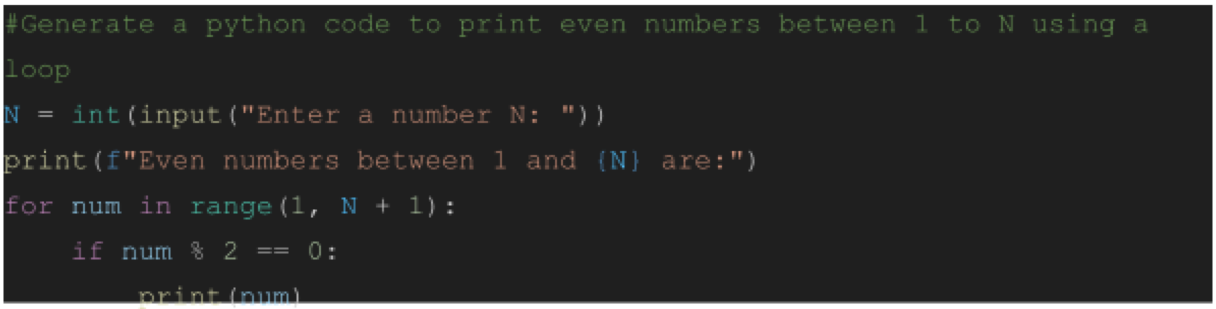
AI Assisted Coding Lab 6.1

M Vikas

2303A51155
Batch-17

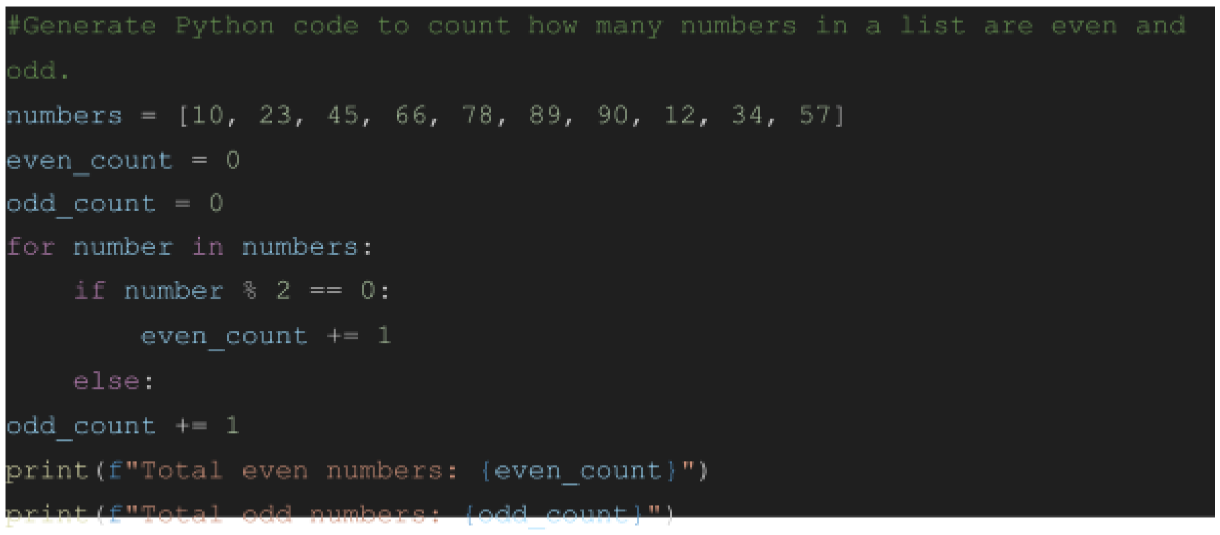
Task Description #1 (AI-Based Code Completion for Loops) Task: Use an AI code completion tool to generate a loop-based program.
Prompt:
“Generate Python code to print all even numbers between 1 and N using a loop.”
Expected Output:
• AI-generated loop logic.
• Identification of loop type used (for or while).
• Validation with sample inputs.



```
#Generate a python code to print even numbers between 1 to N using a
loop
N = int(input("Enter a number N: "))
print(f"Even numbers between 1 and {N} are:")
for num in range(1, N + 1):
    if num % 2 == 0:
        print(num)
```

Input:
Enter a number N: 10
Output:
Even numbers between 1 and 10 are:
2
4
6
8
10

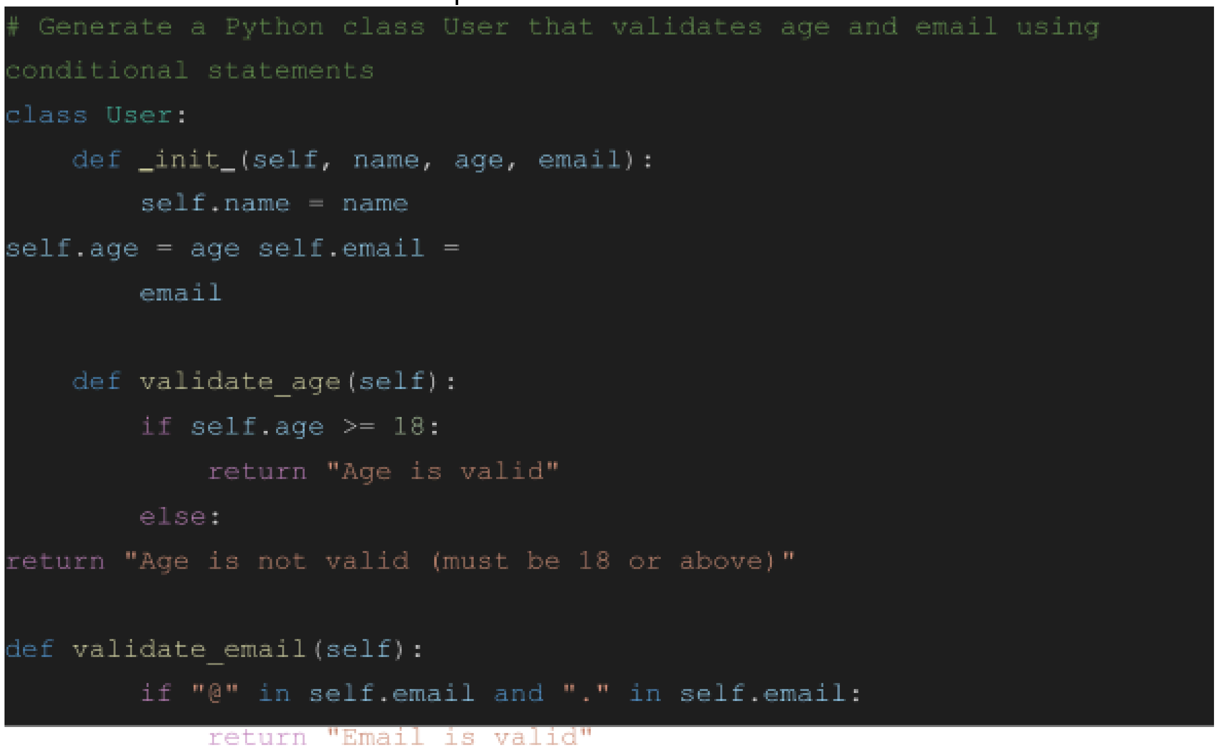
Task Description #2 (AI-Based Code Completion for Loop with Conditionals)
Task: Use an AI code completion tool to combine loops and conditionals.
Prompt:
“Generate Python code to count how many numbers in a list are even and odd.”
Expected Output:
• AI-generated code using loop and if condition.
• Correct count validation.
• Explanation of logic flow.



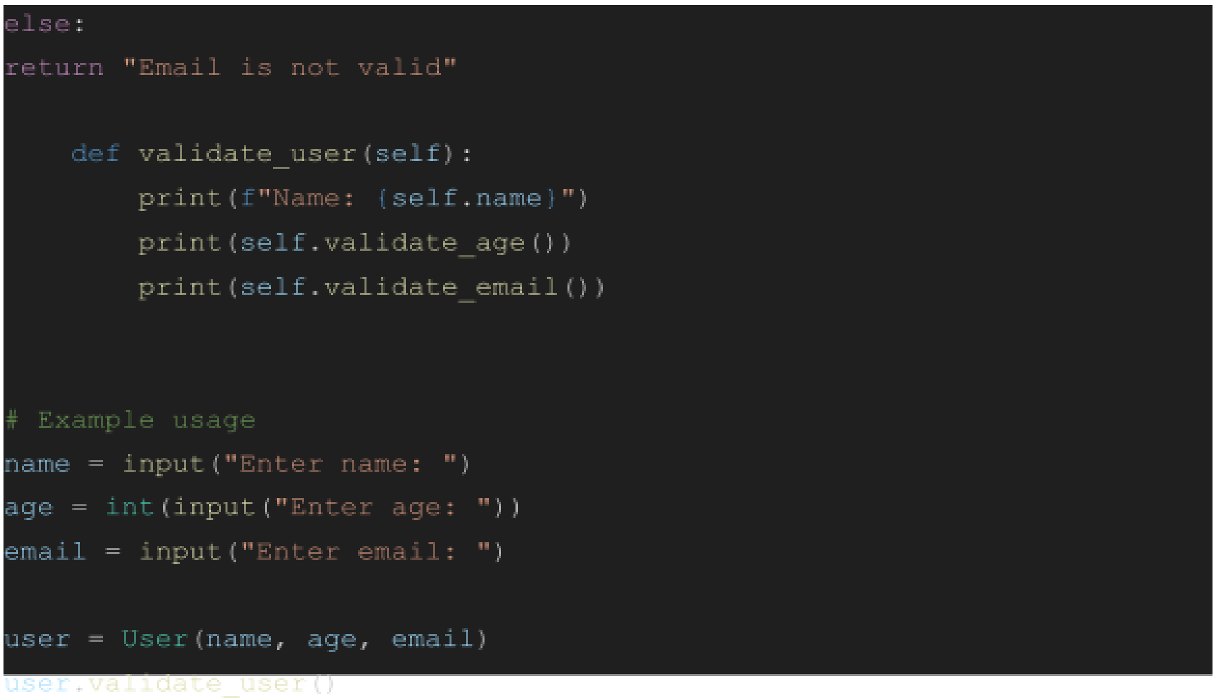
```
#Generate Python code to count how many numbers in a list are even and
odd.
numbers = [10, 23, 45, 66, 78, 89, 90, 12, 34, 57]
even_count = 0
odd_count = 0
for number in numbers:
    if number % 2 == 0:
        even_count += 1
    else:
odd_count += 1
print(f"Total even numbers: {even_count}")
print(f"Total odd numbers: {odd_count}")
```

Output:
Total even numbers: 6
Total odd numbers: 4

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.



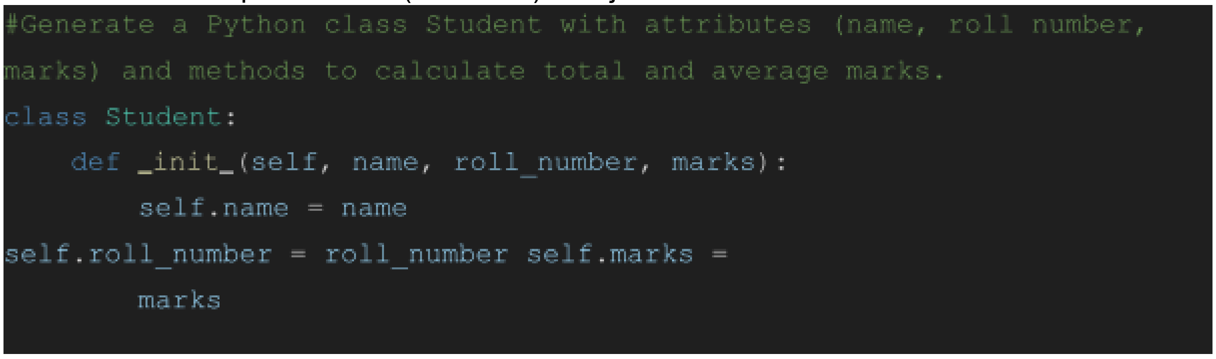
```
# Generate a Python class User that validates age and email using
conditional statements
class User:
    def _init_(self, name, age, email):
        self.name = name
self.age = age self.email =
        email

    def validate_age(self):
        if self.age >= 18:
            return "Age is valid"
        else:
return "Age is not valid (must be 18 or above)"

def validate_email(self):
        if "@" in self.email and "." in self.email:
            return "Email is valid"
```



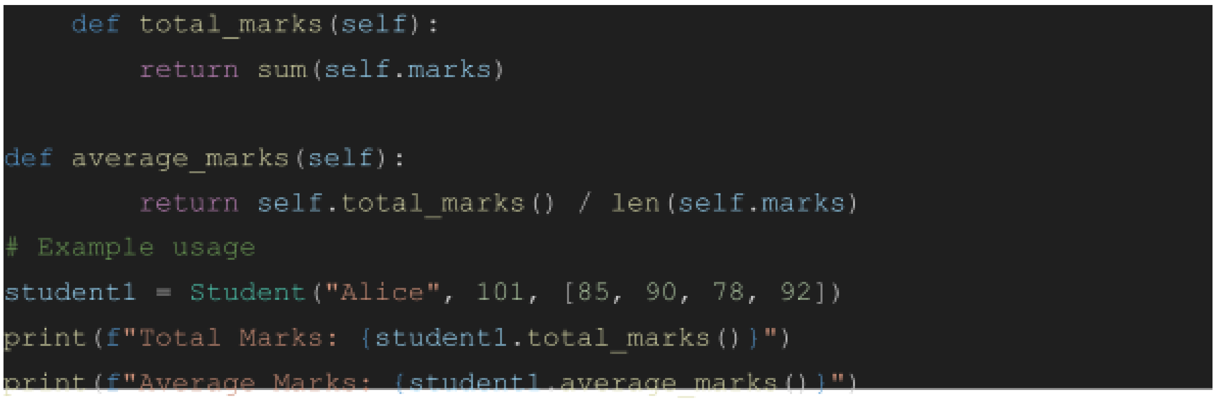
```
else:
return "Email is not valid"

    def validate_user(self):
        print(f"Name: {self.name}")
        print(self.validate_age())
        print(self.validate_email())


# Example usage
name = input("Enter name: ")
age = int(input("Enter age: "))
email = input("Enter email: ")

user = User(name, age, email)
user.validate_user()
```

Output:
Enter name: ram
Enter age: 21
Enter email: temp@gmail.com Name:
ram
Age is valid
Email is valid

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.



```
#Generate a Python class Student with attributes (name, roll number,
marks) and 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
```

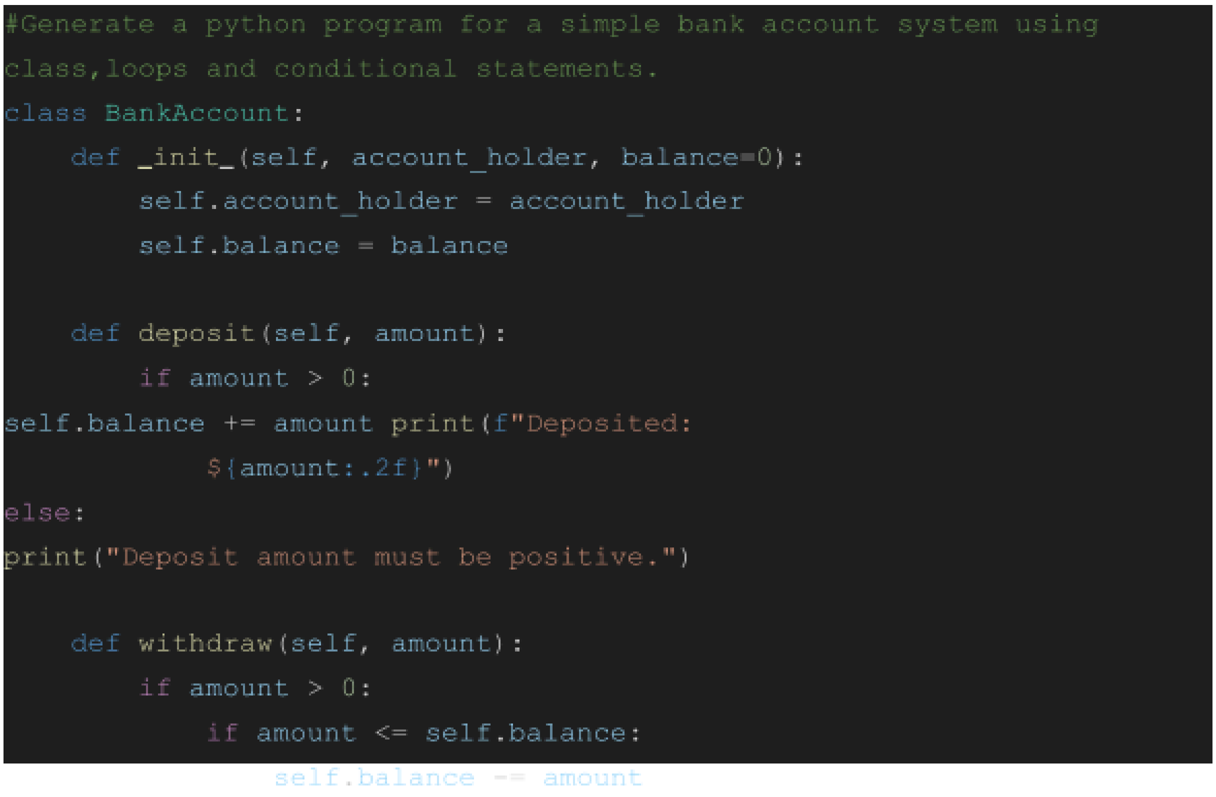


```
    def total_marks(self):
        return sum(self.marks)

def average_marks(self):
        return self.total_marks() / len(self.marks)
# Example usage
student1 = Student("Alice", 101, [85, 90, 78, 92])
print(f"Total Marks: {student1.total_marks()}")
print(f"Average Marks: {student1.average_marks()}")
```

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.”
Expected Output:
• Complete AI-generated program.
• Identification of strengths and limitations of AI suggestions.
• Reflection on how AI assisted coding productivity.



```
#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:.2f}")
else:
print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
```

```python
                print(f"Withdrew: ${amount:.2f}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def get_balance(self):
        return self.balance
def main():
    print("Welcome to the Simple 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':
            balance = account.get_balance()
            print(f"Current balance: ${balance:.2f}")
        elif choice == '4':
            print("Thank you for using the Simple Bank Account System.
Goodbye!")
            break
        else:
            print("Invalid choice. Please try
again.") if _name_ == "_main_":
    main()
```

Input & Output:

```
Welcome to the Simple Bank Account System
Enter account holder name: ram
```

```
Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 1
Enter amount to deposit: 2000
Deposited: $2000.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 3
Current balance: $2000.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 2
Enter amount to withdraw: 1200
Withdrew: $1200.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 3
Current balance: $800.00

Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option (1-4): 4
Thank you for using the Simple Bank Account System. Goodbye!
```