Srinija.G

2403A51268

**AI ASSISTED CODING – ASSIGNMENT**

**Lab 5:** AI-Based Code Auto-Completion – Classes, Loops, and Conditionals in Python usingGitHub Copilot

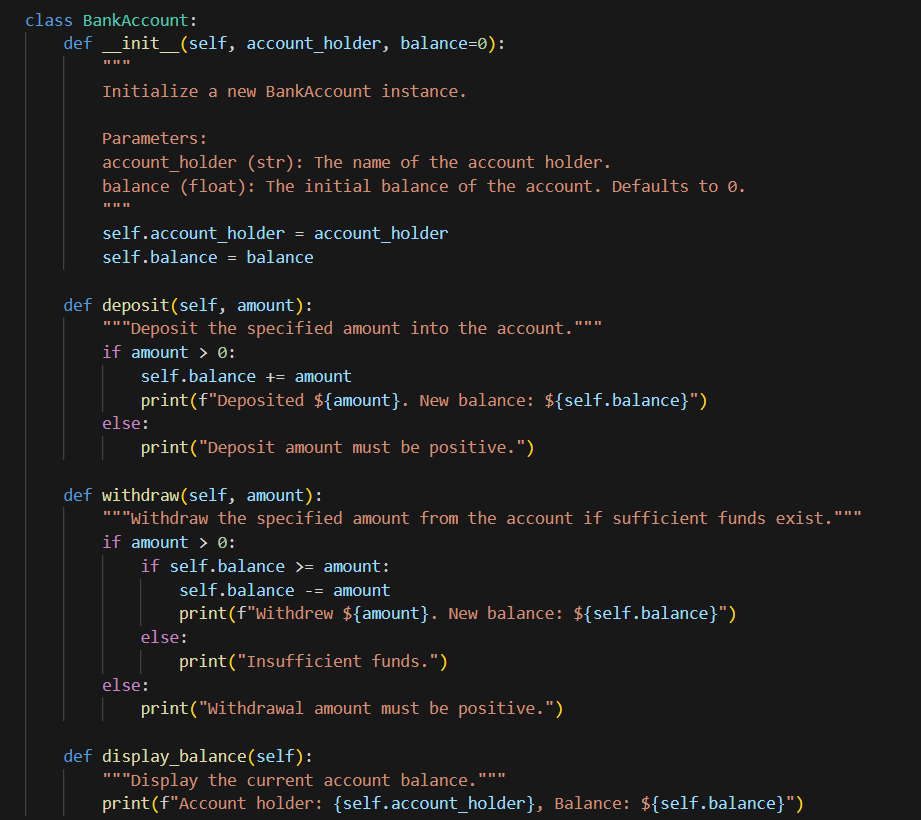
**Task 1: Auto-Complete a Python Class for Bank Account**

Prompt –

Write a class definition comment and construct a class called BankAccount with

account\_holder and balance attributes. The class must include methods to deposit, withdraw, and display balance.

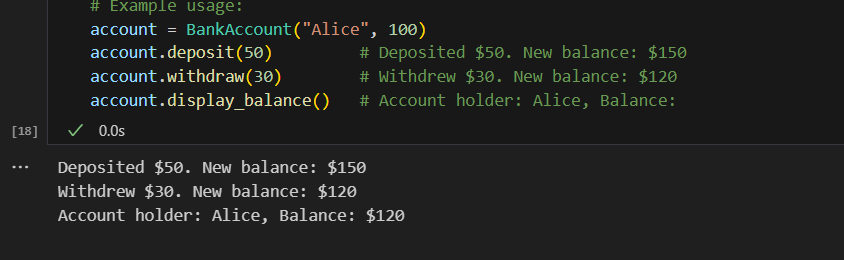
Code –

****

Code Explanation –

1. Class Definition: The BankAccount class is defined to represent a simple bank account with basic operations.
2. Constructor (\_\_init\_\_): Initializes the account with the holder's name and an optional starting balance (default is 0).
3. Deposit Method: Adds a positive amount to the balance and prints the new balance. If the amount is not positive, it prints an error message.
4. Withdraw Method: Subtracts a positive amount from the balance if there are sufficient funds and prints the new balance. If not enough funds or the amount is not positive, it prints an error message.
5. Display Balance Method: Prints the account holder's name and the current balance.
6. Example Usage: Creates an account for "Alice" with $100, deposits $50, withdraws $30, and displays the final balance.

Output –

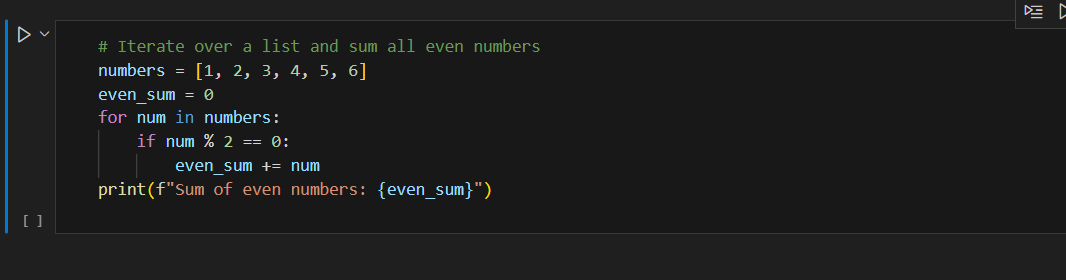


**Task 2: Auto-Complete a For Loop to Sum Even Numbers in a List**

Prompt –

write a comment and the initial line of a loop to iterate over a list. complete the logic to sum all even numbers in the list.

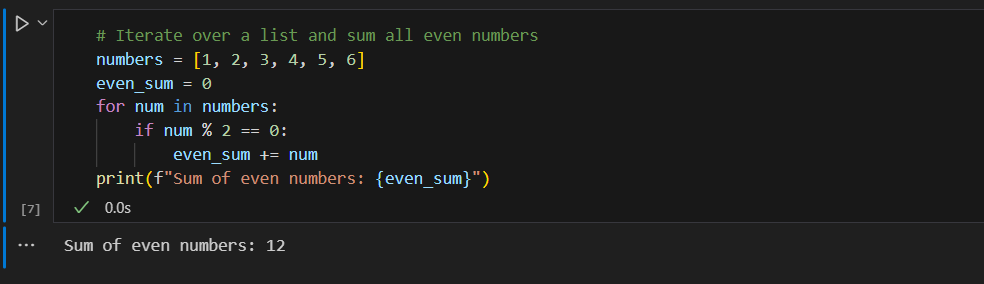
Code –

****

Code Explanation –

1. numbers = […] → creates a list of numbers from 1 to 10.
2. even\_sum = 0 → variable to store the total of even numbers.
3. for num in numbers: → loop goes through each number in the list.
4. if num % 2 == 0: → checks if the number is even (remainder 0 when divided by 2).
5. even\_sum += num → adds the even number to the total sum.
6. After the loop finishes, print(…) displays the final sum.

Output –



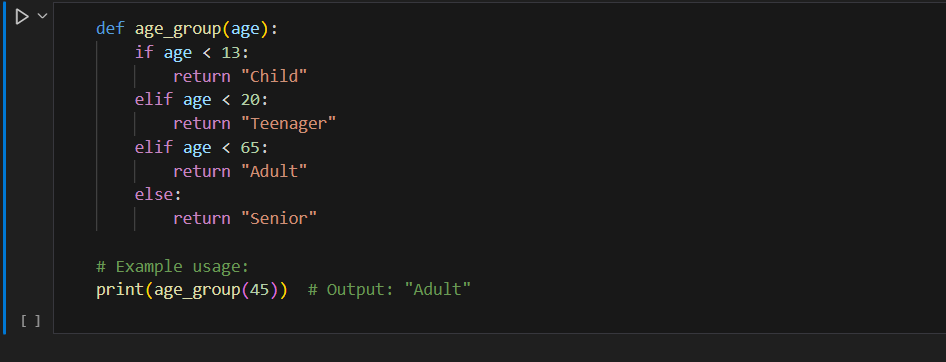
**Task 3: Auto-Complete Conditional Logic to Check Age Group**

Prompt –

Start a function that takes age as input and returns whether the person is a child, teenager,

adult, or senior using if-elif-else.

Code –



Code explanation –

1. def age\_group(age): → defines a function that takes age as input.
2. if age < 13: → if the age is less than 13, return “Child”.
3. elif age < 20: → if the age is less than 20 (but ≥ 13), return “Teenager”.
4. elif age < 60: → if the age is less than 60 (but ≥ 20), return “Adult”.
5. else: → if age is 60 or more, return “Senior”.
6. print(age\_group(45)) → calls the function with 45, which satisfies the “Adult” condition.

Output –

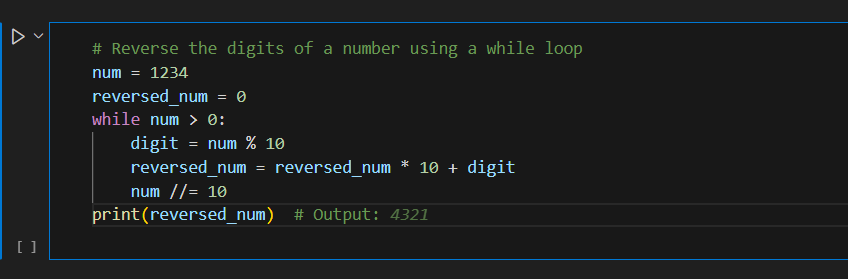


**Task 4: Auto-Complete a While Loop to Reverse Digits of a Number**

Prompt –

Write a comment and start a while loop to reverse the digits of a number. complete the loop logic.

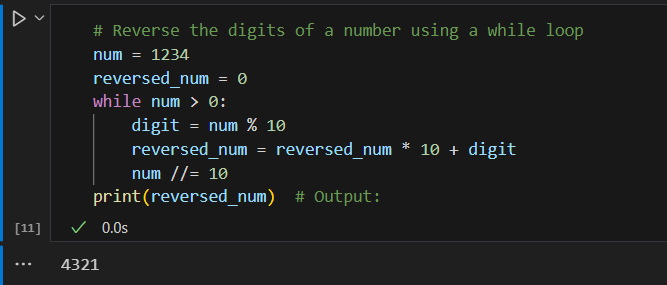
Code –



Code explanation –

1. num = 1234 → the number we want to reverse.
2. reversed\_num = 0 → variable to store the reversed number.
3. while num > 0: → loop continues until all digits are processed.
4. digit = num % 10 → extracts the last digit of num.
5. reversed\_num = reversed\_num \* 10 + digit → shifts existing digits left and adds the new digit.
6. num //= 10 → removes the last digit from num (integer division).
7. After loop ends, print(…) shows the reversed number.

Output –

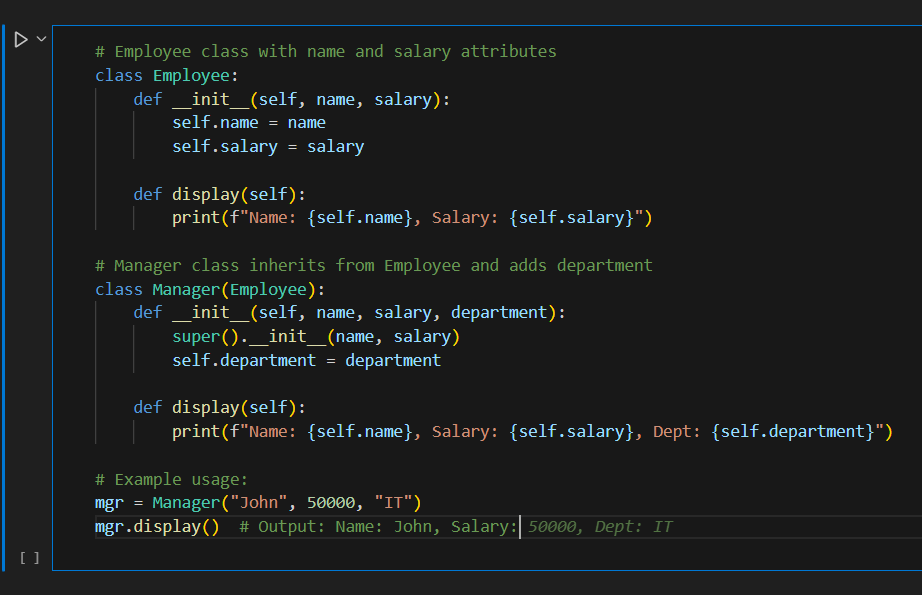


**Task 5: Auto-Complete Class with Inheritance (Employee → Manager)**

Prompt –

Begin a class Employee with attributes name and salary. Then, start a derived class Manager  
that inherits from Employee and adds department. complete the methods  
and constructor chaining.

Code –



Code explanation –

1. class Employee: → base class with attributes name and salary.
2. \_\_init\_\_ in Employee → constructor initializes name and salary.
3. display() in Employee → prints employee details.
4. class Manager(Employee): → derived class that inherits from Employee.
5. \_\_init\_\_ in Manager → uses super().\_\_init\_\_ to call the parent constructor, then adds department.
6. display() in Manager → overrides parent’s display method to also show department.
7. manager = Manager(“John”, 50000, “IT”) → creates a Manager object.

Output –

