NAME:CH.Kruthankiran H.NO:2303A51404 BATCH:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju	
Course Code	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week3 – Wednesday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
AssignmentNumber:6.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals** **Lab Objectives** • To explore AI-powered auto-completion features for core Python constructs such as classes,	Week3 - Wednesday

loops, and conditional statements.
• To analyze how AI tools suggest logic for object-oriented programming and control structures.
• To evaluate the correctness, readability, and completeness of AI-generated Python code.

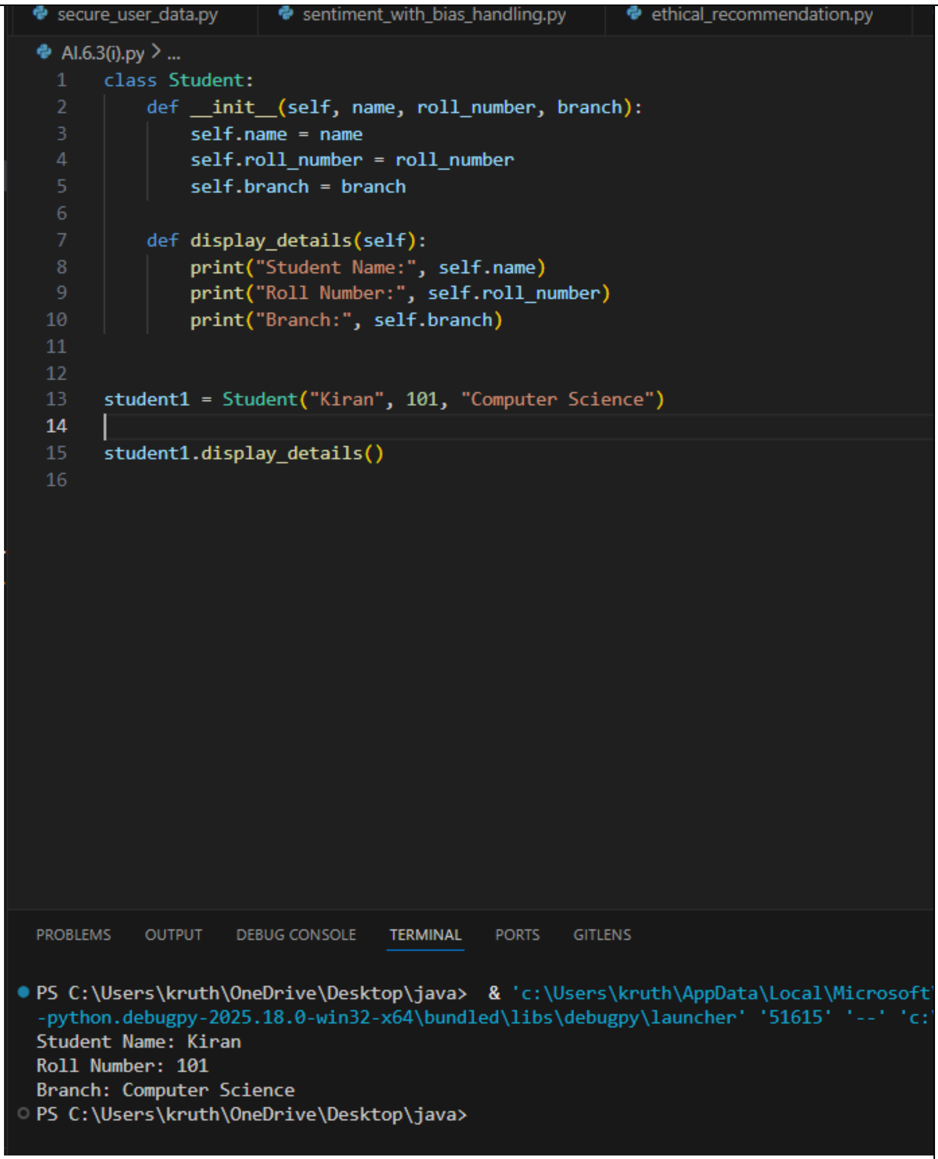
Lab Outcomes (LOs)
After completing this lab, students will be able to:
• Use AI tools to generate and complete Python class definitions and methods.
• Understand and assess AI-suggested loop constructs for iterative tasks.
• Generate and evaluate conditional statements using AI-driven prompts.
• Critically analyze AI-assisted code for correctness, clarity, and efficiency.

Task Description #1: Classes (Student Class)

Scenario
You are developing a simple student information management module.

Task
• Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
• The class should include attributes such as name, roll number, and branch.
• Add a method display_details() to print student information.
• Execute the code and verify the output.
• Analyze the code generated by the AI tool for correctness and clarity.

Expected Output #1
• A Python class with a constructor (__init__) and a display_details() method.
• Sample object creation and output displayed on the console.
• Brief analysis of AI-generated code.



```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("Student Name:", self.name)
        print("Roll Number:", self.roll_number)
        print("Branch:", self.branch)


student1 = Student("Kiran", 101, "Computer Science")

student1.display_details()
```

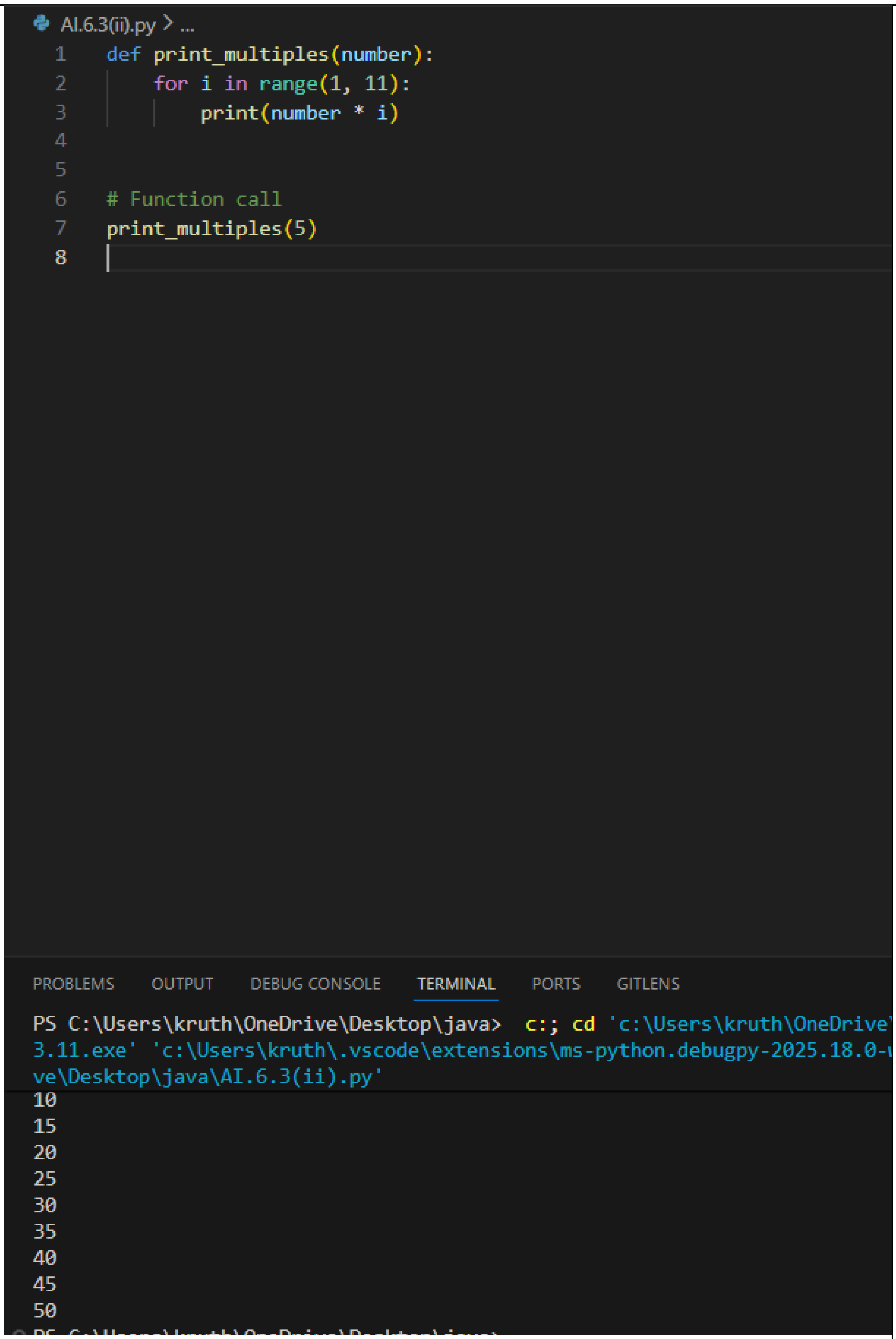
```
PS C:\Users\kruth\OneDrive\Desktop\java>  & 'c:\Users\kruth\AppData\Local\Microsoft
-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51615' '--' 'c:
Student Name: Kiran
Roll Number: 101
Branch: Computer Science
PS C:\Users\kruth\OneDrive\Desktop\java>
```

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

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

Task
• Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.
• Analyze the generated loop logic.
• Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).

Expected Output #2
• Correct loop-based Python implementation.
• Output showing the first 10 multiples of a number.
• Comparison and analysis of different looping approaches.


AI.6.3(ii).py > ...

```python
def print_multiples(number):
    for i in range(1, 11):
        print(number * i)


# Function call
print_multiples(5)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDrive'
3.11.exe' 'c:\Users\kruth\.vscode\extensions\ms-python.debugpy-2025.18.0-
ve\Desktop\java\AI.6.3(ii).py'
10
15
20
25
30
35
40
45
50
```

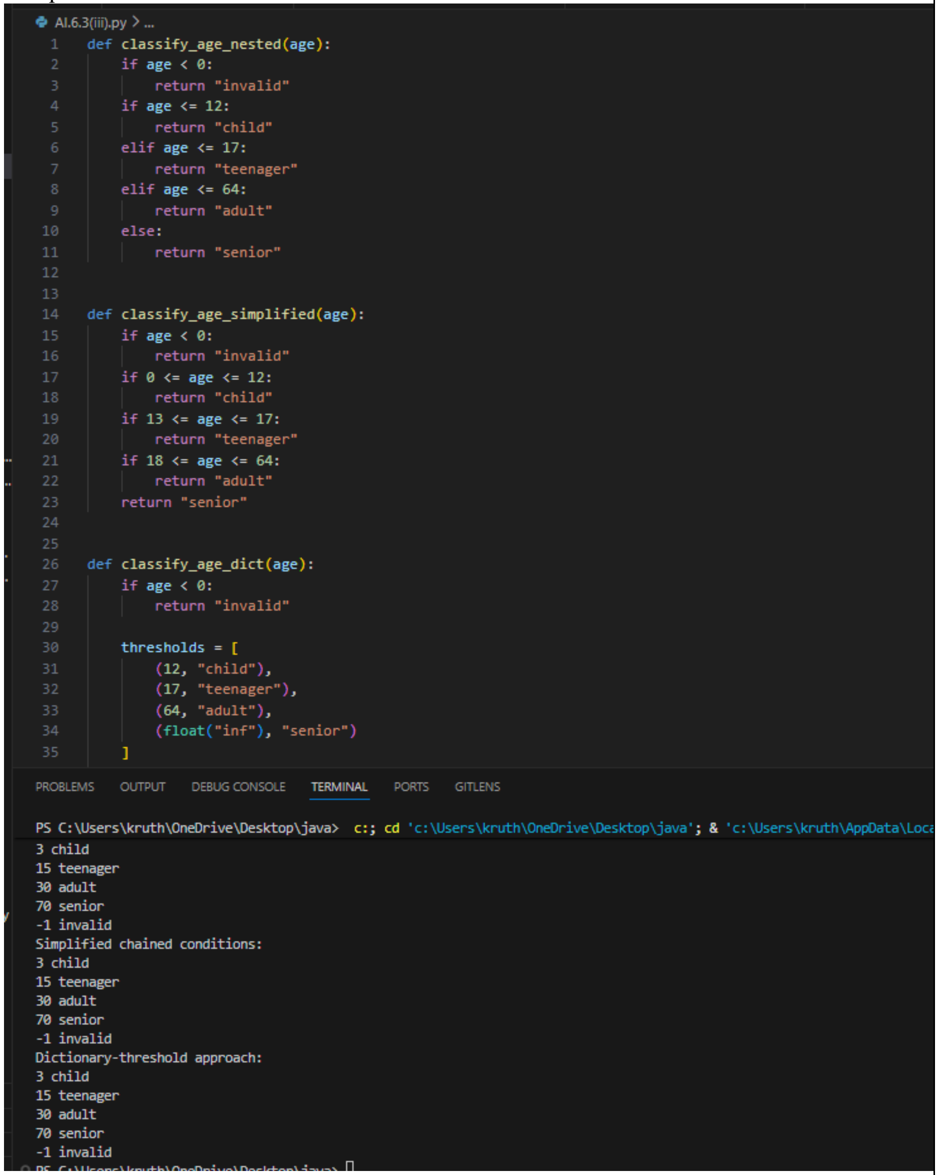
Task Description #3: Conditional Statements (Age Classification)

Scenario
You are building a basic classification system based on age.

Task
• Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).
• Analyze the generated conditions and logic.

• Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

Expected Output #3
• A Python function that classifies age into appropriate groups.
• Clear and correct conditional logic.
• Explanation of how the conditions work.


AI.6.3(iii).py > ...

```python
def classify_age_nested(age):
    if age < 0:
        return "invalid"
    if age <= 12:
        return "child"
    elif age <= 17:
        return "teenager"
    elif age <= 64:
        return "adult"
    else:
        return "senior"


def classify_age_simplified(age):
    if age < 0:
        return "invalid"
    if 0 <= age <= 12:
        return "child"
    if 13 <= age <= 17:
        return "teenager"
    if 18 <= age <= 64:
        return "adult"
    return "senior"


def classify_age_dict(age):
    if age < 0:
        return "invalid"

    thresholds = [
        (12, "child"),
        (17, "teenager"),
        (64, "adult"),
        (float("inf"), "senior")
    ]
```

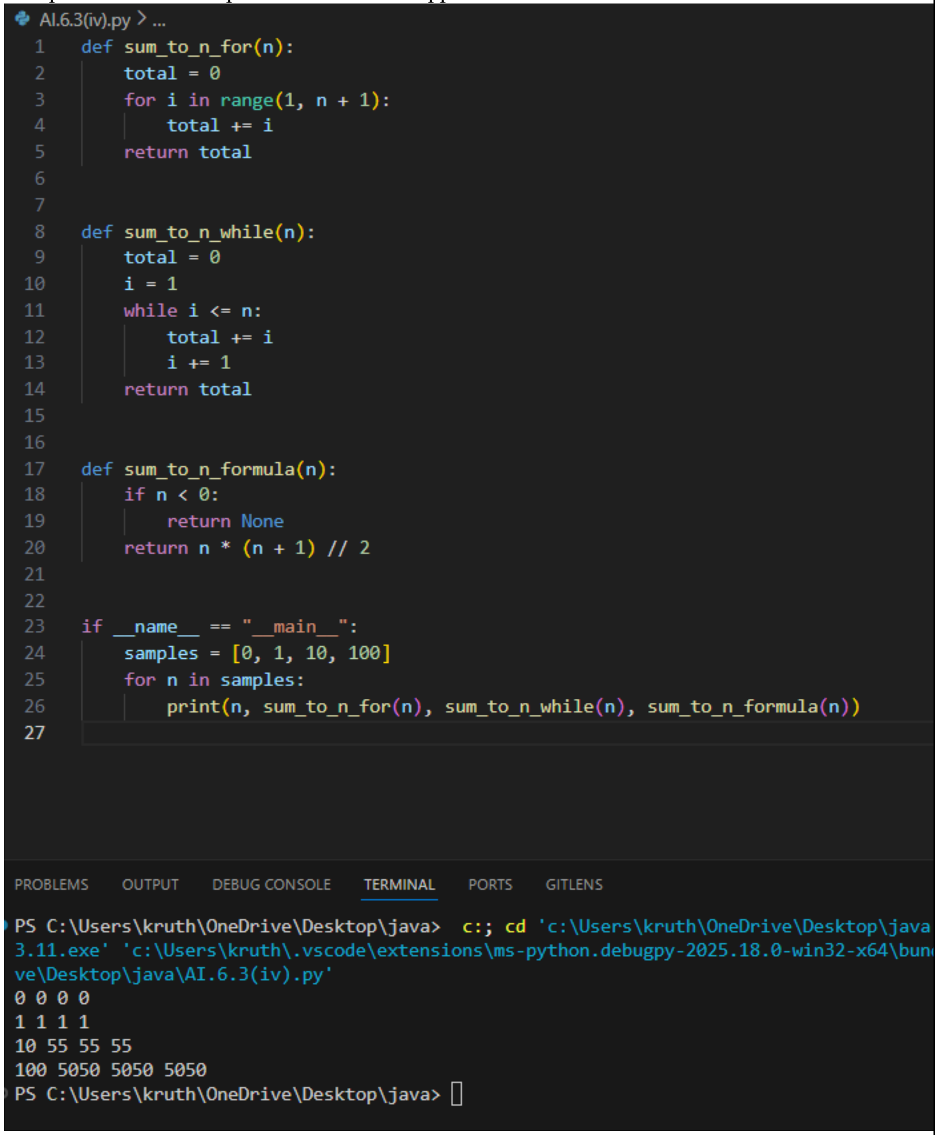
```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDrive\Desktop\java'; & 'c:\Users\kruth\AppData\Loca
3 child
15 teenager
30 adult
70 senior
-1 invalid
Simplified chained conditions:
3 child
15 teenager
30 adult
70 senior
-1 invalid
Dictionary-threshold approach:
3 child
15 teenager
30 adult
70 senior
-1 invalid
PS C:\Users\kruth\OneDrive\Desktop\java> 
```

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.

Task
• Use AI assistance to generate a sum_to_n() function using a for loop.
• Analyze the generated code.
• Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

Expected Output #4
• Python function to compute the sum of first n numbers.
• Correct output for sample inputs.
• Explanation and comparison of different approaches.


AI.6.3(iv).py > ...

```python
def sum_to_n_for(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total


def sum_to_n_while(n):
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total


def sum_to_n_formula(n):
    if n < 0:
        return None
    return n * (n + 1) // 2


if __name__ == "__main__":
    samples = [0, 1, 10, 100]
    for n in samples:
        print(n, sum_to_n_for(n), sum_to_n_while(n), sum_to_n_formula(n))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

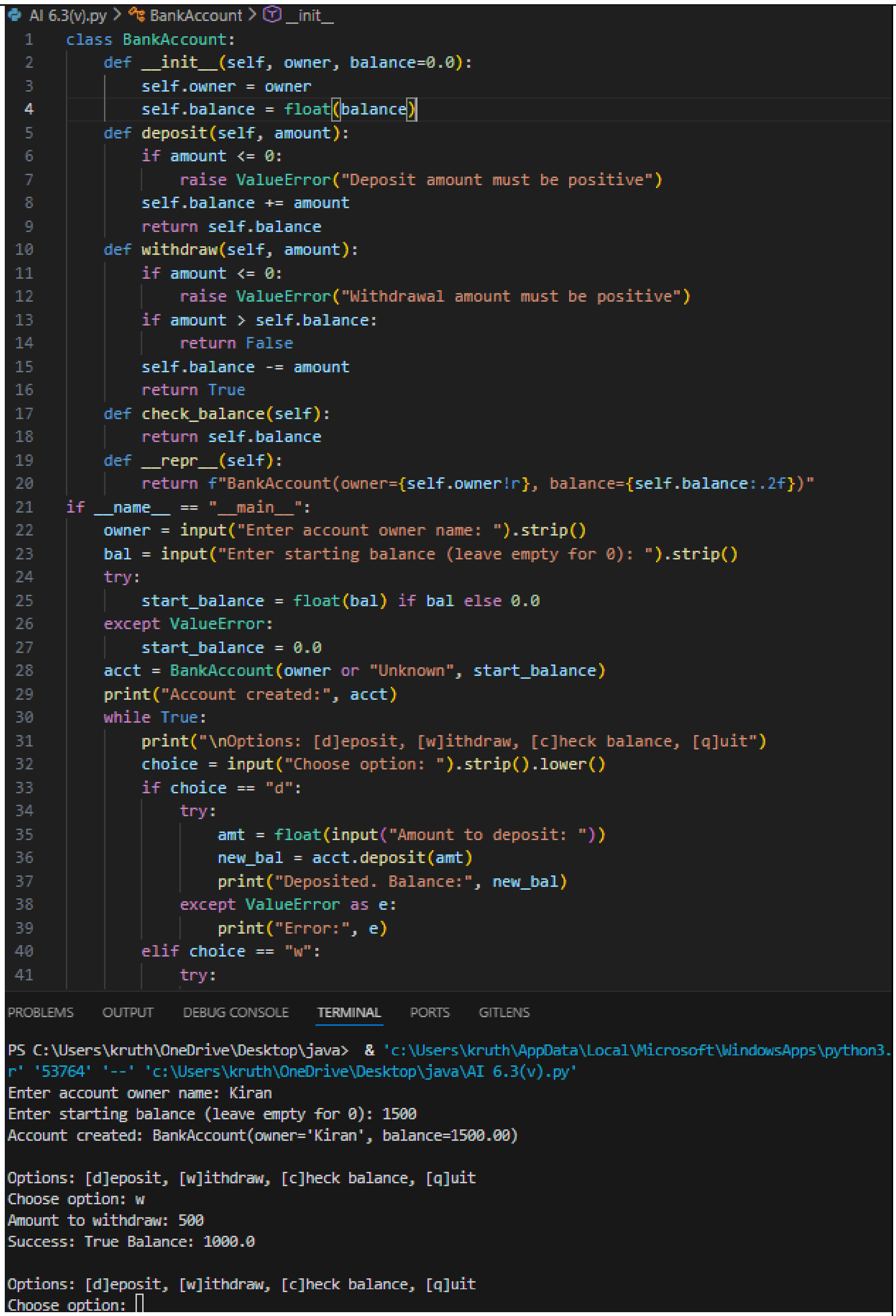
```
PS C:\Users\kruth\OneDrive\Desktop\java>  c:; cd 'c:\Users\kruth\OneDrive\Desktop\java
3.11.exe' 'c:\Users\kruth\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bun
ve\Desktop\java\AI.6.3(iv).py'
0 0 0 0
1 1 1 1
10 55 55 55
100 5050 5050 5050
PS C:\Users\kruth\OneDrive\Desktop\java> 
```

Task Description #5: Classes (Bank Account Class)

Scenario
You are designing a basic banking application.

Task
• Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().
• Analyze the AI-generated class structure and logic.
• Add meaningful comments and explain the working of the code.

Expected Output #5
• Complete Python Bank Account class.
• Demonstration of deposit and withdrawal operations with updated balance.



```python
class BankAccount:
    def __init__(self, owner, balance=0.0):
        self.owner = owner
        self.balance = float(balance)
    def deposit(self, amount):
        if amount <= 0:
            raise ValueError("Deposit amount must be positive")
        self.balance += amount
        return self.balance
    def withdraw(self, amount):
        if amount <= 0:
            raise ValueError("Withdrawal amount must be positive")
        if amount > self.balance:
            return False
        self.balance -= amount
        return True
    def check_balance(self):
        return self.balance
    def __repr__(self):
        return f"BankAccount(owner={self.owner!r}, balance={self.balance:.2f})"
if __name__ == "__main__":
    owner = input("Enter account owner name: ").strip()
    bal = input("Enter starting balance (leave empty for 0): ").strip()
    try:
        start_balance = float(bal) if bal else 0.0
    except ValueError:
        start_balance = 0.0
    acct = BankAccount(owner or "Unknown", start_balance)
    print("Account created:", acct)
    while True:
        print("\nOptions: [d]eposit, [w]ithdraw, [c]heck balance, [q]uit")
        choice = input("Choose option: ").strip().lower()
        if choice == "d":
            try:
                amt = float(input("Amount to deposit: "))
                new_bal = acct.deposit(amt)
                print("Deposited. Balance:", new_bal)
            except ValueError as e:
                print("Error:", e)
        elif choice == "w":
            try:
```

```
PS C:\Users\kruth\OneDrive\Desktop\java>  & 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.
r' '53764' '--' 'c:\Users\kruth\OneDrive\Desktop\java\AI 6.3(v).py'
Enter account owner name: Kiran
Enter starting balance (leave empty for 0): 1500
Account created: BankAccount(owner='Kiran', balance=1500.00)

Options: [d]eposit, [w]ithdraw, [c]heck balance, [q]uit
Choose option: w
Amount to withdraw: 500
Success: True Balance: 1000.0

Options: [d]eposit, [w]ithdraw, [c]heck balance, [q]uit
Choose option: 
```

• Well-commented code with a clear explanation.

Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.