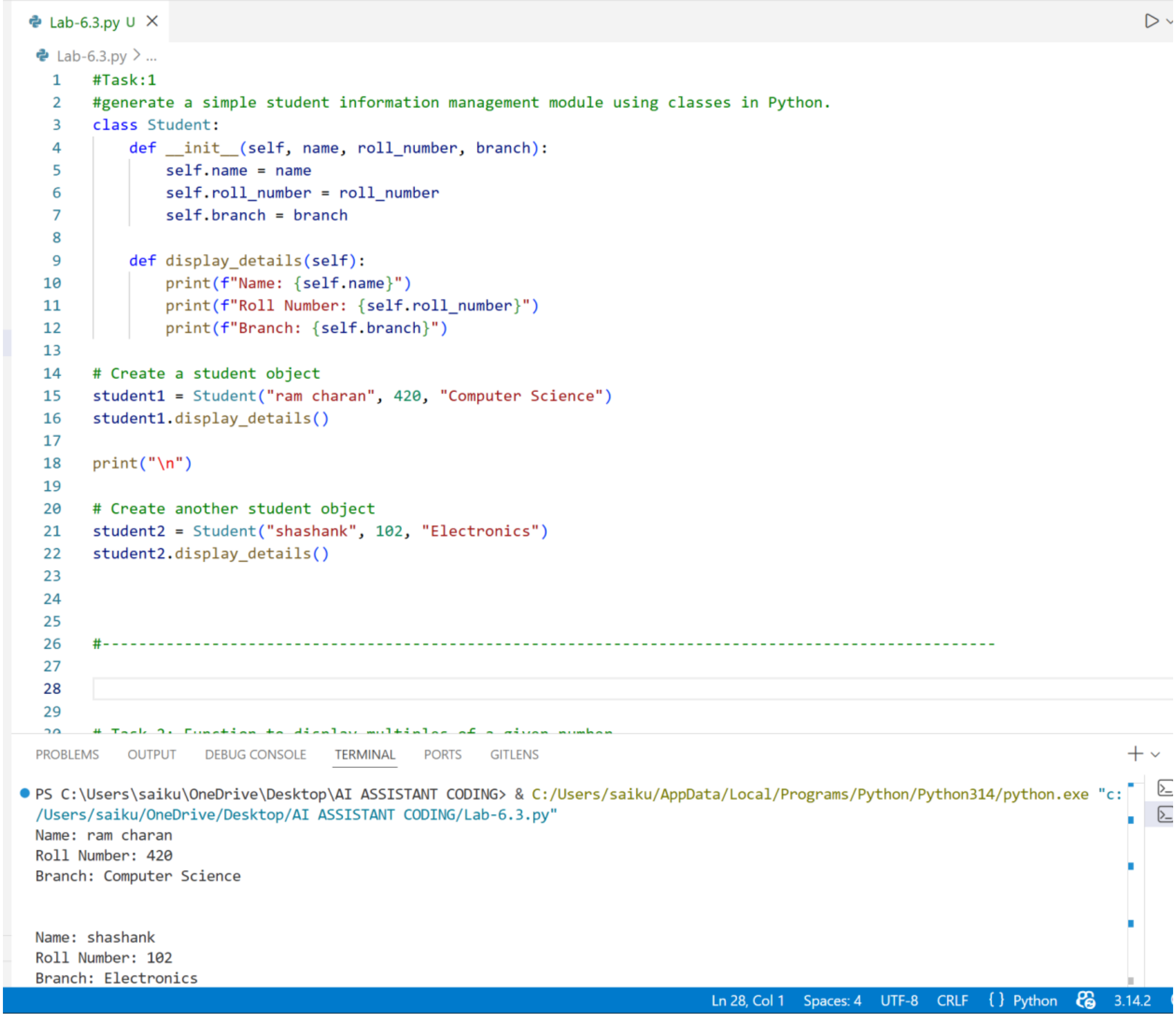
AI ASSISTED CODING

J.Sai Kumar || Batch:-09 || 2303A51562

Task Description #1: Classes (Student Class)

Scenario

You are developing a simple student information management module.



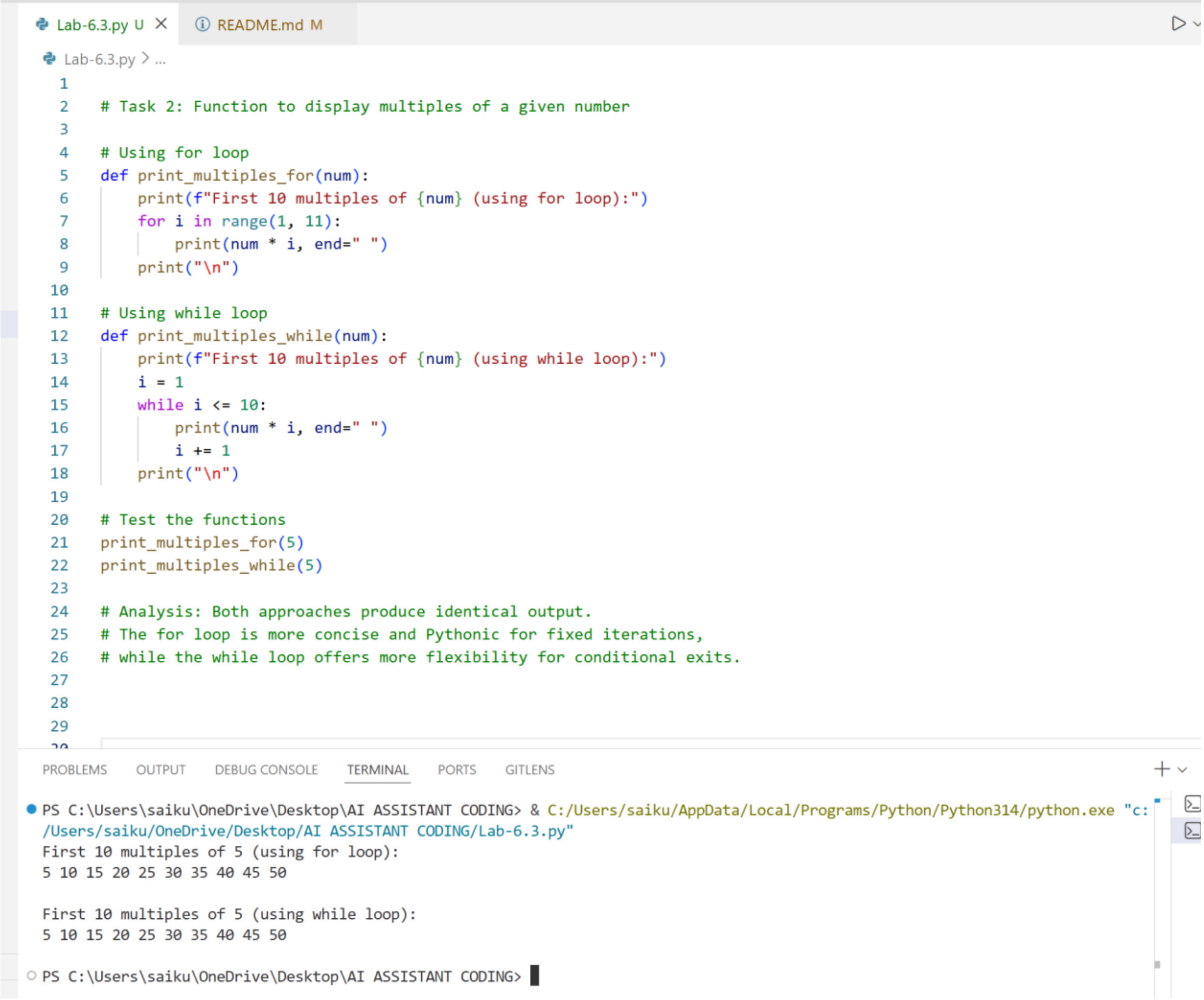
```python
#Task:1
#generate a simple student information management module using classes in Python.
class Student:
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Branch: {self.branch}")

# Create a student object
student1 = Student("ram charan", 420, "Computer Science")
student1.display_details()

print("\n")

# Create another student object
student2 = Student("shashank", 102, "Electronics")
student2.display_details()



#----------------------------------------------------------------------------------------------------


# Task 2: Function to display multiples of a given number
```

```
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python/Python314/python.exe "c:
/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-6.3.py"
Name: ram charan
Roll Number: 420
Branch: Computer Science


Name: shashank
Roll Number: 102
Branch: Electronics
```

Task Description #2: Loops (Multiples of a Number)

Scenario

You are writing a utility function to display multiples of a given number.



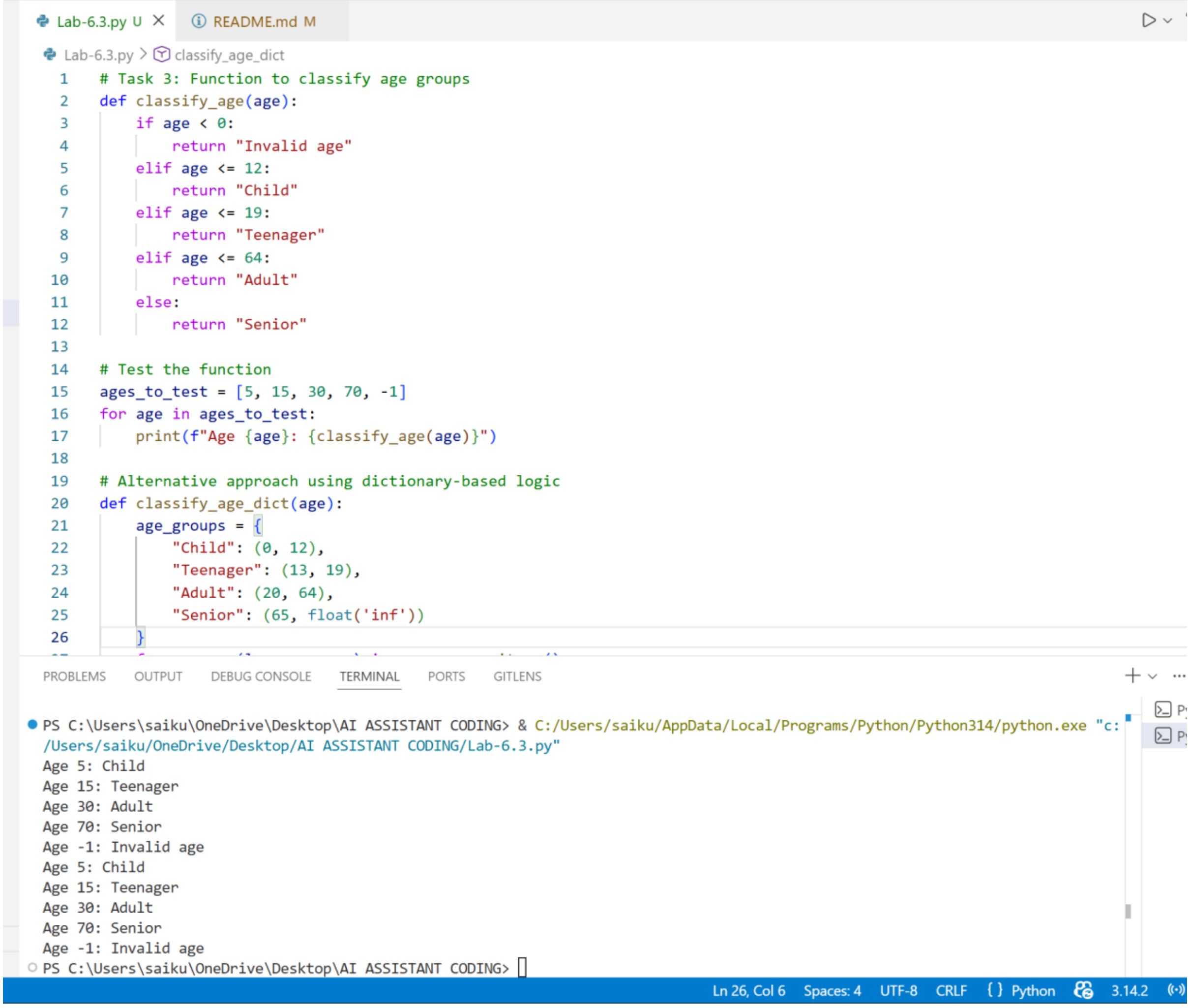
```python

# Task 2: Function to display multiples of a given number

# Using for loop
def print_multiples_for(num):
    print(f"First 10 multiples of {num} (using for loop):")
    for i in range(1, 11):
        print(num * i, end=" ")
    print("\n")

# Using while loop
def print_multiples_while(num):
    print(f"First 10 multiples of {num} (using while loop):")
    i = 1
    while i <= 10:
        print(num * i, end=" ")
        i += 1
    print("\n")

# Test the functions
print_multiples_for(5)
print_multiples_while(5)

# Analysis: Both approaches produce identical output.
# The for loop is more concise and Pythonic for fixed iterations,
# while the while loop offers more flexibility for conditional exits.
```

```
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python/Python314/python.exe "c:
/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-6.3.py"
First 10 multiples of 5 (using for loop):
5 10 15 20 25 30 35 40 45 50

First 10 multiples of 5 (using while loop):
5 10 15 20 25 30 35 40 45 50

PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING>
```

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.



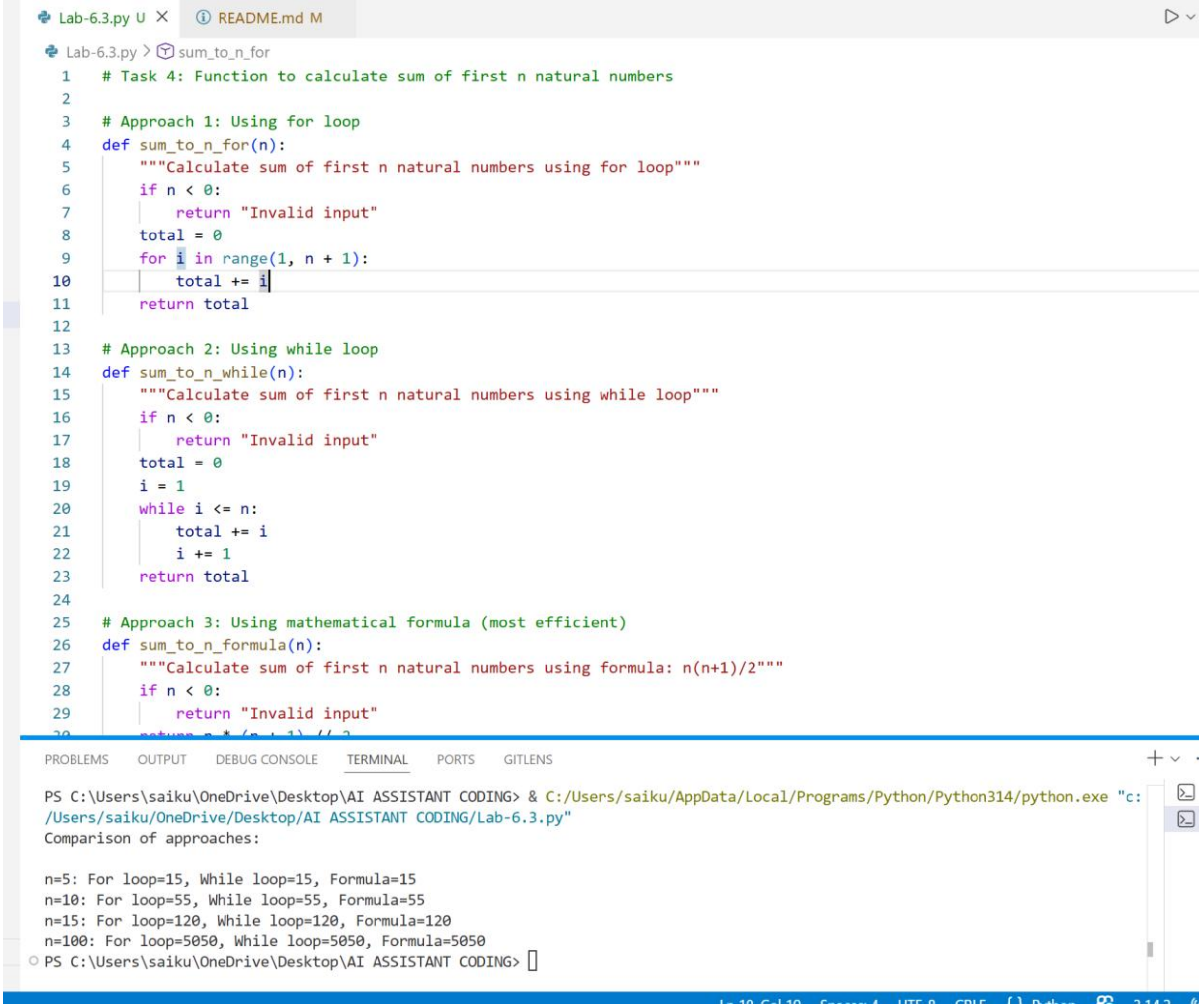
```python
# Task 3: Function to classify age groups
def classify_age(age):
    if age < 0:
        return "Invalid age"
    elif age <= 12:
        return "Child"
    elif age <= 19:
        return "Teenager"
    elif age <= 64:
        return "Adult"
    else:
        return "Senior"

# Test the function
ages_to_test = [5, 15, 30, 70, -1]
for age in ages_to_test:
    print(f"Age {age}: {classify_age(age)}")

# Alternative approach using dictionary-based logic
def classify_age_dict(age):
    age_groups = {
        "Child": (0, 12),
        "Teenager": (13, 19),
        "Adult": (20, 64),
        "Senior": (65, float('inf'))
    }
```

```
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python/Python314/python.exe "c:
/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-6.3.py"
Age 5: Child
Age 15: Teenager
Age 30: Adult
Age 70: Senior
Age -1: Invalid age
Age 5: Child
Age 15: Teenager
Age 30: Adult
Age 70: Senior
Age -1: Invalid age
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING>
```

Task Description #4: For and While Loops (Sum of First n Numbers)

Scenario

You need to calculate the sum of the first n natural numbers.



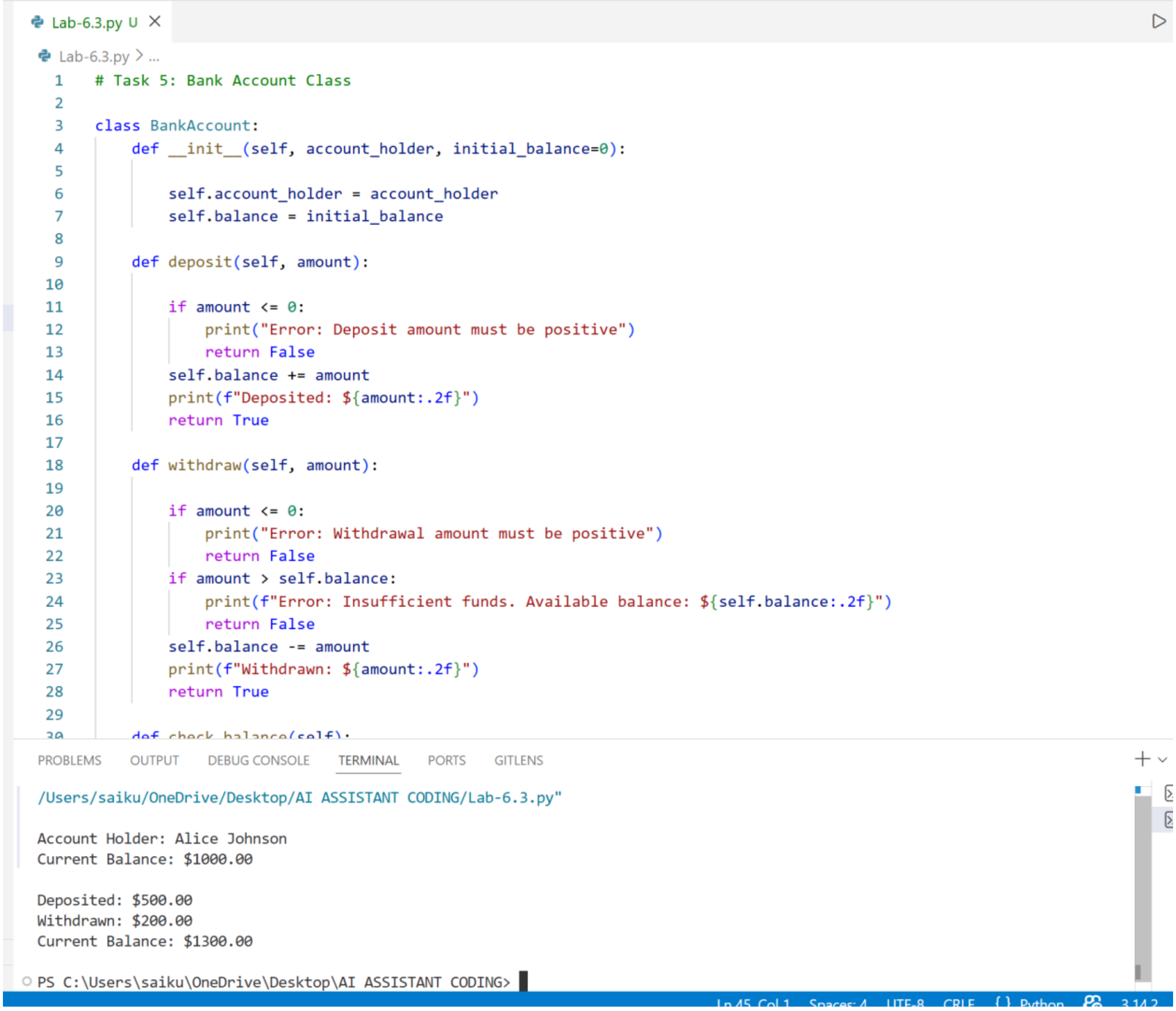
```python
# Task 4: Function to calculate sum of first n natural numbers

# Approach 1: Using for loop
def sum_to_n_for(n):
    """Calculate sum of first n natural numbers using for loop"""
    if n < 0:
        return "Invalid input"
    total = 0
    for i in range(1, n + 1):
        total += i
    return total

# Approach 2: Using while loop
def sum_to_n_while(n):
    """Calculate sum of first n natural numbers using while loop"""
    if n < 0:
        return "Invalid input"
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total

# Approach 3: Using mathematical formula (most efficient)
def sum_to_n_formula(n):
    """Calculate sum of first n natural numbers using formula: n(n+1)/2"""
    if n < 0:
        return "Invalid input"
    return n * (n + 1) // 2
```

```
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python/Python314/python.exe "c:
/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-6.3.py"
Comparison of approaches:

n=5: For loop=15, While loop=15, Formula=15
n=10: For loop=55, While loop=55, Formula=55
n=15: For loop=120, While loop=120, Formula=120
n=100: For loop=5050, While loop=5050, Formula=5050
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> 
```

Task Description #5: Classes (Bank Account Class)

Scenario

You are designing a basic banking application.



```python
# Task 5: Bank Account Class

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:
            print("Error: Deposit amount must be positive")
            return False
        self.balance += amount
        print(f"Deposited: ${amount:.2f}")
        return True

    def withdraw(self, amount):

        if amount <= 0:
            print("Error: Withdrawal amount must be positive")
            return False
        if amount > self.balance:
            print(f"Error: Insufficient funds. Available balance: ${self.balance:.2f}")
            return False
        self.balance -= amount
        print(f"Withdrawn: ${amount:.2f}")
        return True

    def check_balance(self):
```

```
/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-6.3.py"

Account Holder: Alice Johnson
Current Balance: $1000.00

Deposited: $500.00
Withdrawn: $200.00
Current Balance: $1300.00

PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING>
```