ระบบจัดการสินค้าคงคลังร้านค้า "ตลาดสดผลไม้ป้าแดง"

เรื่องราว:

ป้าแดงมีร้านขายผลไม้เล็กๆ ในตลาดสด ชื่อ "ตลาดสดผลไม้ป้าแดง" ป้าแดงอยากจะพัฒนาระบบง่ายๆ ขึ้นมาเพื่อช่วยจัดการสินค้าคงคลังในร้านของป้าแดง เพื่อให้รู้ว่าผลไม้ชนิดไหนมีเท่าไหร่ และราคาเท่าไหร่ ป้าแดงต้องการความสามารถดังนี้:

1. **ดูและจัดการรายการผลไม้:** แสดงชื่อผลไม้, จำนวน, และราคา และสามารถเพิ่มผลไม้ใหม่เข้าสต็อกได้ด้วย
2. **ซื้อผลไม้:** ลูกค้าสามารถระบุว่าจะซื้อผลไม้อะไรบ้าง และโปรแกรมจะคำนวณราคารวมพร้อมทั้งอัปเดตจำนวนในสต็อก

โจทย์ที่คุณต้องทำ:

คุณจะต้องสร้างโปรแกรม Python ที่มีโครงสร้างดังนี้:

- ใช้ **dictionary** ในการเก็บข้อมูลสินค้าคงคลัง โดย **key** คือชื่อผลไม้ (string) และ **value** เป็น **list** ที่มี 2 สมาชิกคือ [จำนวน (หน่วย), ราคาต่อหน่วย]
- ใช้ **2 ฟังก์ชันหลัก** ดังนี้:
    1. manage_and_display_inventory(inventory, fruit_name=None, quantity=None, price=None)
    2. process_customer_purchase(inventory, shopping_list)

  รายละเอียดอยู่หน้าถัดไป
- ใช้ **for loop** ในการวนลูปแสดงรายการผลไม้ หรือในการคำนวณราคารวม
- ใช้ **if-else** ในการจัดการเงื่อนไขต่างๆ เช่น ผลไม้มีไม่พอให้ซื้อ, ผลไม้ยังไม่มีในสต็อก, หรือการตัดสินใจว่าจะแสดงผลหรือเพิ่มผลไม้

ข้อมูลเริ่มต้น:

```
inventory = { "apple": [10, 25.00],  # [จำนวน, ราคาต่อหน่วย]
              "banana": [20, 15.00],
              "orange": [15, 30.00]}
```

ฟังก์ชันที่ต้องการ:

1. **manage_and_display_inventory(inventory, fruit_name=None, quantity=None, price=None)**
    - รับ **inventory** (dictionary)
    - รับ **fruit_name** (string, optional): ชื่อผลไม้ที่จะเพิ่ม (ถ้าไม่ระบุ แสดงว่าต้องการแค่แสดงผล)
    - รับ **quantity** (int, optional): จำนวน (ถ้า fruit_name ถูกระบุคือจำนวนเริ่มต้น)
    - รับ **price** (float, optional): ราคาต่อหน่วย (ถ้า fruit_name ถูกระบุ)
    - **การทำงาน:**
        - **ถ้า fruit_name เป็น None** (หรือไม่ได้ใส่มา): ให้ใช้ **for loop** วนแสดงรายการผลไม้ทั้งหมดใน inventory ในรูปแบบที่อ่านง่าย
            - ถ้า inventory ว่างเปล่า ให้พิมพ์ว่า "ไม่มีผลไม้ในสต็อก!"
        - **ถ้า fruit_name ถูกระบุ** (พร้อม quantity และ price):
            - ใช้ **if-else** ตรวจสอบว่า fruit_name ยังไม่มีใน inventory และ quantity, price เป็นค่าบวกหรือไม่
            - ถ้าใช่: เพิ่มผลไม้ใหม่เข้าไปใน inventory และพิมพ์ยืนยัน
            - ถ้า fruit_name มีอยู่แล้ว หรือ quantity/price ไม่ถูกต้อง: พิมพ์ข้อความเตือน

2. **process_customer_purchase(inventory, shopping_list)**
    - รับ **inventory** (dictionary)
    - รับ **shopping_list** (dictionary): มี key เป็นชื่อผลไม้ และ value เป็นจำนวนที่ต้องการซื้อ
    - **การทำงาน:**
        - คำนวณราคารวมทั้งหมด (เริ่มต้นที่ 0)
        - ใช้ **for loop** วนไปทีละรายการใน shopping_list:
            - ใช้ **if-else** ตรวจสอบว่าผลไม้ที่ต้องการซื้อมีใน inventory และมีจำนวนเพียงพอหรือไม่
            - ถ้ามีและพอ:
                - คำนวณราคารวมสำหรับผลไม้นั้น
                - บวกเข้าราคารวมทั้งหมด
                - **ลดจำนวนผลไม้ใน inventory** ตามจำนวนที่ลูกค้าซื้อ
                - พิมพ์ยืนยันการซื้อและจำนวนคงเหลือ
            - ถ้าไม่มีหรือมีไม่พอ: พิมพ์ข้อความเตือนและไม่รวมราคานี้
        - หลังจากวนลูปครบแล้ว ให้พิมพ์ราคารวมทั้งหมดที่ลูกค้าต้องจ่าย
        - คืนค่าราคารวมทั้งหมด (float)

TEST CASE

```
print("--- Test Case 1: Display Initial Inventory ---")
manage_and_display_inventory(inventory) # แสดงผลอย่างเดียว

print("\n--- Test Case 2: Add New Fruit ---")
manage_and_display_inventory(inventory, "grape", 30, 45.00) # เพิ่มองุ่น
manage_and_display_inventory(inventory, "apple", 5, 20.00) # ลองเพิ่ม apple ที่มีอยู่แล้ว
manage_and_display_inventory(inventory, "kiwi", -2, 10.00) # ลองเพิ่ม kiwi ด้วยจำนวนที่ไม่ถูกต้อง
manage_and_display_inventory(inventory) # แสดงผลหลังการเพิ่ม

print("\n--- Test Case 3: Process Customer Purchase ---")
customer_shopping_list1 = {
    "banana": 3,
    "grape": 10,
    "orange": 2
}
total_cost1 = process_customer_purchase(inventory, customer_shopping_list1)
print(f"ลูกค้าต้องจ่ายทั้งหมด: {total_cost1:.2f} บาท")
manage_and_display_inventory(inventory) # แสดงผลหลังการซื้อ (จำนวนควรลดลง)

print("\n--- Test Case 4: Process Purchase with Insufficient/Missing Items ---")
customer_shopping_list2 = {
    "apple": 12, # มี 10 ลูก ไม่พอ
    "grape": 5,
    "mango": 2 # ไม่มีในสต็อก
}
total_cost2 = process_customer_purchase(inventory, customer_shopping_list2)
print(f"ลูกค้าต้องจ่ายทั้งหมด: {total_cost2:.2f} บาท")
manage_and_display_inventory(inventory) # แสดงผลหลังการซื้อ (apple, mango ไม่ควรเปลี่ยน, grape ลด)

print("\n--- Test Case 5: Empty Inventory Display ---")
empty_inventory = {}
manage_and_display_inventory(empty_inventory) # แสดงผลสำหรับสต็อกว่าง
```

SOLUTION

```
def manage_and_display_inventory(inventory, fruit_name=None, quantity=None, price=None):
    """
    แสดงรายการผลไม้ทั้งหมด หรือเพิ่มผลไม้ใหม่เข้าสต็อก
    """
    if fruit_name is None: # ถ้าไม่ได้ระบุชื่อผลไม้ แสดงว่าต้องการแค่แสดงผล
        if not inventory:
            print("ไม่มีผลไม้ในสต็อก!")
            return

        print("--- รายการสินค้าคงคลัง ---")
        for fruit, details in inventory.items():
            current_quantity, current_price = details
            print(f"ผลไม้: {fruit.capitalize()}, จำนวน: {current_quantity}, ราคา: {current_price:.2f} บาท")
        print("--------------------------")
    else: # ถ้ามีชื่อผลไม้ แสดงว่าต้องการเพิ่ม
        fruit_name_lower = fruit_name.lower()
        if fruit_name_lower in inventory:
            print(f"ผลไม้ '{fruit_name.capitalize()}' มีอยู่แล้ว! หากต้องการอัปเดต ต้องทำผ่านฟังก์ชันซื้อหรือจัดการแบบ
เฉพาะเจาะจง.")
            return

        if quantity is None or price is None or quantity <= 0 or price <= 0:
            print(f"ไม่สามารถเพิ่มผลไม้ '{fruit_name.capitalize()}' ได้: จำนวนและราคาต้องเป็นค่าบวก")
            return

        inventory[fruit_name_lower] = [quantity, price]
        print(f"เพิ่ม '{fruit_name.capitalize()}' จำนวน {quantity} ในราคา {price:.2f} บาท เรียบร้อยแล้ว.")
```

```
def process_customer_purchase(inventory, shopping_list):
    """
    ประมวลผลการซื้อของลูกค้า คำนวณราคารวม และอัปเดตสต็อก
    """
    total_cost = 0.0
    print("\n--- กำลังประมวลผลการซื้อของลูกค้า ---")
    for fruit, requested_quantity in shopping_list.items():
        fruit_lower = fruit.lower()

        if fruit_lower not in inventory:
            print(f"  แจ้งเตือน: ผลไม้ '{fruit.capitalize()}' ไม่มีในสต็อก ไม่สามารถซื้อได้.")
            continue

        available_quantity, price_per_unit = inventory[fruit_lower]

        if requested_quantity <= 0:
            print(f"  แจ้งเตือน: จำนวน '{fruit.capitalize()}' ที่ต้องการซื้อไม่ถูกต้อง (ต้องมากกว่า 0).")
            continue

        if requested_quantity > available_quantity:
            print(f"  แจ้งเตือน: ผลไม้ '{fruit.capitalize()}' มีไม่พอในสต็อก. มีอยู่ {available_quantity} ลูก แต่ต้องการ
{requested_quantity} ลูก.")
            continue # ไม่รวมราคาสินค้านั้นๆ เข้าไป

        # ถ้ามีพอและถูกต้อง
        cost_for_fruit = requested_quantity * price_per_unit
        total_cost += cost_for_fruit
        inventory[fruit_lower][0] -= requested_quantity # ลดจำนวนในสต็อก
        print(f"  ซื้อ {fruit.capitalize()} จำนวน {requested_quantity} ลูก ราคา {cost_for_fruit:.2f} บาท. เหลือใน
สต็อก: {inventory[fruit_lower][0]} ลูก")
    print("-----------------------------------")
    return total_cost
```