ASSIGNMENT:4.4

M.KEERTHANA

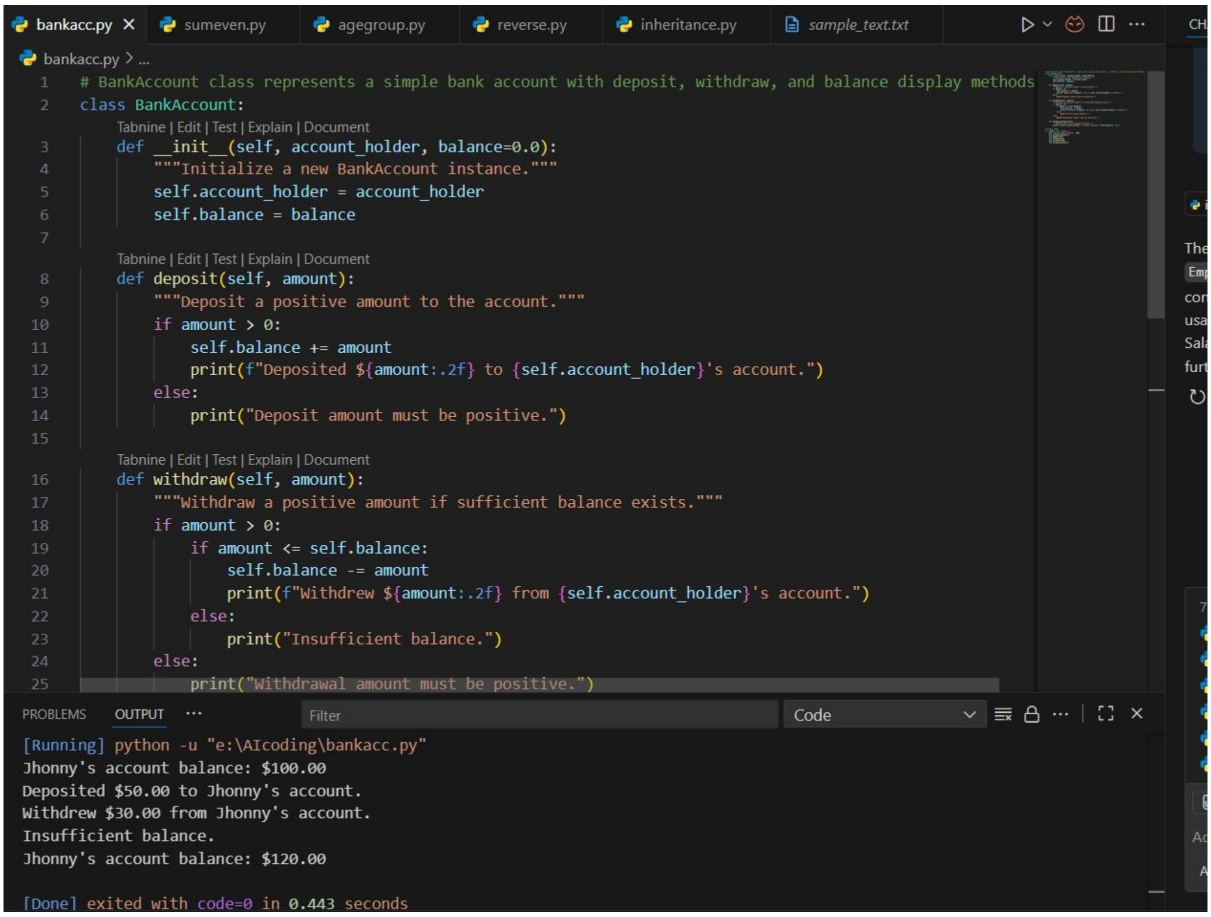
2403A51259

BATCH 11 CSE-GEN

AI ASSISTED CODING

TASK1:

Auto-Complete a Python Class for Bank Account• Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance.



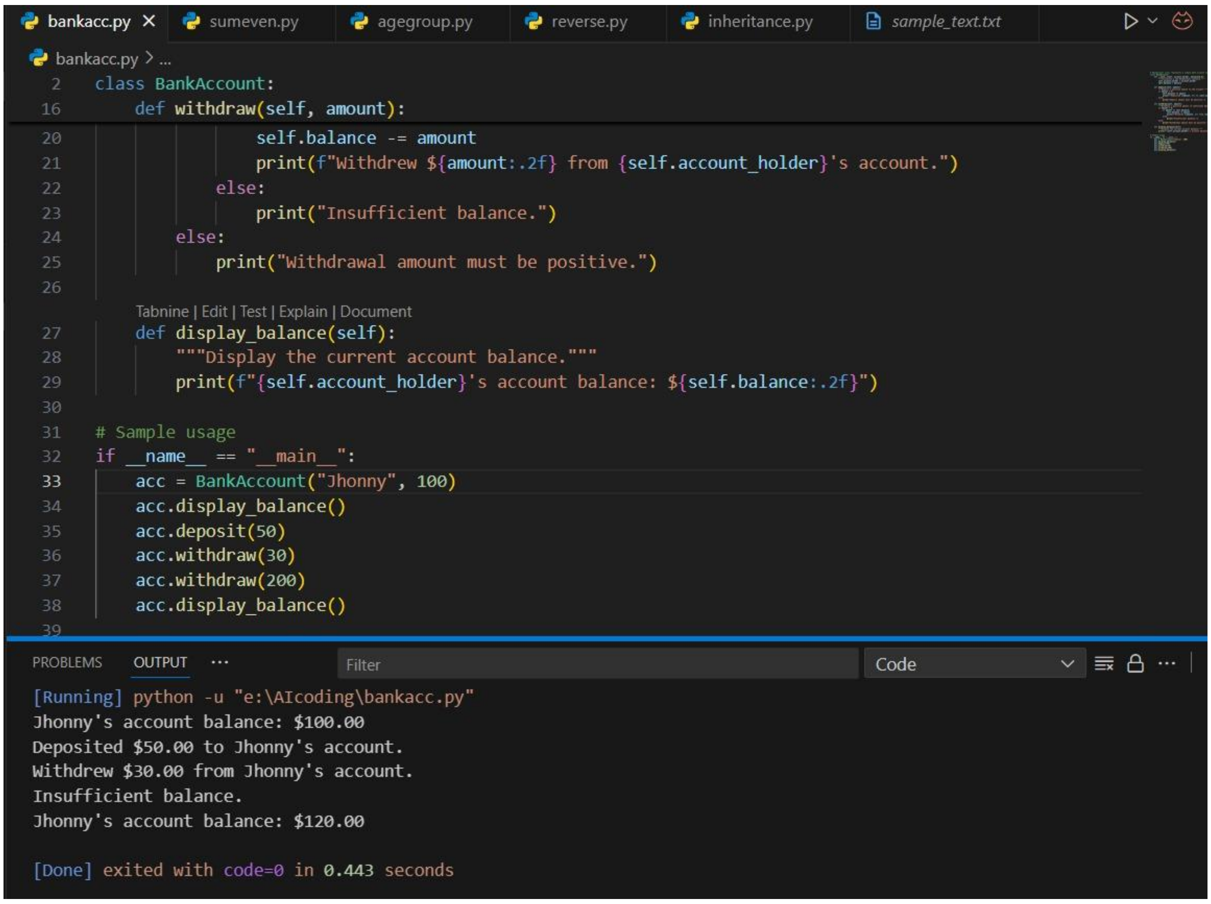
```python
# BankAccount class represents a simple bank account with deposit, withdraw, and balance display methods
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        """Initialize a new BankAccount instance."""
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        """Deposit a positive amount to the account."""
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount:.2f} to {self.account_holder}'s account.")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        """Withdraw a positive amount if sufficient balance exists."""
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew ${amount:.2f} from {self.account_holder}'s account.")
            else:
                print("Insufficient balance.")
        else:
            print("Withdrawal amount must be positive.")
```

```
[Running] python -u "e:\AIcoding\bankacc.py"
Jhonny's account balance: $100.00
Deposited $50.00 to Jhonny's account.
Withdrew $30.00 from Jhonny's account.
Insufficient balance.
Jhonny's account balance: $120.00

[Done] exited with code=0 in 0.443 seconds
```



```python
class BankAccount:
    def withdraw(self, amount):
                self.balance -= amount
                print(f"Withdrew ${amount:.2f} from {self.account_holder}'s account.")
            else:
                print("Insufficient balance.")
        else:
            print("Withdrawal amount must be positive.")

    def display_balance(self):
        """Display the current account balance."""
        print(f"{self.account_holder}'s account balance: ${self.balance:.2f}")

# Sample usage
if __name__ == "__main__":
    acc = BankAccount("Jhonny", 100)
    acc.display_balance()
    acc.deposit(50)
    acc.withdraw(30)
    acc.withdraw(200)
    acc.display_balance()
```

```
[Running] python -u "e:\AIcoding\bankacc.py"
Jhonny's account balance: $100.00
Deposited $50.00 to Jhonny's account.
Withdrew $30.00 from Jhonny's account.
Insufficient balance.
Jhonny's account balance: $120.00

[Done] exited with code=0 in 0.443 seconds
```

EXPLANATION:

This code defines a **BankAccount** class to manage basic bank operations:

- __init__: Initializes the account with holder name, balance, and a list for transaction history.
- deposit: Adds money to the account and logs the transaction.
- withdraw: Deducts money if there's enough balance and logs the transaction.
- display_balance: Shows the current balance.
- show_transactions: Prints all transaction history with timestamps.
- __str__: Returns a string summary of the account.

1. __init__ and __str__ are misspelled as _init_ and _str_ (should be **double underscores**: __init__, __str__)
2. if _name_ == "_main_" should be if __name__ == "__main__"

Task 2:

Auto-Complete a For Loop to Sum Even Numbers in a List • Write a comment and the initial line of a loop to iterate over a list. Allow GitHub Copilot to complete the logic to sum all even numbers in the list.

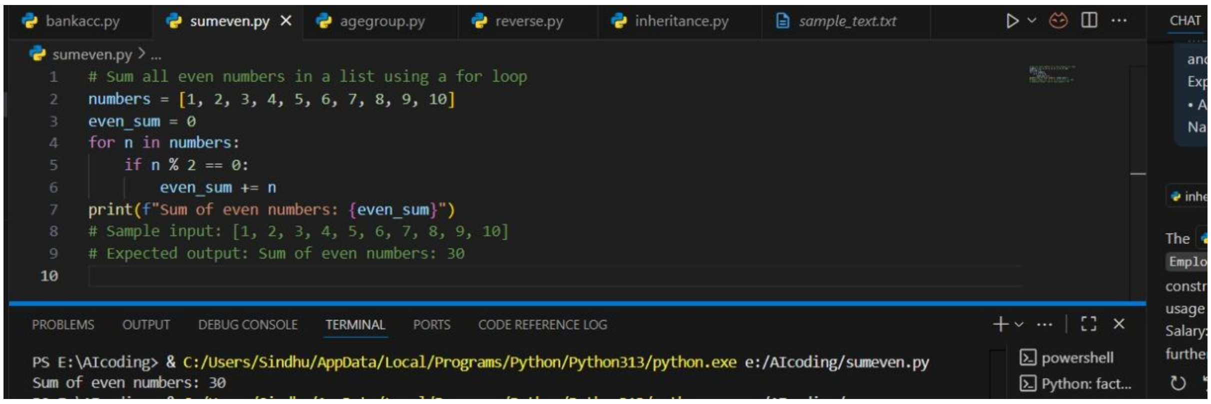
Expected Output #2

• Code that:

• Iterates over a list

• Checks if the number is even using % 2 == 0

• Accumulates the sum



```python
# Sum all even numbers in a list using a for loop
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum = 0
for n in numbers:
    if n % 2 == 0:
        even_sum += n
print(f"Sum of even numbers: {even_sum}")
# Sample input: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# Expected output: Sum of even numbers: 30
```

```
PS E:\AIcoding> & C:/Users/Sindhu/AppData/Local/Programs/Python/Python313/python.exe e:/AIcoding/sumeven.py
Sum of even numbers: 30
```

EXPLANATION:

The code sums all **even numbers** in a list using a for loop.

- It checks if each number is even (n % 2 == 0)
- If true, it adds the number to even_sum
- Finally, it prints the total, which is **30** for the given list.

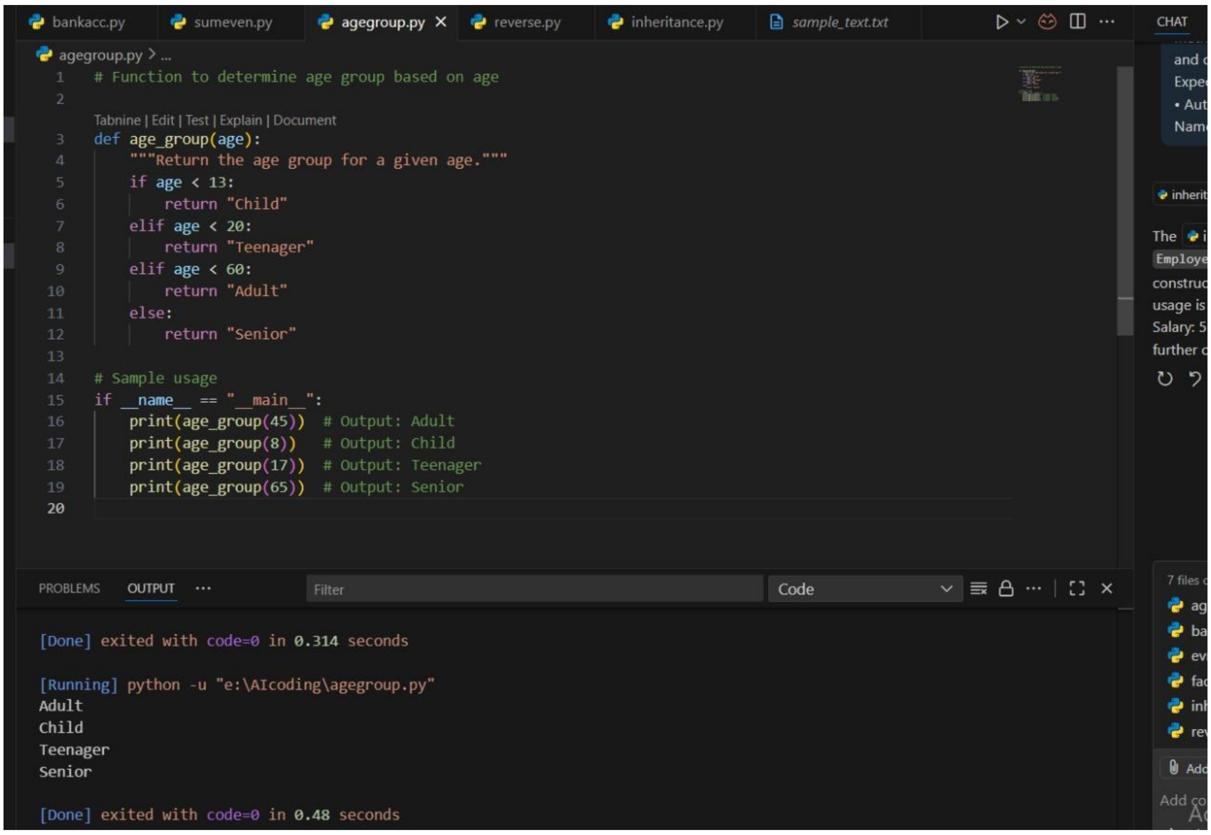
TASK3:

Auto-Complete Conditional Logic to Check Age Group Start a function that takes age as input and returns whether the person is a child, teenager, adult, or senior using if-elif-else. Use Copilot to complete the conditionals.

Expected Output #3

• Function like:

Output for age_group(45) → "Adult"



```python
# Function to determine age group based on age

def age_group(age):
    """Return the age group for a given age."""
    if age < 13:
        return "Child"
    elif age < 20:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"

# Sample usage
if __name__ == "__main__":
    print(age_group(45))  # Output: Adult
    print(age_group(8))   # Output: Child
    print(age_group(17))  # Output: Teenager
    print(age_group(65))  # Output: Senior
```

```
[Done] exited with code=0 in 0.314 seconds

[Running] python -u "e:\AIcoding\agegroup.py"
Adult
Child
Teenager
Senior

[Done] exited with code=0 in 0.48 seconds
```

EXPLANATION:

The code defines a function age_group(age) that returns the age group based on the given age:

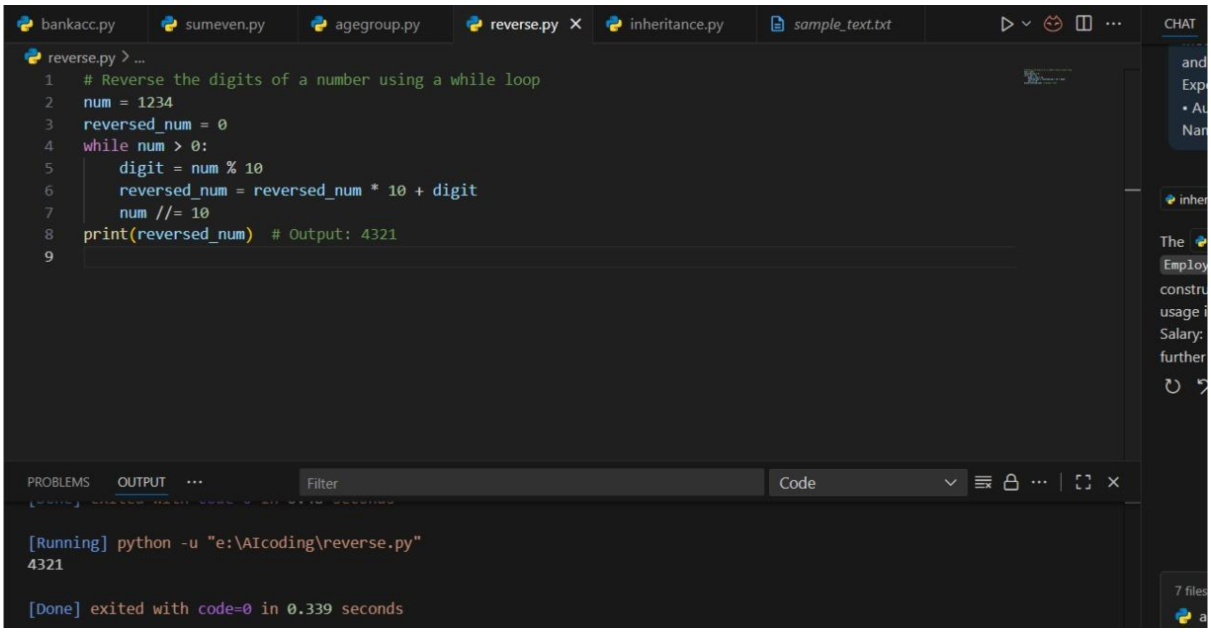
- **< 13** → "Child"
- **13–19** → "Teenager"
- **20–59** → "Adult"
- **60 and above** → "Senior"

TASK4:

Auto-Complete a While Loop to Reverse Digits of a Number • Write a comment and start a while loop to reverse the digits of a number. Let Copilot complete the loop logic.

Expected Output #4

• Functional loop: Output: 4321



```python
# Reverse the digits of a number using a while loop
num = 1234
reversed_num = 0
while num > 0:
    digit = num % 10
    reversed_num = reversed_num * 10 + digit
    num //= 10
print(reversed_num)  # Output: 4321
```

```
[Running] python -u "e:\AIcoding\reverse.py"
4321

[Done] exited with code=0 in 0.339 seconds
```

EXPLANATION:

The code reverses the digits of the number 1234 using a while loop.

- It extracts the last digit using num % 10.
- Builds the reversed number by shifting existing digits left (* 10) and adding the new digit.
- Removes the last digit from num using integer division (// 10).
- Finally, it prints the reversed number, which is 4321

Task 5:

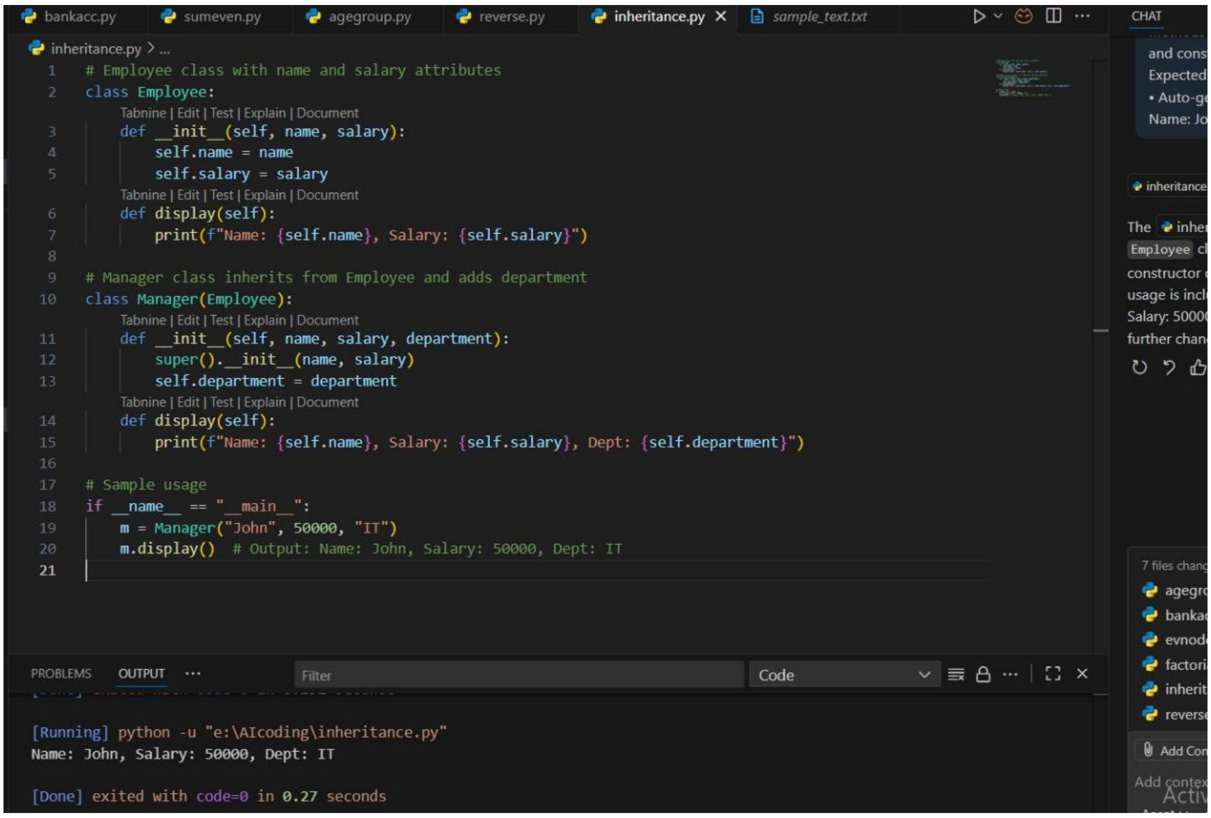
Auto-Complete Class with Inheritance (Employee → Manager)

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

Expected Output #5

• Auto-generated code like:

Name: John, Salary: 50000, Dept: IT



```python
# Employee class with name and salary attributes
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def display(self):
        print(f"Name: {self.name}, Salary: {self.salary}")

# Manager class inherits from Employee and adds department
class Manager(Employee):
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department
    def display(self):
        print(f"Name: {self.name}, Salary: {self.salary}, Dept: {self.department}")

# Sample usage
if __name__ == "__main__":
    m = Manager("John", 50000, "IT")
    m.display()  # Output: Name: John, Salary: 50000, Dept: IT
```

```
[Running] python -u "e:\AIcoding\inheritance.py"
Name: John, Salary: 50000, Dept: IT

[Done] exited with code=0 in 0.27 seconds
```

EXPLANATION:

The code shows **inheritance** in Python:

- Employee class has name and salary.
- Manager class inherits from Employee and adds department.
- Both classes have a display() method, with Manager overriding it.
- The output shows details of a Manager object: name, salary, and department.