SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M.Tech. and MCA		**Assignment Type: Lab**	**AcademicYear:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		**Course Title**	AI Assisted Problem Solving Using Python
Year/Sem	I/I	**Regulation**	R24
Date and Day of Assignment	Week3 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	M.Tech. and MCA
AssignmentNumber:6.3(Present assignment number)/**24**(Total number of assignments)			

ASSIGNMENT – 6

Name: Rimsha Mujeeb **Roll Number:2503B05138 (M.Tech - CSE)**

Q.No.	Question
Q1.	**Task Description#1 (Classes)** • Use AI to complete a Student class with attributes and a method. • Check output • Analyze the code generated by AI tool **Expected Output#1** • Class with constructor and display_details() method

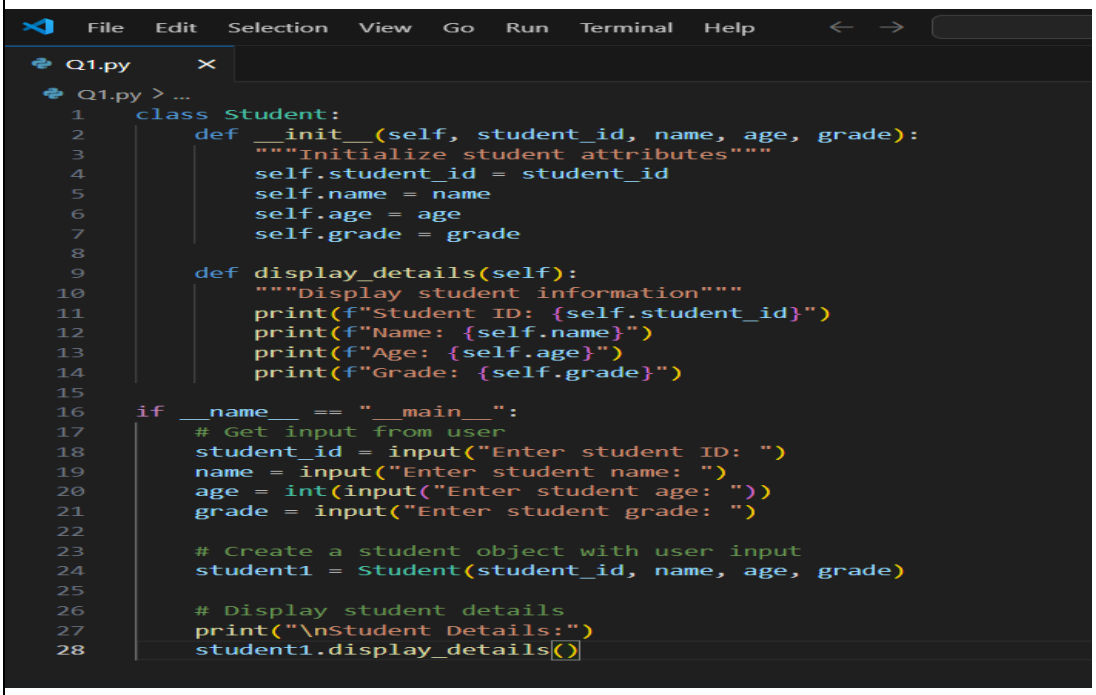
PROMPT:

Create a Python class named Student.
Include attributes for name, roll number, and grade.
Add a constructor to initialize these attributes.
Implement a method display_details() to print all the details.

CODE ANALYSIS:

1. Constructor __init__ initializes ID, name, age, and grade.
2. display_details() method prints details using f-strings.
3. Main block takes user input and creates a Student object.
4. Displays the entered student details clearly.

5. Demonstrates basic OOP concepts — class, object, and method.
6. Code is simple, readable, and effective.
7. Can be improved with input validation and multiple records support.



Q2.

Task Description#2 (Loops)

- Prompt AI to complete a function that prints the first 10 multiples of a number using a loop.
- Analyze the generated code
- Ask AI to generate code using other controlled looping

Expected Output#2

Correct loop-based implementation

PROMPT :

1- Write a Python function that takes a number as input and prints its first 10 multiples using a loop.

2- Generate an alternative implementation using a while loop. Include comments explaining the code.(modify the given program without changing previous one)

```python
# Q2.py
# Task: print the first 10 multiples of a number using loops
# Contains two controlled-loop implementations and a short analysis.

def print_multiples_for(n):
    """Print first 10 multiples of n using a for loop."""
    for i in range(1, 11):
        print(f"{n} x {i} = {n * i}")

def print_multiples_while(n):
    """Print first 10 multiples of n using a while loop (alternative)."""
    i = 1
    while i <= 10:
        print(f"{n} x {i} = {n * i}")
        i += 1

# Analysis (brief):
# - Both functions produce the same output: multiples 1..10 of the given n.
# - for loop version is concise and idiomatic for a known fixed range.
# - while loop version demonstrates manual counter control.
# - Time complexity: O(k) where k=10 (constant), effectively O(1) for this fixed task.

if __name__ == "__main__":
    try:
        num = int(input("Enter an integer to print its first 10 multiples: ").strip())
    except ValueError:
        print("Invalid input. Please enter an integer.")
    else:
        print("\nUsing for loop:")
        print_multiples_for(num)
        print("\nUsing while loop:")
        print_multiples_while(num)

# If you want another controlled-loop variant (e.g., recursion, generator, itertools),
# ask to generate that alternative implementation.
```

```
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Desktop/Mtech/AIPP/ASSIGNMENT-6/Q2.py
Enter an integer to print its first 10 multiples: 5

Using for loop:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

Using while loop:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6>
```

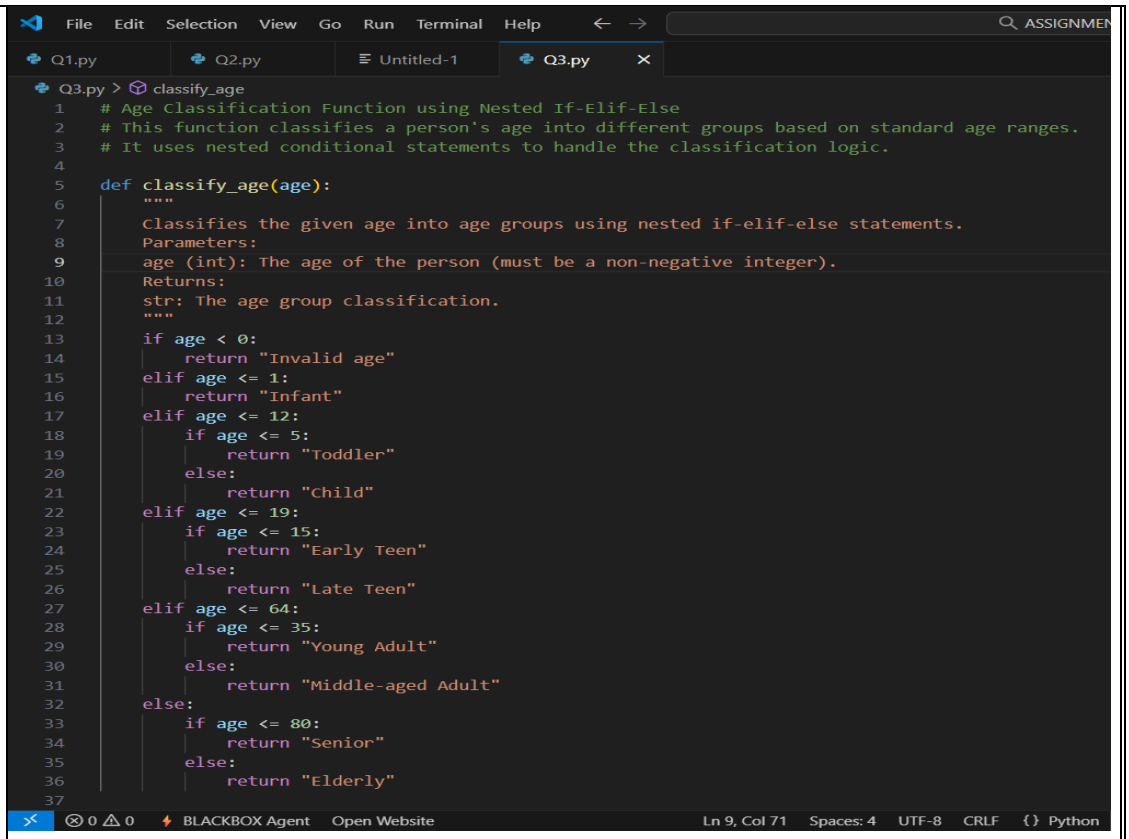
Q3.

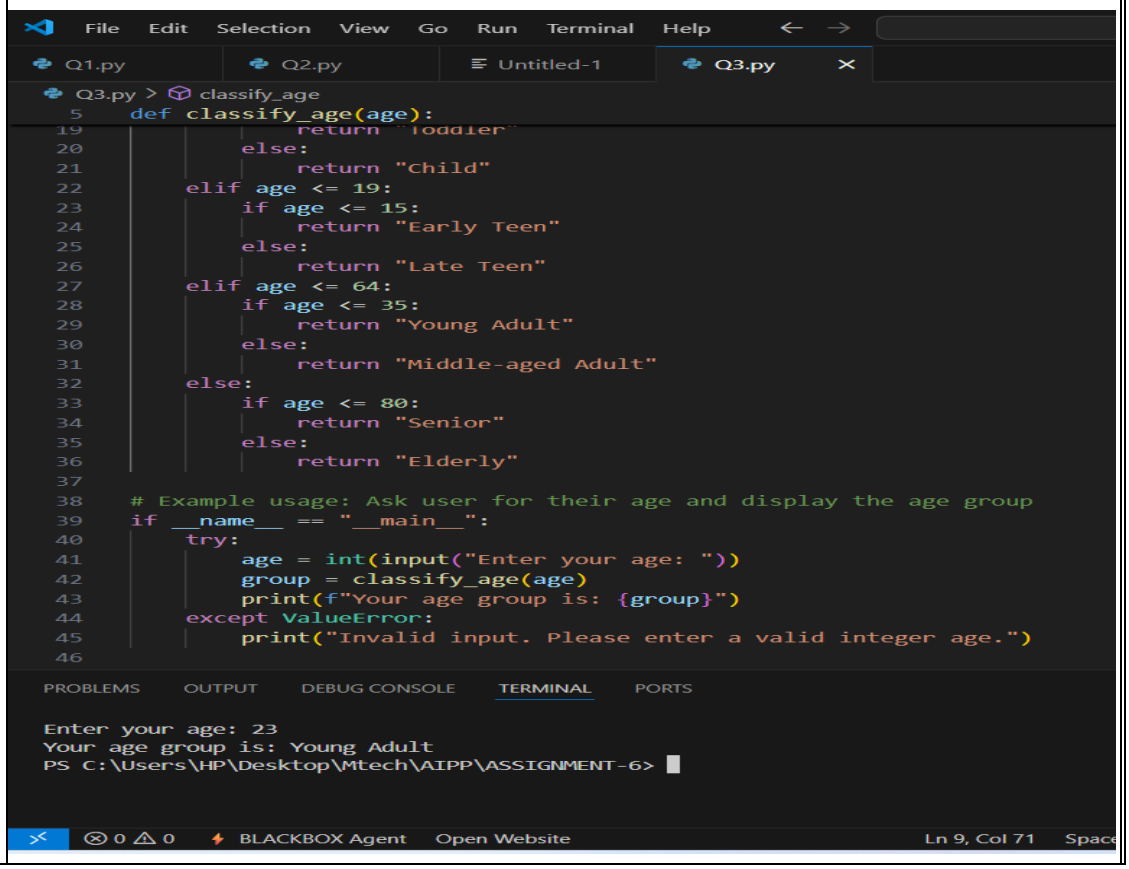
Task Description#3 (Conditional Statements)

- Ask AI to write nested if-elif-else conditionals to classify age groups.
- Analyze the generated code
- Ask AI to generate code using other conditional statements

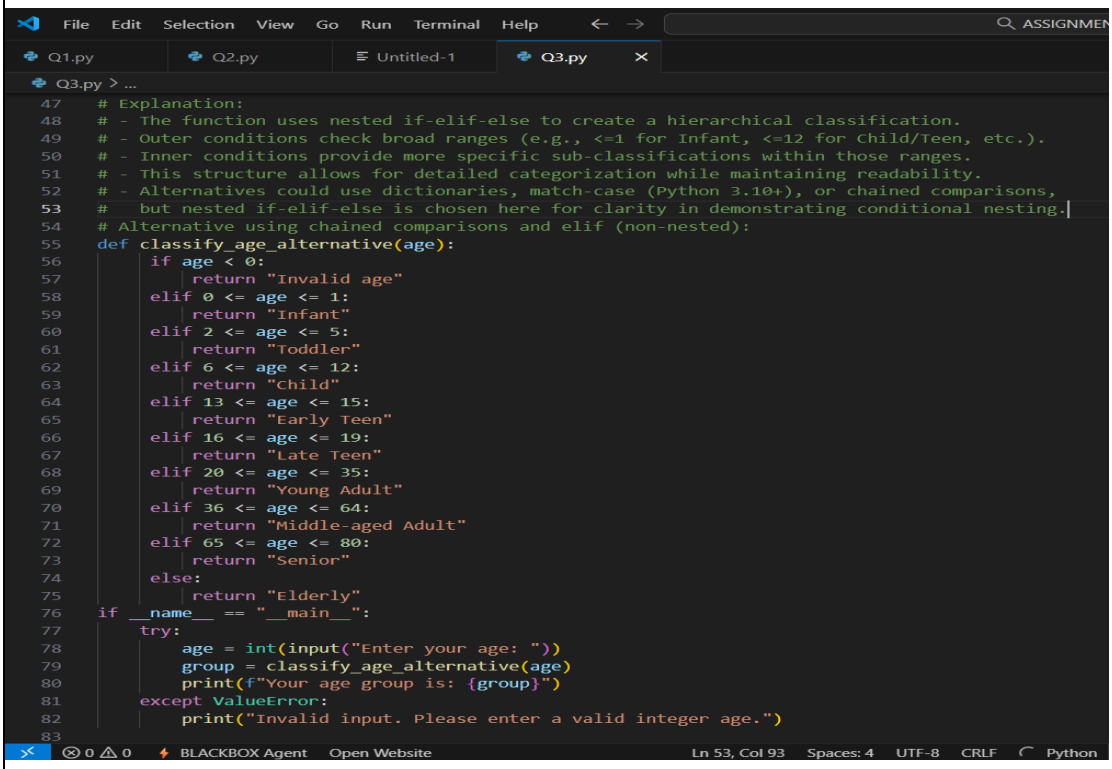
Expected Output#3

- Age classification function with appropriate conditions and with explanation


File Edit Selection View Go Run Terminal Help
Q1.py
Q2.py
Untitled-1
Q3.py
Q3.py > classify_age
# Age Classification Function using Nested If-Elif-Else
# This function classifies a person's age into different groups based on standard age ranges.
# It uses nested conditional statements to handle the classification logic.
def classify_age(age):
"""
Classifies the given age into age groups using nested if-elif-else statements.
Parameters:
age (int): The age of the person (must be a non-negative integer).
Returns:
str: The age group classification.
"""
if age < 0:
return "Invalid age"
elif age <= 1:
return "Infant"
elif age <= 12:
if age <= 5:
return "Toddler"
else:
return "Child"
elif age <= 19:
if age <= 15:
return "Early Teen"
else:
return "Late Teen"
elif age <= 64:
if age <= 35:
return "Young Adult"
else:
return "Middle-aged Adult"
else:
if age <= 80:
return "Senior"
else:
return "Elderly"
BLACKBOX Agent
Open Website
Ln 9, Col 71
Spaces: 4
UTF-8
CRLF
Python


File Edit Selection View Go Run Terminal Help
Q1.py
Q2.py
Untitled-1
Q3.py
Q3.py > classify_age
def classify_age(age):
return "Toddler"
else:
return "Child"
elif age <= 19:
if age <= 15:
return "Early Teen"
else:
return "Late Teen"
elif age <= 64:
if age <= 35:
return "Young Adult"
else:
return "Middle-aged Adult"
else:
if age <= 80:
return "Senior"
else:
return "Elderly"
# Example usage: Ask user for their age and display the age group
if __name__ == "__main__":
try:
age = int(input("Enter your age: "))
group = classify_age(age)
print(f"Your age group is: {group}")
except ValueError:
print("Invalid input. Please enter a valid integer age.")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Enter your age: 23
Your age group is: Young Adult
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6>
BLACKBOX Agent
Open Website
Ln 9, Col 71

ALTERNATIVE CODE:



```
# Explanation:
# - The function uses nested if-elif-else to create a hierarchical classification.
# - Outer conditions check broad ranges (e.g., <=1 for Infant, <=12 for Child/Teen, etc.).
# - Inner conditions provide more specific sub-classifications within those ranges.
# - This structure allows for detailed categorization while maintaining readability.
# - Alternatives could use dictionaries, match-case (Python 3.10+), or chained comparisons,
#   but nested if-elif-else is chosen here for clarity in demonstrating conditional nesting.
# Alternative using chained comparisons and elif (non-nested):
def classify_age_alternative(age):
    if age < 0:
        return "Invalid age"
    elif 0 <= age <= 1:
        return "Infant"
    elif 2 <= age <= 5:
        return "Toddler"
    elif 6 <= age <= 12:
        return "Child"
    elif 13 <= age <= 15:
        return "Early Teen"
    elif 16 <= age <= 19:
        return "Late Teen"
    elif 20 <= age <= 35:
        return "Young Adult"
    elif 36 <= age <= 64:
        return "Middle-aged Adult"
    elif 65 <= age <= 80:
        return "Senior"
    else:
        return "Elderly"
if __name__ == "__main__":
    try:
        age = int(input("Enter your age: "))
        group = classify_age_alternative(age)
        print(f"Your age group is: {group}")
    except ValueError:
        print("Invalid input. Please enter a valid integer age.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Desktop/Mtech/AIPP/ASSIGNMENT-6/Q3.py
Enter your age: 12
Your age group is: Child
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6>
```

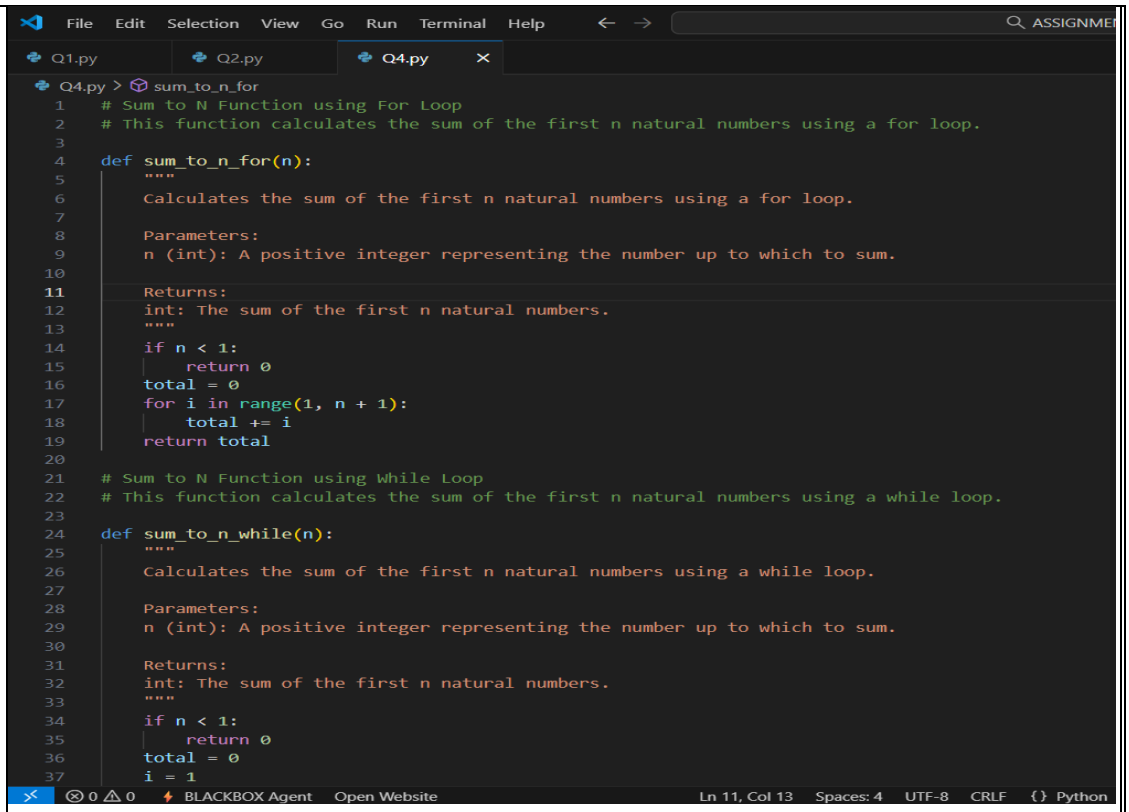
Q4.

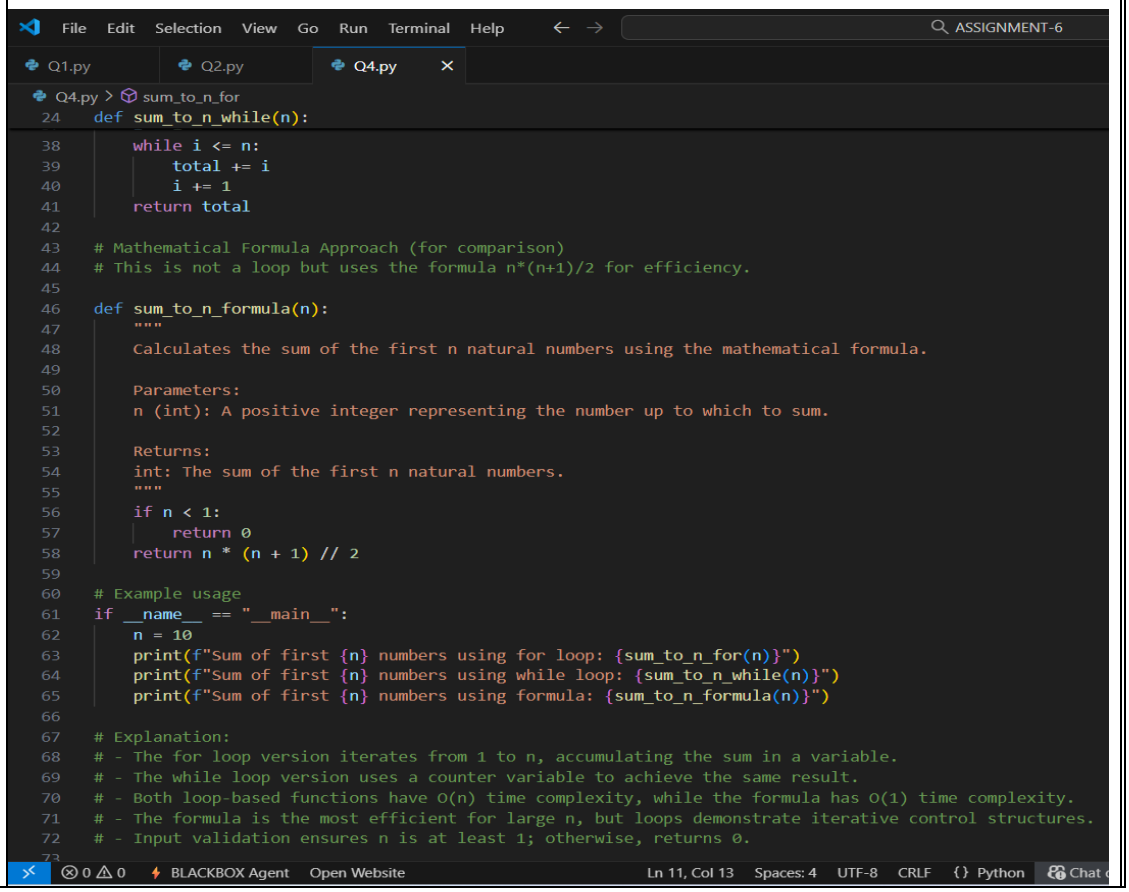
Task Description#4 (For and While loops)

- Generate a sum_to_n() function to calculate sum of first n numbers
- Analyze the generated code
- Get suggestions from AI with other controlled looping

Expected Output#4

- Python code with explanation


File Edit Selection View Go Run Terminal Help
Q1.py
Q2.py
Q4.py
Q4.py > sum_to_n_for
# Sum to N Function using For Loop
# This function calculates the sum of the first n natural numbers using a for loop.
def sum_to_n_for(n):
"""
Calculates the sum of the first n natural numbers using a for loop.
Parameters:
n (int): A positive integer representing the number up to which to sum.
Returns:
int: The sum of the first n natural numbers.
"""
if n < 1:
return 0
total = 0
for i in range(1, n + 1):
total += i
return total
# Sum to N Function using While Loop
# This function calculates the sum of the first n natural numbers using a while loop.
def sum_to_n_while(n):
"""
Calculates the sum of the first n natural numbers using a while loop.
Parameters:
n (int): A positive integer representing the number up to which to sum.
Returns:
int: The sum of the first n natural numbers.
"""
if n < 1:
return 0
total = 0
i = 1
0 0 BLACKBOX Agent Open Website
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF Python


File Edit Selection View Go Run Terminal Help
ASSIGNMENT-6
Q1.py
Q2.py
Q4.py
Q4.py > sum_to_n_for
def sum_to_n_while(n):
while i <= n:
total += i
i += 1
return total
# Mathematical Formula Approach (for comparison)
# This is not a loop but uses the formula n*(n+1)/2 for efficiency.
def sum_to_n_formula(n):
"""
Calculates the sum of the first n natural numbers using the mathematical formula.
Parameters:
n (int): A positive integer representing the number up to which to sum.
Returns:
int: The sum of the first n natural numbers.
"""
if n < 1:
return 0
return n * (n + 1) // 2
# Example usage
if __name__ == "__main__":
n = 10
print(f"Sum of first {n} numbers using for loop: {sum_to_n_for(n)}")
print(f"Sum of first {n} numbers using while loop: {sum_to_n_while(n)}")
print(f"Sum of first {n} numbers using formula: {sum_to_n_formula(n)}")
# Explanation:
# - The for loop version iterates from 1 to n, accumulating the sum in a variable.
# - The while loop version uses a counter variable to achieve the same result.
# - Both loop-based functions have O(n) time complexity, while the formula has O(1) time complexity.
# - The formula is the most efficient for large n, but loops demonstrate iterative control structures.
# - Input validation ensures n is at least 1; otherwise, returns 0.
0 0 BLACKBOX Agent Open Website
Ln 11, Col 13 Spaces: 4 UTF-8 CRLF Python Chat

```
66
67  # Explanation:
68  # - The for loop version iterates from 1 to n, accumulating the sum in a variable.
69  # - The while loop version uses a counter variable to achieve the same result.
70  # - Both loop-based functions have O(n) time complexity, while the formula has O(1) time complexity.
71  # - The formula is the most efficient for large n, but loops demonstrate iterative control structures.
72  # - Input validation ensures n is at least 1; otherwise, returns 0.
73
74  # Suggestions for other controlled looping:
75  # 1. Use recursion: def sum_to_n_recursive(n): return n + sum_to_n_recursive(n-1) if n > 0 else 0
76  # 2. Use list comprehension with sum: def sum_to_n_list(n): return sum([i for i in range(1, n+1)])
77  # 3. Use generator expression: def sum_to_n_gen(n): return sum(i for i in range(1, n+1))
78  # 4. Use reduce from functools: from functools import reduce; def sum_to_n_reduce(n): return reduce(lambda x, y: x + y, range(1, n+1), 0)
79  # These alternatives vary in readability, efficiency, and Pythonic style.

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Desktop/Mtech/AIPP/ASSIGNMENT-6/Q4.py
Sum of first 10 numbers using for loop: 55
Sum of first 10 numbers using while loop: 55
Sum of first 10 numbers using formula: 55
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6>
```

ANALYZING CODE:

1. Sum_to_n_for(n) – uses a for loop to iterate from 1 to *n* and accumulate the sum.
2. Sum_to_n_while(n) – performs the same task using a while loop with a counter variable.
3. Sum_to_n_formula(n) – applies the mathematical formula n*(n+1)//2, which is the most efficient (O(1) time).
4. All functions check if *n* is less than 1 and return 0 in that case.
5. In the main section, the code demonstrates each method for *n = 10* and prints the results.

Overall, it effectively compares iterative and formula-based approaches, showing how different control structures can solve the same problem with varying efficiency.

Q5.

Task Description#5 (Class)

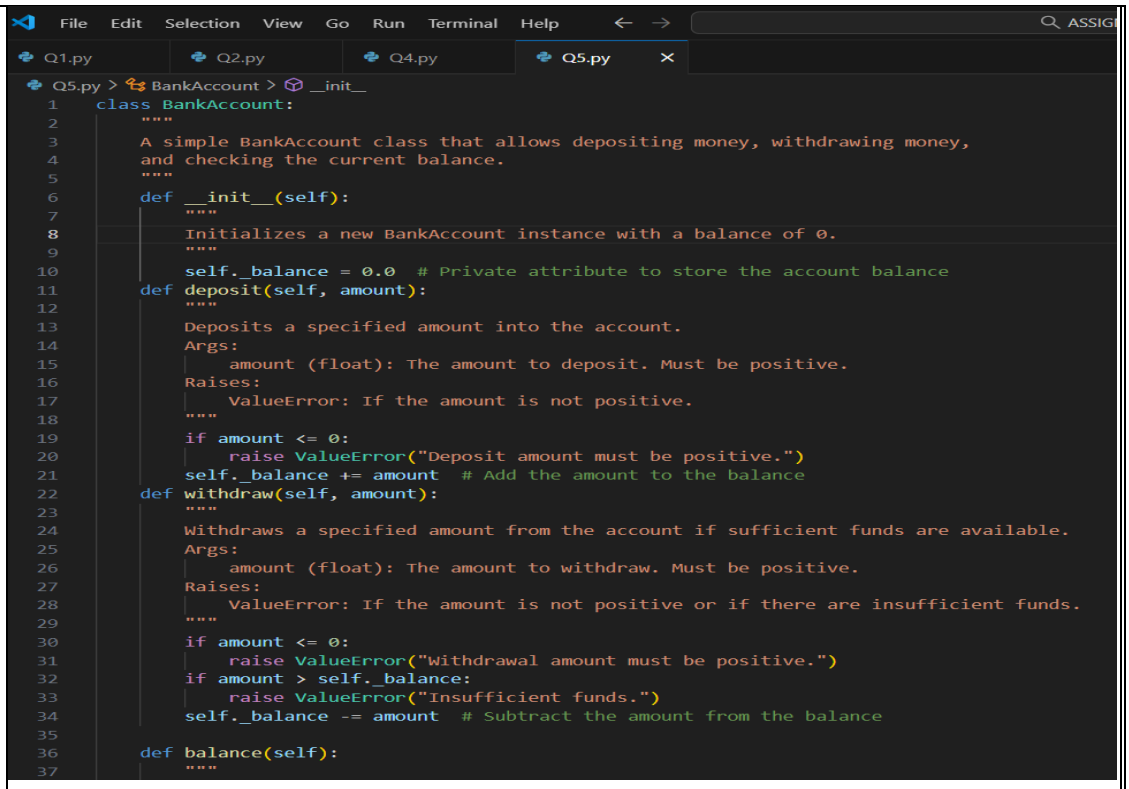
- Use AI to build a BankAccount class with deposit, withdraw, and balance methods.
- Analyze the generated code
- Add comments and explain code

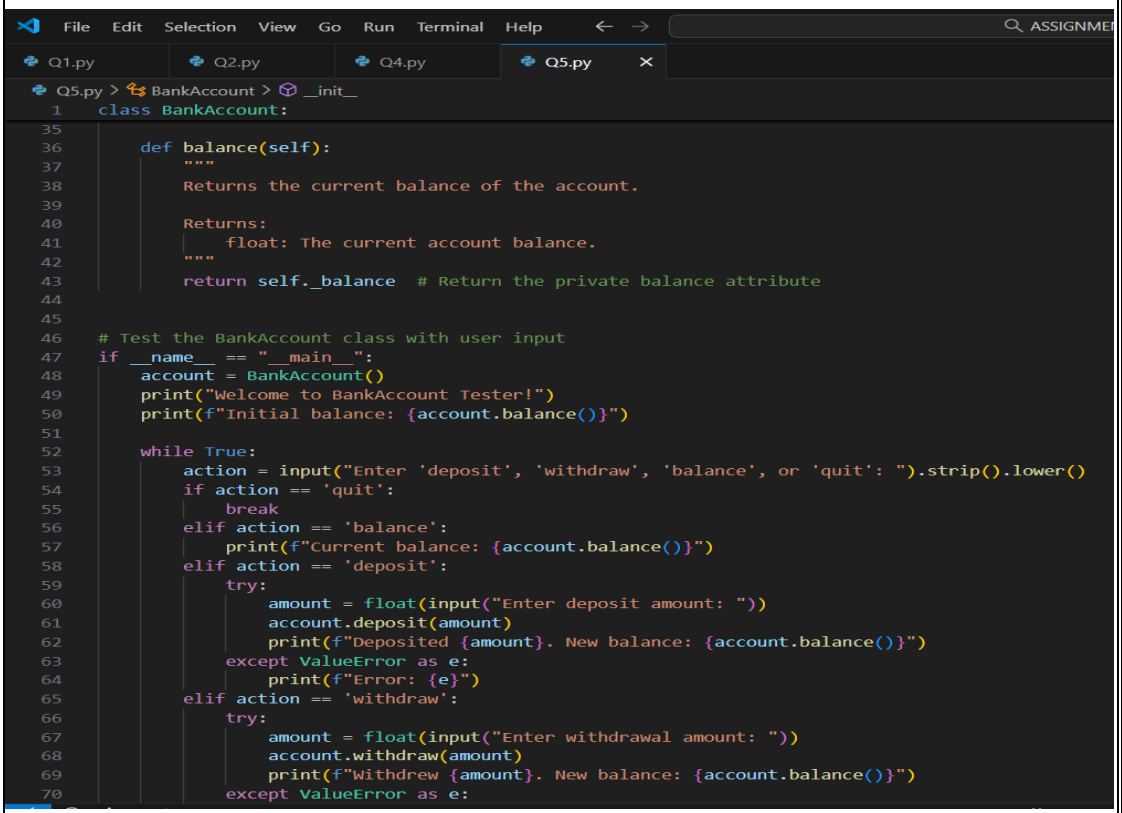
Expected Output#5

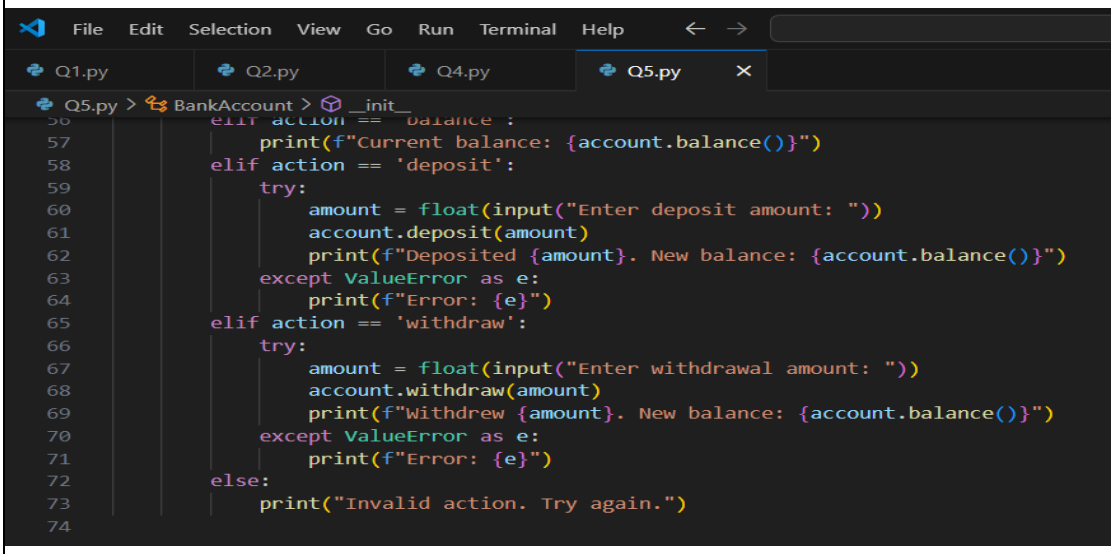
- Python code with explanation

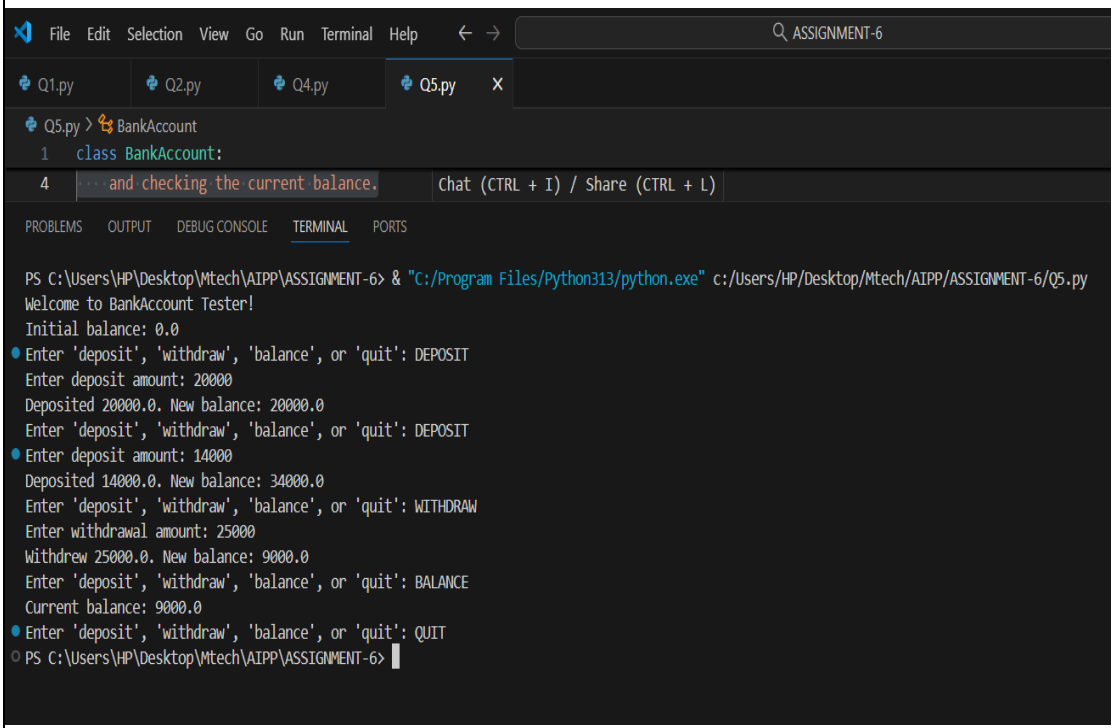
PROMPT:

A simple BankAccount class that allows depositing money, withdrawing money, and checking the current balance.


File Edit Selection View Go Run Terminal Help
Q1.py Q2.py Q4.py Q5.py
Q5.py > BankAccount > __init__
class BankAccount:
"""
A simple BankAccount class that allows depositing money, withdrawing money,
and checking the current balance.
"""
def __init__(self):
"""
Initializes a new BankAccount instance with a balance of 0.
"""
self._balance = 0.0 # Private attribute to store the account balance
def deposit(self, amount):
"""
Deposits a specified amount into the account.
Args:
amount (float): The amount to deposit. Must be positive.
Raises:
ValueError: If the amount is not positive.
"""
if amount <= 0:
raise ValueError("Deposit amount must be positive.")
self._balance += amount # Add the amount to the balance
def withdraw(self, amount):
"""
Withdraws a specified amount from the account if sufficient funds are available.
Args:
amount (float): The amount to withdraw. Must be positive.
Raises:
ValueError: If the amount is not positive or if there are insufficient funds.
"""
if amount <= 0:
raise ValueError("Withdrawal amount must be positive.")
if amount > self._balance:
raise ValueError("Insufficient funds.")
self._balance -= amount # Subtract the amount from the balance
def balance(self):
"""


File Edit Selection View Go Run Terminal Help
Q1.py Q2.py Q4.py Q5.py
Q5.py > BankAccount > __init__
class BankAccount:
def balance(self):
"""
Returns the current balance of the account.
Returns:
float: The current account balance.
"""
return self._balance # Return the private balance attribute
# Test the BankAccount class with user input
if __name__ == "__main__":
account = BankAccount()
print("Welcome to BankAccount Tester!")
print(f"Initial balance: {account.balance()}")
while True:
action = input("Enter 'deposit', 'withdraw', 'balance', or 'quit': ").strip().lower()
if action == 'quit':
break
elif action == 'balance':
print(f"Current balance: {account.balance()}")
elif action == 'deposit':
try:
amount = float(input("Enter deposit amount: "))
account.deposit(amount)
print(f"Deposited {amount}. New balance: {account.balance()}")
except ValueError as e:
print(f"Error: {e}")
elif action == 'withdraw':
try:
amount = float(input("Enter withdrawal amount: "))
account.withdraw(amount)
print(f"Withdrew {amount}. New balance: {account.balance()}")
except ValueError as e:


File Edit Selection View Go Run Terminal Help
Q1.py Q2.py Q4.py Q5.py
Q5.py > BankAccount > __init__
print(f"Current balance: {account.balance()}")
elif action == 'deposit':
try:
amount = float(input("Enter deposit amount: "))
account.deposit(amount)
print(f"Deposited {amount}. New balance: {account.balance()}")
except ValueError as e:
print(f"Error: {e}")
elif action == 'withdraw':
try:
amount = float(input("Enter withdrawal amount: "))
account.withdraw(amount)
print(f"Withdrew {amount}. New balance: {account.balance()}")
except ValueError as e:
print(f"Error: {e}")
else:
print("Invalid action. Try again.")


File Edit Selection View Go Run Terminal Help
ASSIGNMENT-6
Q1.py Q2.py Q4.py Q5.py
Q5.py > BankAccount
class BankAccount:
and checking the current balance.
Chat (CTRL + I) / Share (CTRL + L)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6> & "C:/Program Files/Python313/python.exe" c:/Users/HP/Desktop/Mtech/AIPP/ASSIGNMENT-6/Q5.py
Welcome to BankAccount Tester!
Initial balance: 0.0
Enter 'deposit', 'withdraw', 'balance', or 'quit': DEPOSIT
Enter deposit amount: 20000
Deposited 20000.0. New balance: 20000.0
Enter 'deposit', 'withdraw', 'balance', or 'quit': DEPOSIT
Enter deposit amount: 14000
Deposited 14000.0. New balance: 34000.0
Enter 'deposit', 'withdraw', 'balance', or 'quit': WITHDRAW
Enter withdrawal amount: 25000
Withdrew 25000.0. New balance: 9000.0
Enter 'deposit', 'withdraw', 'balance', or 'quit': BALANCE
Current balance: 9000.0
Enter 'deposit', 'withdraw', 'balance', or 'quit': QUIT
PS C:\Users\HP\Desktop\Mtech\AIPP\ASSIGNMENT-6>