AI ASSISTED CODING

# ASSIGNMENT 6.3

Name: Aluwala Sharanya

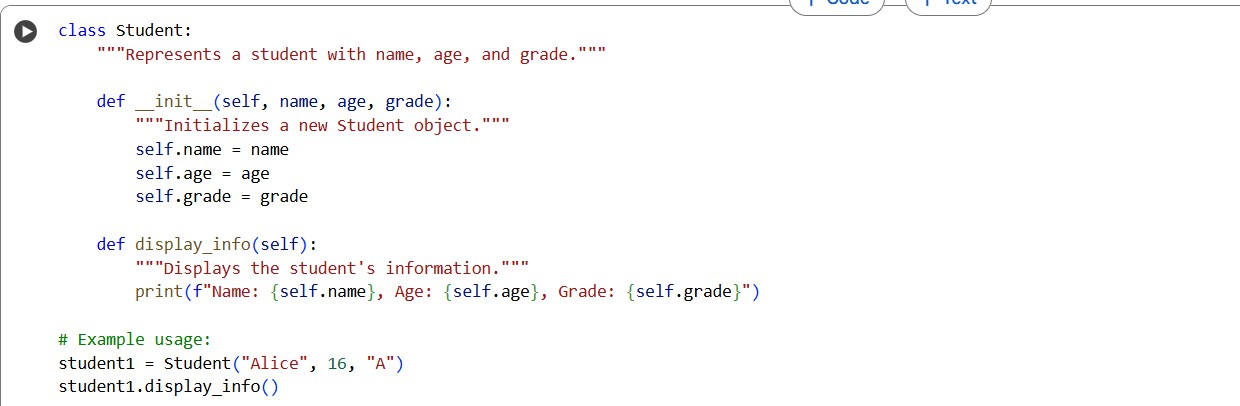
Ht No: 2403A52013

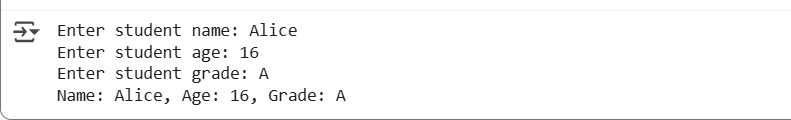
BATCH : 02

# TASK 1

* Use AI to complete a Student class with attributes and a method.
* Check output
* Analyze the code generated by AI tool PROMPT:

Use AI to generate a Python Student class with attributes and one method, run it to check output, and analyze the code

CODE:

OUTPUT:

OBSERVATION:

The code defines a Student class with attributes for name, age, and grade. It takes user input to create a

student object, converting age to an integer. Finally, it uses a method to display the student's details, as shown in the output.

CONCLUSION:

we have successfully defined a Student class in Python, which can store a student's name, age, and grade. The code allows for interactive input from the user to create

a Student object and then displays the stored information using a dedicated method. This demonstrates the basic principles of object-oriented programming with classes, attributes, and methods.

# TASK 2

* Prompt AI to complete a function that prints the first 10 multiples of a number using a

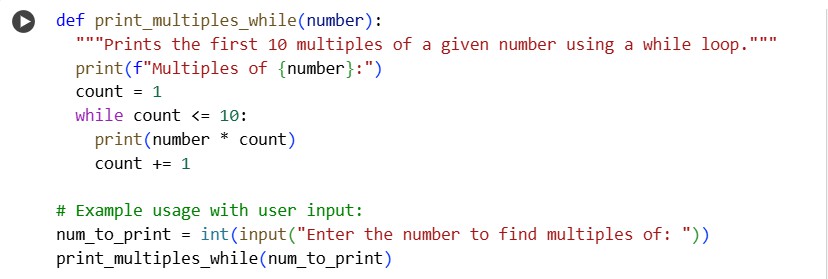
loop.

* Analyze the generated code
* Ask AI to generate code using other controlled looping PROMPT:

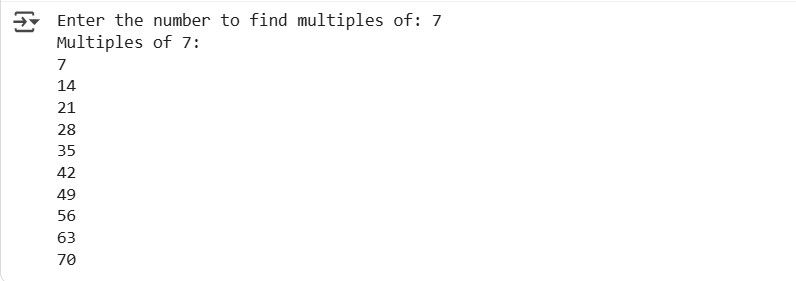
Write a function that prints the first 10 multiples of a given number using a loop. Check and analyze the

generated code. Again generate the same program using a different looping construct (e.g., while instead of for).

CODE:



OUTPUT:

OBSERVATION:

The code defines a print\_multiples\_while function using a while loop to print the first 10 multiples of a number. It takes user input for the number, converts it to an integer, and then calls the function to display the multiples.

CONCLUSION:

The code successfully implements a function using

a while loop to find and print the first 10 multiples of a

number. By incorporating user input, the code becomes interactive, allowing the user to specify the number and see the results directly, demonstrating a practical

application of while loops and user interaction in Python.

# TASK 3

* 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

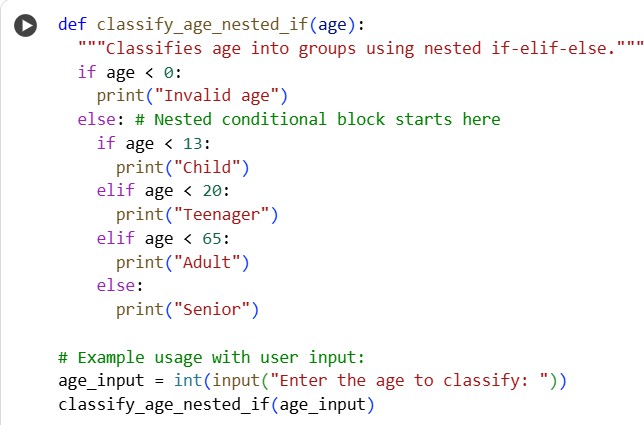
PROMPT:

Write a Python program with nested if-elif-else

conditionals to classify age groups. Check and analyze the generated code. Then, rewrite the program using other conditional approaches (e.g., match-case or logical

operators)

CODE:



OUTPUT:

OBSERVATION:

The code defines a function classify\_age\_nested\_if that uses nested if-elif-else statements to categorize age. It takes user input for age, handles invalid negative input,

and correctly classifies ages into Child, Teenager, Adult, or Senior groups.

CONCLUSION:

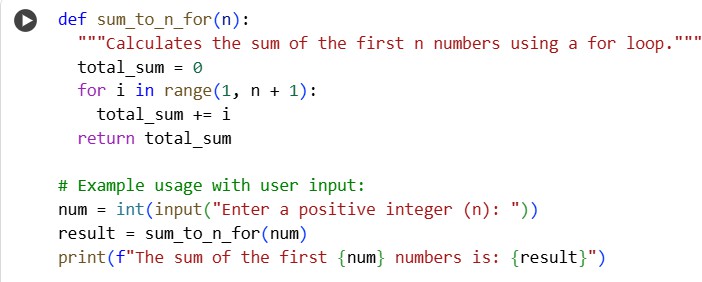
In conclusion, the code successfully demonstrates age group classification using nested if-elif-else statements and handles user input. It effectively categorizes ages based on defined ranges, illustrating a fundamental conditional control flow structure in Python.

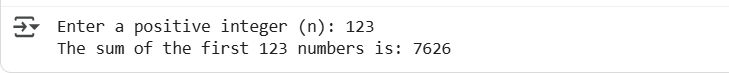
# TASK 4

* 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 PROMPT:

Generate a sum\_to\_n() function in Python to calculate the sum of the first n numbers using a loop? Then show

how it can be written using a different looping construct. CODE:



OUTPUT:

OBSERVATION:

The code defines a function sum\_to\_n\_for using

a for loop to calculate the sum of the first 'n' numbers. It takes user input for 'n', iterates from 1 to 'n', accumulates the sum, and returns the result, which is then displayed

CONCLUSION:

The code effectively calculates the sum of the first 'n' positive integers using a for loop and user input. It demonstrates a fundamental iterative approach to

summation, providing a clear and functional solution to the task

# TASK 5

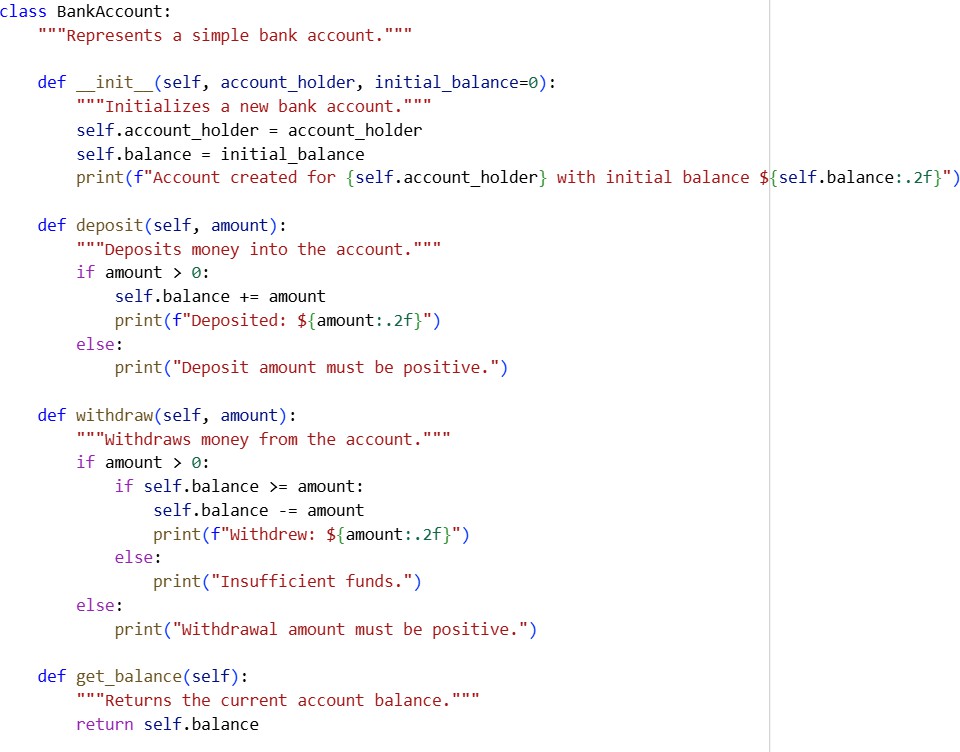
* Use AI to build a BankAccount class with deposit, withdraw, and balance methods.
* Analyze the generated code
* Add comments and explain code

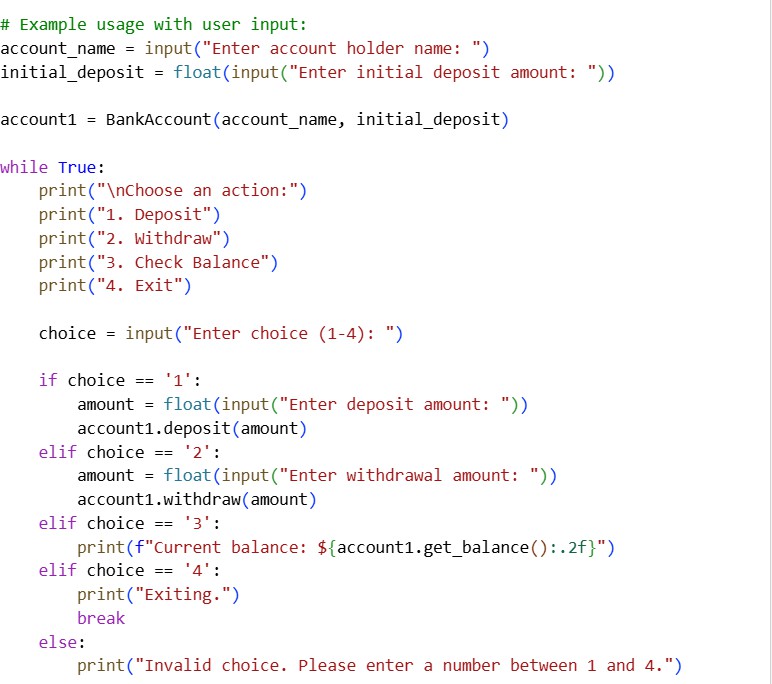
PROMPT:

Can you build a BankAccount class in Python with deposit, withdraw, and balance methods? After

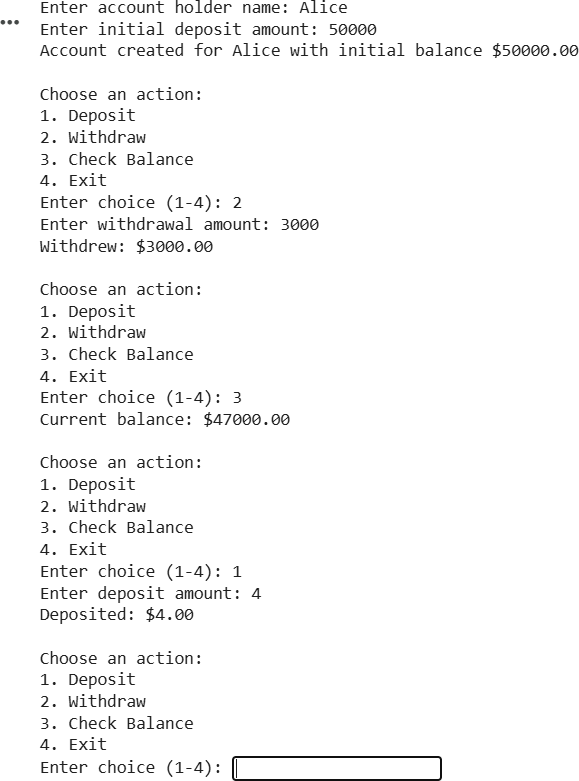
generating the code, analyze it, add comments, and explain how it works.

CODE:





OUTPUT:



OBSERVATION:

The code defines a BankAccount class with deposit, withdraw, and balance methods. It interactively takes user input for account actions. The output shows

successful deposit, withdrawal, and balance checks based on user choices, demonstrating basic bank transactions.

CONCLUSION:

The code provides a functional simulation of a bank

account using object-oriented programming. It allows users to perform core transactions like deposit and

withdrawal and check their balance interactively, demonstrating class methods and user input handling.