

Ass-6.3

ID:2303A51940

Task Description #1: Classes (Student Class)

Scenario

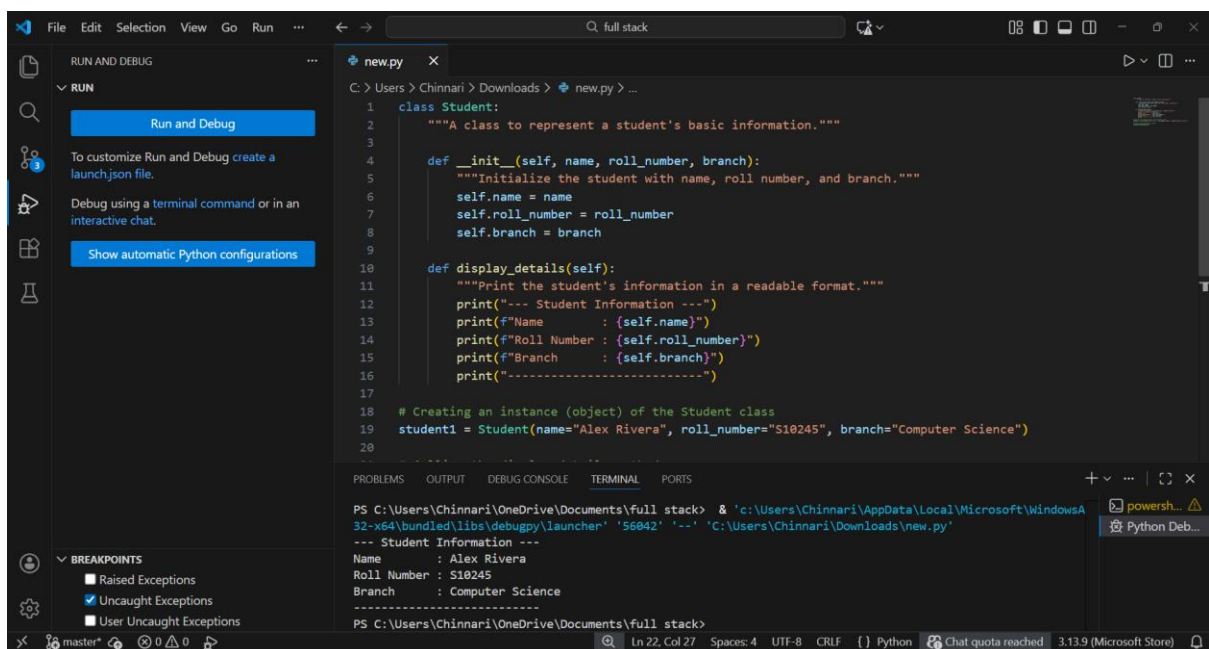
You are developing a simple student information management module.

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



```
1 class Student:
2     """A class to represent a student's basic information."""
3
4     def __init__(self, name, roll_number, branch):
5         """Initialize the student with name, roll number, and branch."""
6         self.name = name
7         self.roll_number = roll_number
8         self.branch = branch
9
10    def display_details(self):
11        """Print the student's information in a readable format."""
12        print("--- Student Information ---")
13        print(f"Name       : {self.name}")
14        print(f"Roll Number : {self.roll_number}")
15        print(f"Branch      : {self.branch}")
16        print("-----")
17
18    # Creating an instance (object) of the Student class
19    student1 = Student(name="Alex Rivera", roll_number="S10245", branch="Computer Science")
20
```

```
PS C:\Users\Chinnari\OneDrive\Documents\full stack> & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsA
32-x64\bundle\libs\debugpy\launcher' '56042' '--' 'C:\Users\Chinnari\Downloads\new.py'
--- Student Information ---
Name       : Alex Rivera
Roll Number : S10245
Branch      : Computer Science
-----
PS C:\Users\Chinnari\OneDrive\Documents\full stack>
```

The screenshot shows a Python IDE with a dark theme. The main editor window displays a Python script named `new.py`. The script defines a `Student` class with a `display_details` method. An instance of the class, `student1`, is created with the name "Alex Rivera", roll number "S10245", and branch "Computer Science". The `display_details` method is then called on `student1`. The output of the script is shown in the terminal window at the bottom, which displays the student's information in a formatted string.

```
1 class Student:
2
3     def display_details(self):
4         """Print the student's information in a readable format."""
5         print("--- Student Information ---")
6         print(f"Name       : {self.name}")
7         print(f"Roll Number : {self.roll_number}")
8         print(f"Branch      : {self.branch}")
9         print("-----")
10
11 # Creating an instance (object) of the Student class
12 student1 = Student(name="Alex Rivera", roll_number="S10245", branch="Computer Science")
13
14 # Calling the display_details method
15 student1.display_details()
```

The terminal output shows the following text:

```
PS C:\Users\Chinnari\OneDrive\Documents\full_stack> & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsA
32-x64\bundled\libs\debugpy\launcher' '56042' '--' 'C:\Users\Chinnari\Downloads\new.py'
--- Student Information ---
Name       : Alex Rivera
Roll Number : S10245
Branch      : Computer Science
-----
PS C:\Users\Chinnari\OneDrive\Documents\full_stack>
```

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.

The screenshot shows the VS Code interface with a file named `new.py` open. The code defines a `Student` class with an `__init__` method and a `display_details` method. It also includes a `print_multiples_for` function that uses a `for` loop to print the first 10 multiples of a given number. The terminal output shows the execution of the script, where a user enters the number 22, and the program prints the first 10 multiples of 22.

```

class Student:
    """A class to represent a student's basic information."""
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def display_details(self):
        print("\n--- Student Information ---")
        print(f"Name       : {self.name}")
        print(f"Roll Number : {self.roll_number}")
        print(f"Branch      : {self.branch}")
        print("-" * 27)

def print_multiples_for(n):
    """Prints the first 10 multiples using a for loop."""
    print(f"\n[For Loop] First 10 multiples of {n}:")
    for i in range(1, 11):

```

The screenshot shows the VS Code interface with a file named `new.py` open. The code defines a `Student` class with an `__init__` method and a `display_details` method. It also includes a `print_multiples_while` function that uses a `while` loop to print the first 10 multiples of a given number. The terminal output shows the execution of the script, where a user enters the number 22, and the program prints the first 10 multiples of 22.

```

def print_multiples_while(n):
    i = 1
    while i <= 10:
        print(f"{n} x {i} = {n*i}")
        i += 1

# --- Execution ---
if __name__ == "__main__":
    # 1. Class and Object Demo
    student_obj = Student("Alex Rivera", "S10245", "Computer Science")
    student_obj.display_details()

    # 2. Multiples Demo
    try:
        val = int(input("\nEnter a number to see its multiples: "))
        print_multiples_for(val)
        print_multiples_while(val)
    except ValueError:
        print("Invalid input. Please enter an integer.")

```

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

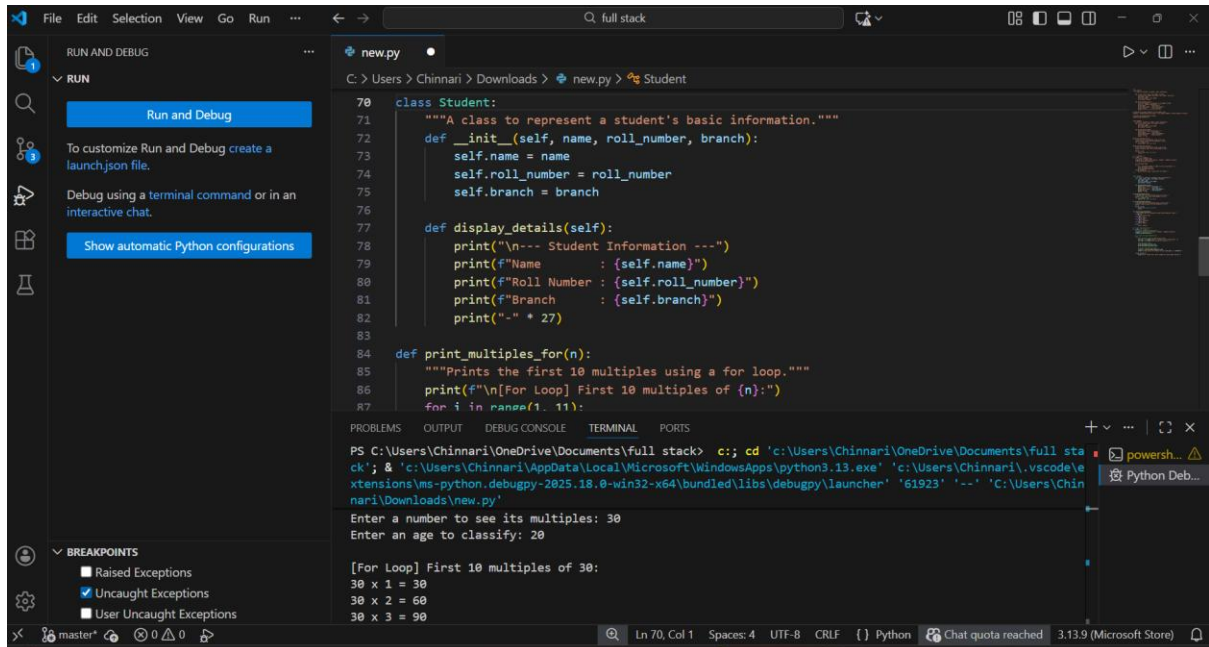
Task

- 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.



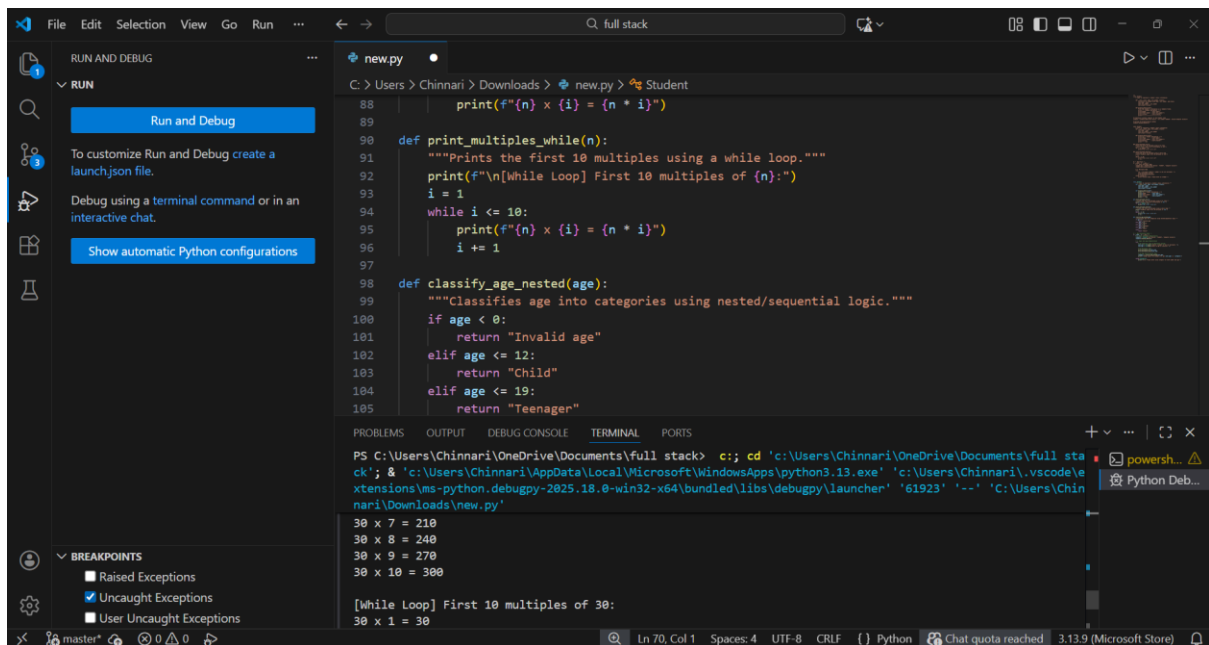
The screenshot shows the VS Code editor with a file named `new.py` open. The code defines a `Student` class with an `__init__` method and a `display_details` method. It also includes a `print_multiples_for` function that uses a `for` loop to print the first 10 multiples of a given number. The terminal output shows the program running, prompting for a number (30) and an age (20), and displaying the first 10 multiples of 30.

```
70 class Student:
71     """A class to represent a student's basic information."""
72     def __init__(self, name, roll_number, branch):
73         self.name = name
74         self.roll_number = roll_number
75         self.branch = branch
76
77     def display_details(self):
78         print("\n--- Student Information ---")
79         print(f"Name      : {self.name}")
80         print(f"Roll Number : {self.roll_number}")
81         print(f"Branch     : {self.branch}")
82         print("-" * 27)
83
84 def print_multiples_for(n):
85     """Prints the first 10 multiples using a for loop."""
86     print(f"\n[For Loop] First 10 multiples of {n}:")
87     for i in range(1, 11):
```

Terminal Output:

```
PS C:\Users\Chinnari\OneDrive\Documents\full stack> c:\cd 'c:\Users\Chinnari\OneDrive\Documents\full sta
ck'; & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\vscode\e
xtensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61923' '-' 'C:\Users\Chin
nari\Downloads\new.py'
Enter a number to see its multiples: 30
Enter an age to classify: 20

[For Loop] First 10 multiples of 30:
30 x 1 = 30
30 x 2 = 60
30 x 3 = 90
```



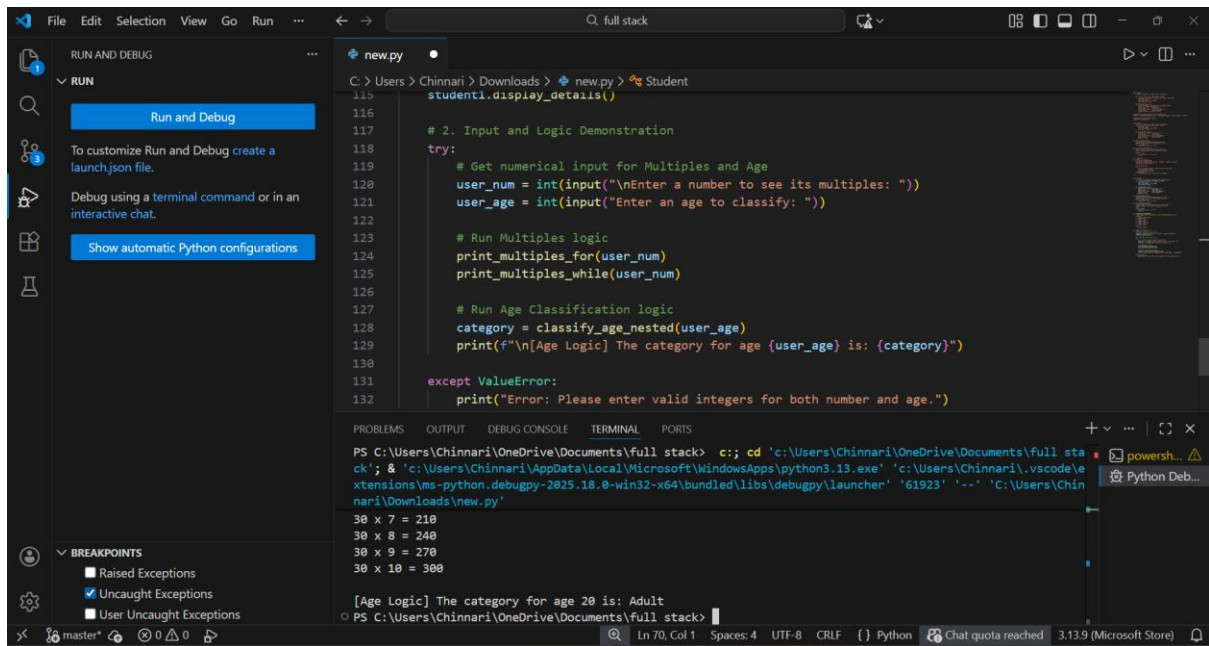
The screenshot shows the VS Code editor with the same `new.py` file. The code now includes a `print_multiples_while` function that uses a `while` loop to print the first 10 multiples of a given number. It also includes a `classify_age_nested` function that uses nested `if-elif` statements to classify an age into categories: "Invalid age", "Child", or "Teenager". The terminal output shows the program running, displaying the first 10 multiples of 30 using a while loop, and then classifying the age 30 as "Invalid age".

```
88     print(f"{n} x {i} = {n * i}")
89
90 def print_multiples_while(n):
91     """Prints the first 10 multiples using a while loop."""
92     print(f"\n[While Loop] First 10 multiples of {n}:")
93     i = 1
94     while i <= 10:
95         print(f"{n} x {i} = {n * i}")
96         i += 1
97
98 def classify_age_nested(age):
99     """Classifies age into categories using nested/sequential logic."""
100     if age < 0:
101         return "Invalid age"
102     elif age <= 12:
103         return "Child"
104     elif age <= 19:
105         return "Teenager"
```

Terminal Output:

```
PS C:\Users\Chinnari\OneDrive\Documents\full stack> c:\cd 'c:\Users\Chinnari\OneDrive\Documents\full sta
ck'; & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\vscode\e
xtensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61923' '-' 'C:\Users\Chin
nari\Downloads\new.py'
30 x 7 = 210
30 x 8 = 240
30 x 9 = 270
30 x 10 = 300

[While Loop] First 10 multiples of 30:
30 x 1 = 30
```



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

Scenario

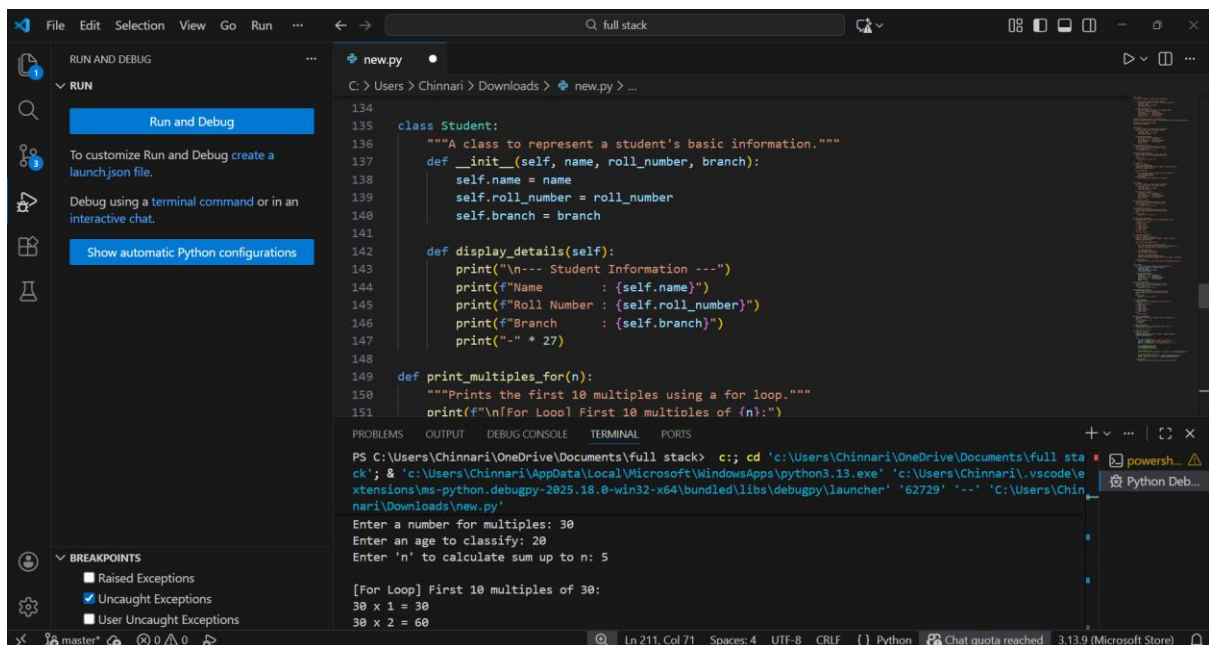
You need to calculate the sum of the first n natural numbers.

Task

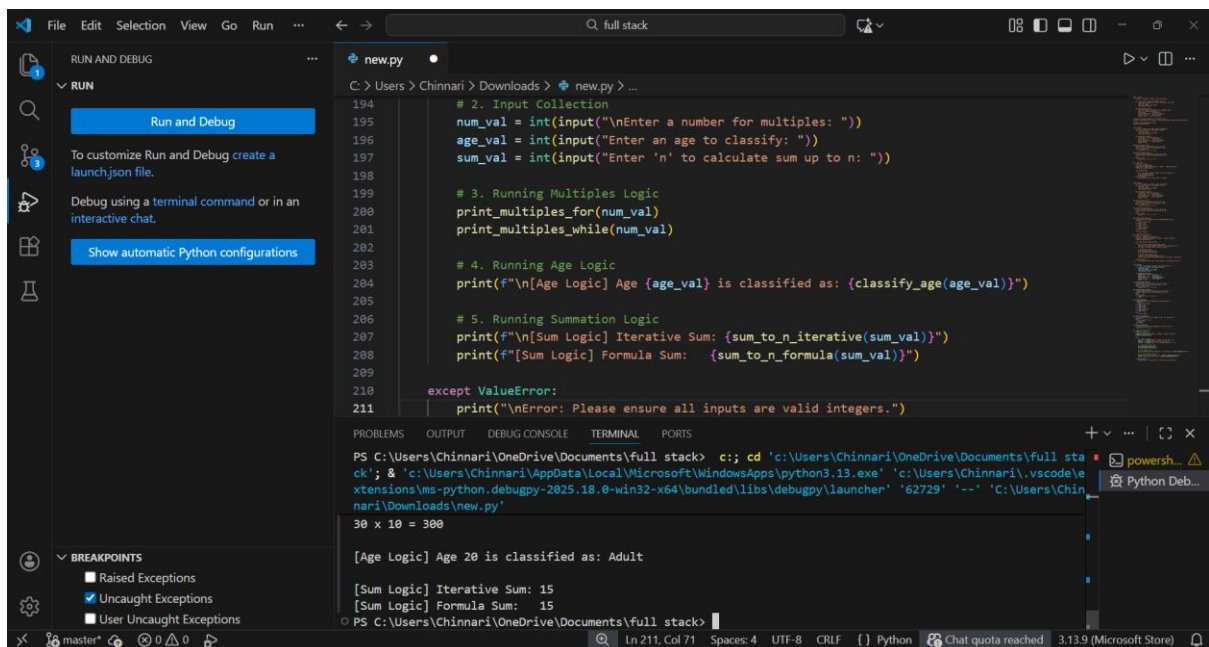
- Use AI assistance to generate a `sum_to_n()` function using a for loop.
- Analyze the generated code.
- Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

Expected Output #4

- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches.



```
134
135
136 class Student:
137     """A class to represent a student's basic information."""
138     def __init__(self, name, roll_number, branch):
139         self.name = name
140         self.roll_number = roll_number
141         self.branch = branch
142
143     def display_details(self):
144         print("\n--- Student Information ---")
145         print(f"Name      : {self.name}")
146         print(f"Roll Number : {self.roll_number}")
147         print(f"Branch    : {self.branch}")
148         print("-" * 27)
149
150 def print_multiples_for(n):
151     """Prints the first 10 multiples using a for loop."""
152     print(f"\n[For Loop] First 10 multiples of {n}:")
153
154 PS C:\Users\Chinnari\OneDrive\Documents\full stack> c:; cd 'c:\Users\Chinnari\OneDrive\Documents\full sta
155 ck'; & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\.vscode\ex
156 tensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62729' '-' 'C:\Users\Chin
157 nari\Downloads\new.py'
158 Enter a number for multiples: 30
159 Enter an age to classify: 20
160 Enter 'n' to calculate sum up to n: 5
161
162 [For Loop] First 10 multiples of 30:
163 30 x 1 = 30
164 30 x 2 = 60
```



```
194
195 # 2. Input Collection
196 num_val = int(input("\nEnter a number for multiples: "))
197 age_val = int(input("\nEnter an age to classify: "))
198 sum_val = int(input("\nEnter 'n' to calculate sum up to n: "))
199
200 # 3. Running Multiples Logic
201 print_multiples_for(num_val)
202 print_multiples_while(num_val)
203
204 # 4. Running Age Logic
205 print(f"\n[Age Logic] Age {age_val} is classified as: {classify_age(age_val)}")
206
207 # 5. Running Summation Logic
208 print(f"\n[Sum Logic] Iterative Sum: {sum_to_n_iterative(sum_val)}")
209 print(f"[Sum Logic] Formula Sum: {sum_to_n_formula(sum_val)}")
210
211 except ValueError:
212     print("\nError: Please ensure all inputs are valid integers.")
213
214 PS C:\Users\Chinnari\OneDrive\Documents\full stack> c:; cd 'c:\Users\Chinnari\OneDrive\Documents\full sta
215 ck'; & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\.vscode\ex
216 tensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62729' '-' 'C:\Users\Chin
217 nari\Downloads\new.py'
218 30 x 10 = 300
219
220 [Age Logic] Age 20 is classified as: Adult
221
222 [Sum Logic] Iterative Sum: 15
223 [Sum Logic] Formula Sum: 15
224 PS C:\Users\Chinnari\OneDrive\Documents\full stack>
```

Task Description #5: Classes (Bank Account Class)

Scenario

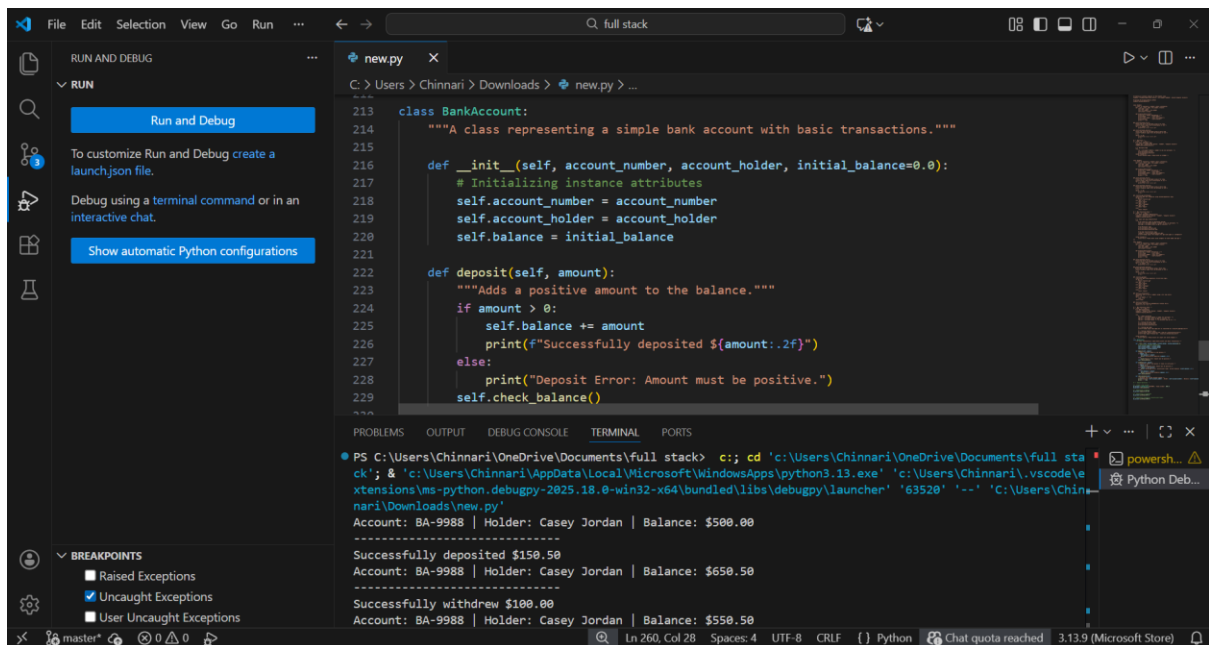
You are designing a basic banking application.

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.
- Well-commented code with a clear explanation



The screenshot shows the VS Code editor with a Python file named `new.py`. The code defines a `BankAccount` class with the following methods:

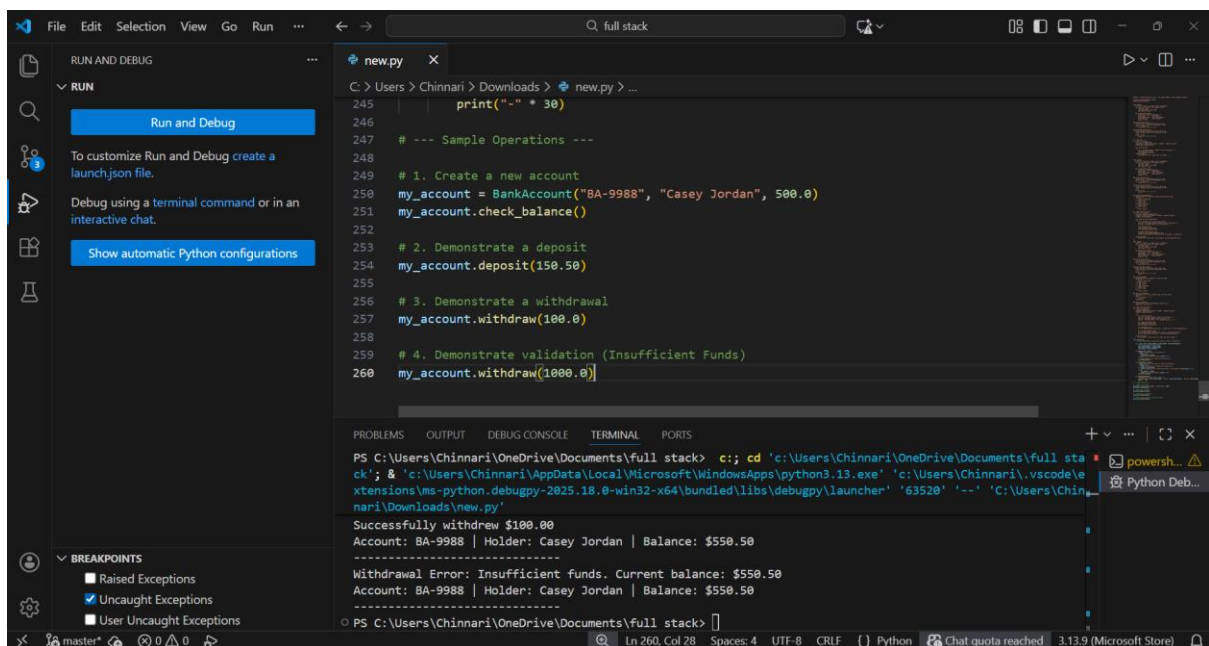
```
class BankAccount:
    """A class representing a simple bank account with basic transactions."""

    def __init__(self, account_number, account_holder, initial_balance=0.0):
        # Initializing instance attributes
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = initial_balance

    def deposit(self, amount):
        """Adds a positive amount to the balance."""
        if amount > 0:
            self.balance += amount
            print(f"Successfully deposited ${amount:.2f}")
        else:
            print("Deposit Error: Amount must be positive.")
        self.check_balance()
```

The terminal output shows the execution of the code:

```
PS C:\Users\Chinnari\OneDrive\Documents\full stack> c:\; cd 'c:\Users\Chinnari\OneDrive\Documents\full stack'; & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63520' '->' 'C:\Users\Chinnari\Downloads\new.py'
Account: BA-9988 | Holder: Casey Jordan | Balance: $500.00
Successfully deposited $150.50
Account: BA-9988 | Holder: Casey Jordan | Balance: $650.50
Successfully withdrew $100.00
Account: BA-9988 | Holder: Casey Jordan | Balance: $550.50
```



The screenshot shows the VS Code editor with the same `new.py` file. The code includes sample operations for the `BankAccount` class:

```
print("-" * 30)

# --- Sample Operations ---

# 1. Create a new account
my_account = BankAccount("BA-9988", "Casey Jordan", 500.0)
my_account.check_balance()

# 2. Demonstrate a deposit
my_account.deposit(150.50)

# 3. Demonstrate a withdrawal
my_account.withdraw(100.0)

# 4. Demonstrate validation (Insufficient Funds)
my_account.withdraw(1000.0)
```

The terminal output shows the execution of these operations:

```
PS C:\Users\Chinnari\OneDrive\Documents\full stack> c:\; cd 'c:\Users\Chinnari\OneDrive\Documents\full stack'; & 'c:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63520' '->' 'C:\Users\Chinnari\Downloads\new.py'
Successfully withdrew $100.00
Account: BA-9988 | Holder: Casey Jordan | Balance: $550.50
Withdrawal Error: Insufficient funds. Current balance: $550.50
Account: BA-9988 | Holder: Casey Jordan | Balance: $550.50
```