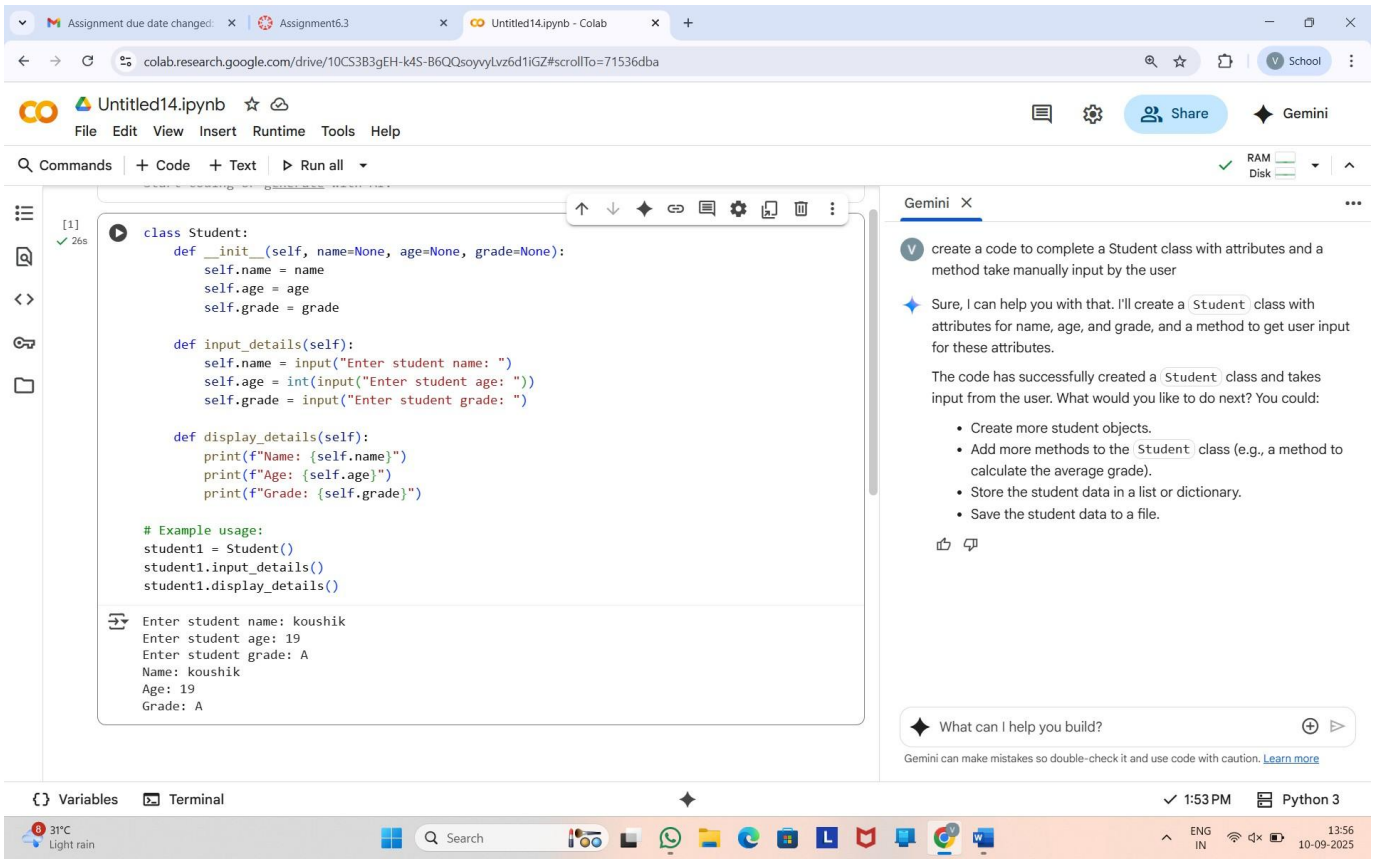
AI-Assisted Coding
Name:Peravena Navadeep
Htno:2403A52019

Task-1:

Prompt: Create a code to complete a Student class with attributes and a method take manually input by the user.

Code&Output:

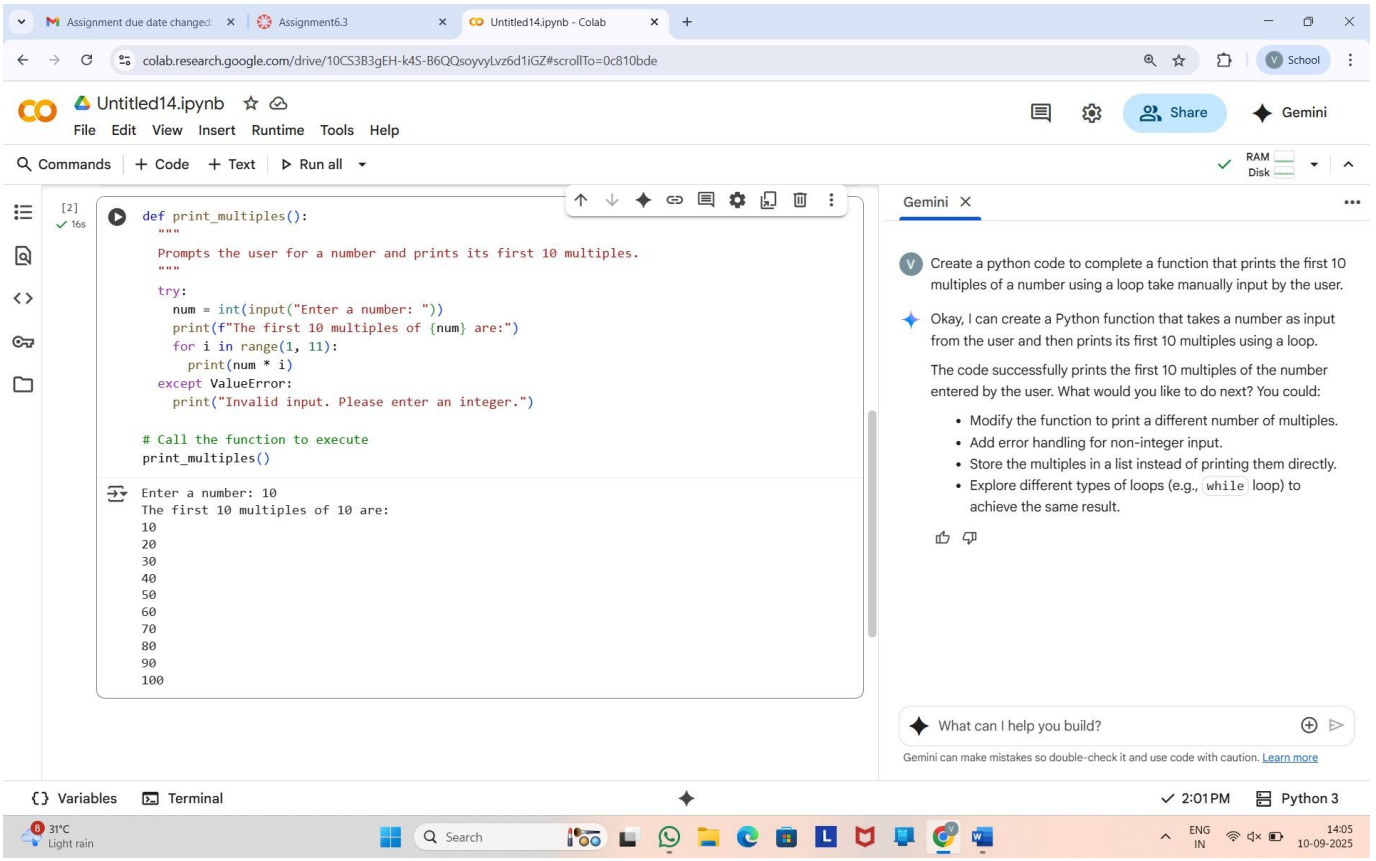


Observation: The task successfully demonstrated how to create a Python class with attributes and a method that takes user input to populate those attributes. This is a fundamental concept in object-oriented programming for creating custom data structures and behaviors.

Task-2:

Prompt: Create a python code to complete a function that prints the first 10 multiples of a number using a loop take manually input by the user.

Code&Output:

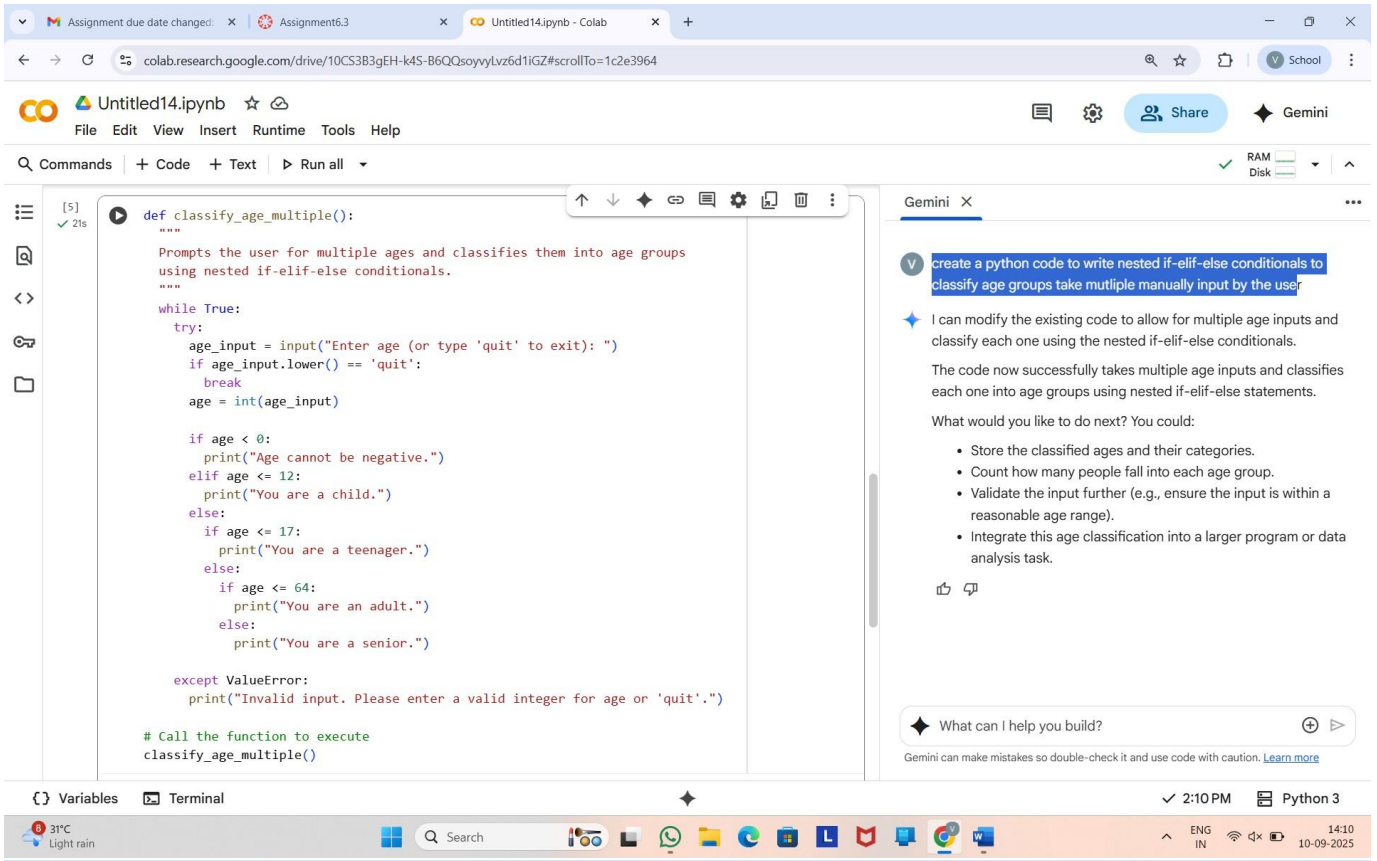


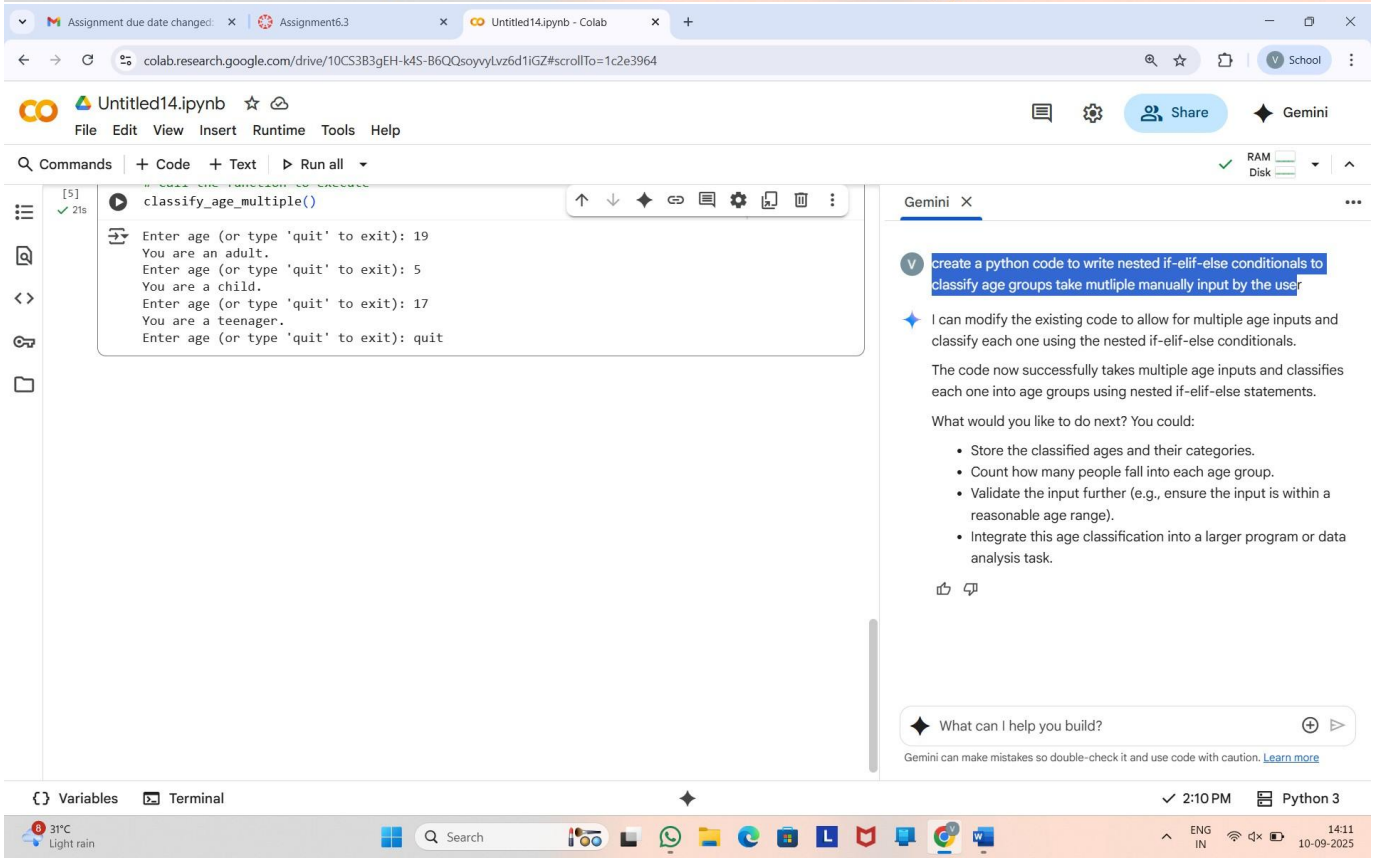
Observation: This task successfully demonstrated how to define a Python function that takes user input and uses a for loop to iterate and perform a calculation (multiplication) a specific number of times. It also included basic error handling for invalid input

Task-3:

Prompt: Create a python code to write nested if-elifelse conditionals to classify age groups take mutliple manually input by the user.

Code&Output:


Assignment due date changed:
Assignment6.3
Untitled14.ipynb - Colab
colab.research.google.com/drive/10CS3B3gEH-k4S-B6QQsoyvyLvz6d1iGZ#scrollTo=1c2e3964
School
Untitled14.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[5]
21s
def classify_age_multiple():
"""
Prompts the user for multiple ages and classifies them into age groups
using nested if-elif-else conditionals.
"""
while True:
try:
age_input = input("Enter age (or type 'quit' to exit): ")
if age_input.lower() == 'quit':
break
age = int(age_input)
if age < 0:
print("Age cannot be negative.")
elif age <= 12:
print("You are a child.")
else:
if age <= 17:
print("You are a teenager.")
else:
if age <= 64:
print("You are an adult.")
else:
print("You are a senior.")
except ValueError:
print("Invalid input. Please enter a valid integer for age or 'quit'.")
# Call the function to execute
classify_age_multiple()
Gemini
create a python code to write nested if-elif-else conditionals to classify age groups take mutliple manually input by the user
I can modify the existing code to allow for multiple age inputs and classify each one using the nested if-elif-else conditionals.
The code now successfully takes multiple age inputs and classifies each one into age groups using nested if-elif-else statements.
What would you like to do next? You could:
Store the classified ages and their categories.
Count how many people fall into each age group.
Validate the input further (e.g., ensure the input is within a reasonable age range).
Integrate this age classification into a larger program or data analysis task.
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
2:10 PM
Python 3
31°C
Light rain
Search
ENG
IN
14:10
10-09-2025

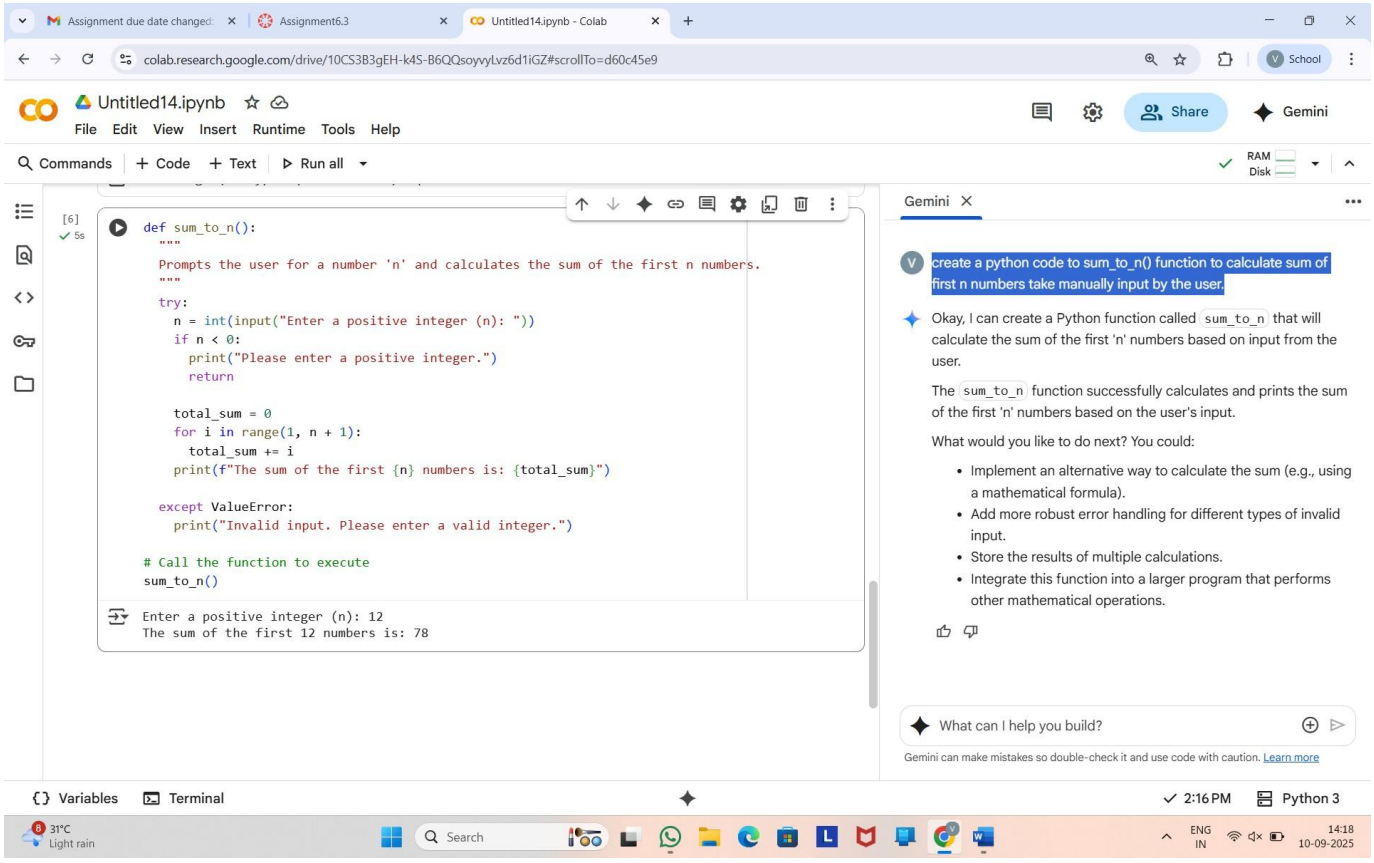

Assignment due date changed:
Assignment6.3
Untitled14.ipynb - Colab
colab.research.google.com/drive/10CS3B3gEH-k4S-B6QQsoyvyLvz6d1iGZ#scrollTo=1c2e3964
School
Untitled14.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[5]
21s
classify_age_multiple()
Enter age (or type 'quit' to exit): 19
You are an adult.
Enter age (or type 'quit' to exit): 5
You are a child.
Enter age (or type 'quit' to exit): 17
You are a teenager.
Enter age (or type 'quit' to exit): quit
Gemini
create a python code to write nested if-elif-else conditionals to classify age groups take mutliple manually input by the user
I can modify the existing code to allow for multiple age inputs and classify each one using the nested if-elif-else conditionals.
The code now successfully takes multiple age inputs and classifies each one into age groups using nested if-elif-else statements.
What would you like to do next? You could:
Store the classified ages and their categories.
Count how many people fall into each age group.
Validate the input further (e.g., ensure the input is within a reasonable age range).
Integrate this age classification into a larger program or data analysis task.
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
2:10 PM
Python 3
31°C
Light rain
Search
ENG
IN
14:11
10-09-2025

Observation: This task successfully demonstrates how to use a while loop to repeatedly take user input and apply nested if-elif-else conditions to classify each input. It also includes a mechanism for the user to exit the input loop.

Task-4:

Prompt: Create a python code to sum_to_n() function to calculate sum of first n numbers take manually input by the user.

Code&Output:

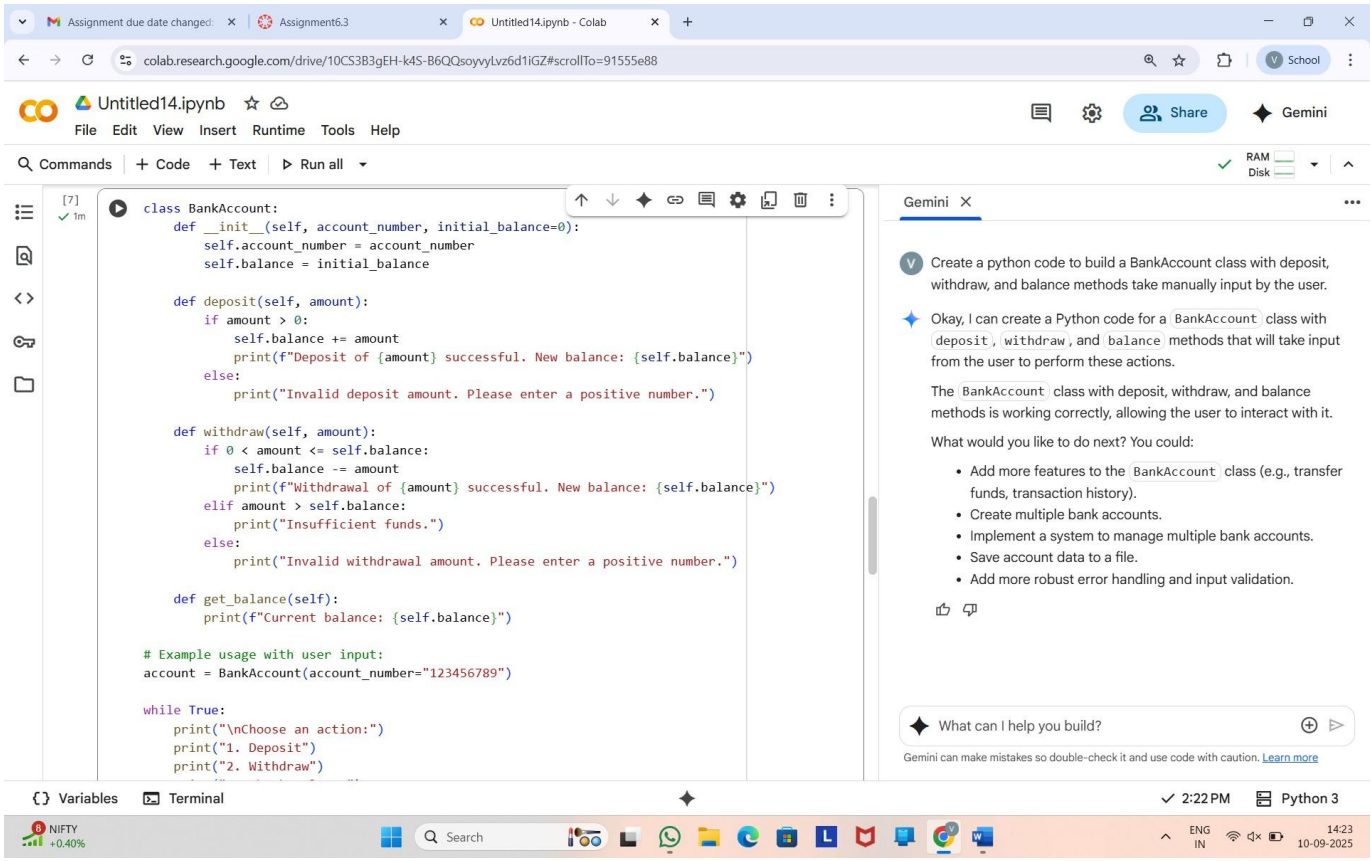


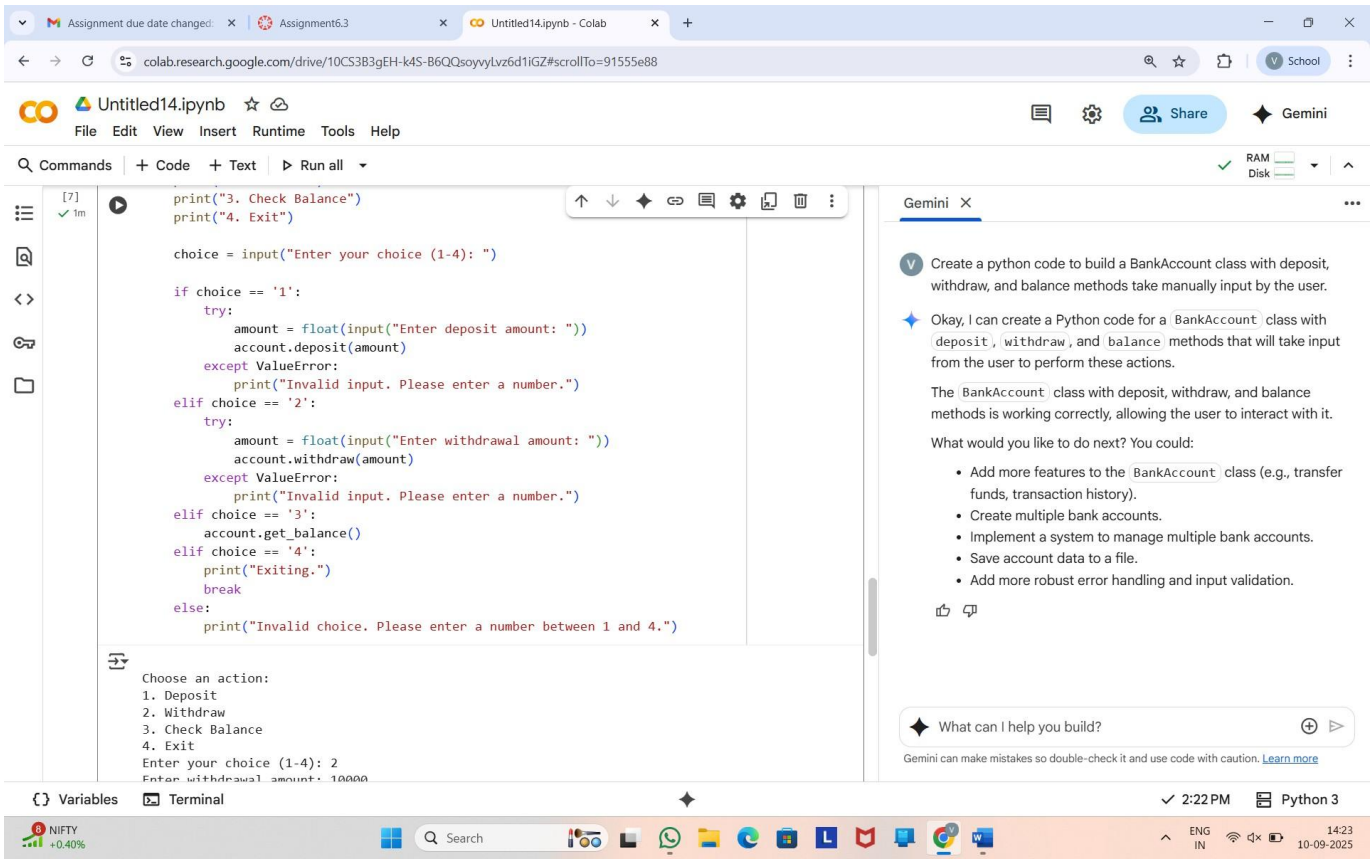
Observation: This task successfully demonstrates how to define a Python function that takes user input, validates it, and uses a for loop to iterate and accumulate a sum. This is a common pattern for performing calculations over a range of numbers.

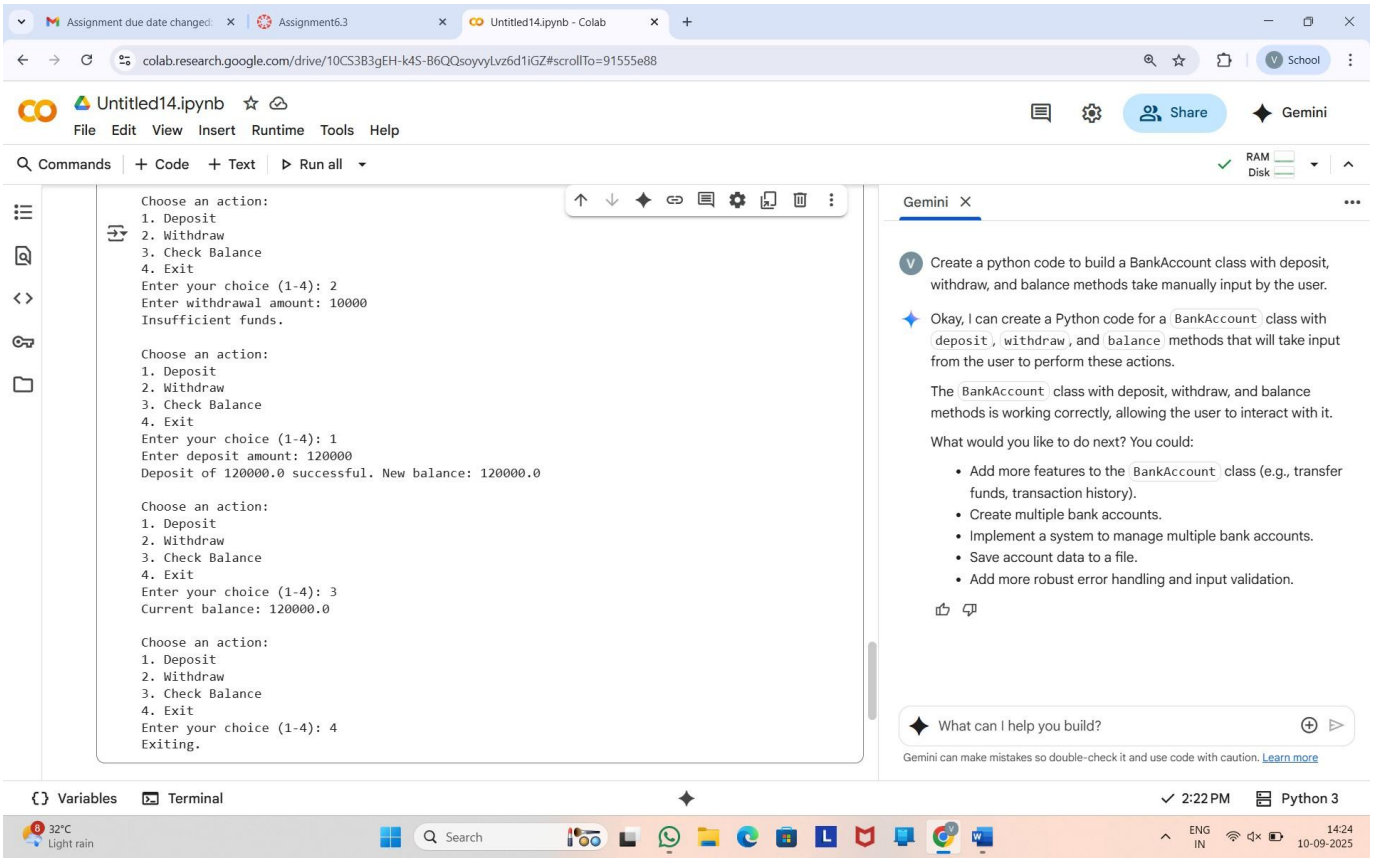
Task-5:

Prompt: Create a python code to build a BankAccount class with deposit, withdraw, and balance methods take manually input by the user.

Code&Output:


Assignment due date changed:
Assignment6.3
Untitled14.ipynb - Colab
colab.research.google.com/drive/10CS3B3gEH-k4S-B6QQsoyvyLvz6d1iGZ#scrollTo=91555e88
Untitled14.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
[7]
1m
class BankAccount:
def __init__(self, account_number, initial_balance=0):
self.account_number = account_number
self.balance = initial_balance
def deposit(self, amount):
if amount > 0:
self.balance += amount
print(f"Deposit of {amount} successful. New balance: {self.balance}")
else:
print("Invalid deposit amount. Please enter a positive number.")
def withdraw(self, amount):
if 0 < amount <= self.balance:
self.balance -= amount
print(f"Withdrawal of {amount} successful. New balance: {self.balance}")
elif amount > self.balance:
print("Insufficient funds.")
else:
print("Invalid withdrawal amount. Please enter a positive number.")
def get_balance(self):
print(f"Current balance: {self.balance}")
# Example usage with user input:
account = BankAccount(account_number="123456789")
while True:
print("\nChoose an action:")
print("1. Deposit")
print("2. Withdraw")
Gemini
Create a python code to build a BankAccount class with deposit, withdraw, and balance methods take manually input by the user.
Okay, I can create a Python code for a BankAccount class with deposit, withdraw, and balance methods that will take input from the user to perform these actions.
The BankAccount class with deposit, withdraw, and balance methods is working correctly, allowing the user to interact with it.
What would you like to do next? You could:
Add more features to the BankAccount class (e.g., transfer funds, transaction history).
Create multiple bank accounts.
Implement a system to manage multiple bank accounts.
Save account data to a file.
Add more robust error handling and input validation.
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
2:22 PM
Python 3
NIFTY +0.40%
Search
ENG IN
14:23 10-09-2025


Assignment due date changed:
Assignment6.3
Untitled14.ipynb - Colab
colab.research.google.com/drive/10CS3B3gEH-k4S-B6QQsoyvyLvz6d1iGZ#scrollTo=91555e88
Untitled14.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
[7]
1m
print("3. Check Balance")
print("4. Exit")
choice = input("Enter your choice (1-4): ")
if choice == '1':
try:
amount = float(input("Enter deposit amount: "))
account.deposit(amount)
except ValueError:
print("Invalid input. Please enter a number.")
elif choice == '2':
try:
amount = float(input("Enter withdrawal amount: "))
account.withdraw(amount)
except ValueError:
print("Invalid input. Please enter a number.")
elif choice == '3':
account.get_balance()
elif choice == '4':
print("Exiting.")
break
else:
print("Invalid choice. Please enter a number between 1 and 4.")
Choose an action:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Gemini
Create a python code to build a BankAccount class with deposit, withdraw, and balance methods take manually input by the user.
Okay, I can create a Python code for a BankAccount class with deposit, withdraw, and balance methods that will take input from the user to perform these actions.
The BankAccount class with deposit, withdraw, and balance methods is working correctly, allowing the user to interact with it.
What would you like to do next? You could:
Add more features to the BankAccount class (e.g., transfer funds, transaction history).
Create multiple bank accounts.
Implement a system to manage multiple bank accounts.
Save account data to a file.
Add more robust error handling and input validation.
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
2:22 PM
Python 3
NIFTY +0.40%
Search
ENG IN
14:23 10-09-2025



Observation: This task successfully demonstrates the creation of a Python class with multiple methods (deposit, withdraw, get_balance) that encapsulate data (balance) and behavior. It also incorporates a loop for continuous user interaction and basic error handling for invalid input, simulating a simple banking application.