", user1.validate_age()) print("User1 Email  
Valid:", user1.validate_email())
```

```
# Invalid age
```

```
user2 = User(16, "teen@example.com") print("User2  
Age Valid:", user2.validate_age()) # Invalid email  
user3 = User(30, "invalidemail") print("User3 Email  
Valid:", user3.validate_email())
```

```
# Invalid age and email
```

```
user4 = User("twenty", "userexample.com") print("User4  
Age Valid:", user4.validate_age()) print("User4 Email Valid:",  
user4.validate_email())
```

**Expected Output**

User1 Age Valid: True

User1 Email Valid: True

User2 Age Valid: False

User3 Email Valid: False

User4 Age Valid: False

User4 Email Valid: False

**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.**

```
class Student:  
    def __init__(self, name,  
     roll_number, marks):  
        self.name = name  
        self.roll_number = roll_number  
        self.marks = marks # list of marks  
    def calculate_total(self):  
        return sum(self.marks)  
    def calculate_average(self):  
        if len(self.marks) == 0:  
            return 0  
        return self.calculate_total()  
    / len(self.marks)
```

- **Verification of correctness and completeness of class structure.**

Class Structure Check

- Attributes
  - name → stores student name
  - roll\_number → stores roll number
  - marks → list of numerical marks
- Methods

- calculate\_total() → correctly returns sum of marks
- calculate\_average() → correctly returns average marks
- Handles division safely when marks list is empty
- **Minor manual improvements (if needed) with justification.**

### **Sample Test Case**

```
student1 = Student("Alice", 101, [85, 90, 78, 92])
```

```
print("Total Marks:", student1.calculate_total()) print("Average Marks:",
student1.calculate_average())
```

### **Expected Output**

Total Marks: 345

Average Marks: 86.25

**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.”

```
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}")
        else:
            print("Deposit amount must be positive.")
    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
        elif amount > self.balance:
            print("Insufficient balance.")
        else:
```

```
        self.balance -= amount
print(f"Withdrawn: {amount}")
def check_balance(self):
    print(f"Current Balance: {self.balance}")
choice = input("Enter your choice:
")
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("Thank you for using the bank system.")
break else:                  print("Invalid choice.
Please try again.")
```