1.Create a dictionary with student names as keys and their marks as values. Allow the user to input a name and display the student's marks. If the student doesn't exist, show an appropriate message.

```
student_marks = {}  # Initialize an empty dictionary
while True:
    name = input("Enter student name (or 'done' to finish): ")
    if name == 'done':
        break
    marks = int(input(f"Enter marks for {name}: "))
    student_marks[name] = marks

# ... (Assuming student_marks dictionary is already populated) ...

# Get the names of students to display marks for
students_to_check = input("Enter names of students to check marks,
separated by commas: ").split(',')
students_to_check = [name.strip() for name in students_to_check] #
Removes leading/trailing spaces from names

# Display marks for the specified students
for student in students_to_check:
    if student in student_marks:
        print(f"Marks for {student}: {student_marks[student]}")
    else:
        print(f"Student {student} not found in the records.")
```

```
Enter student name (or 'done' to finish): Anusha
Enter marks for Anusha: 67
Enter student name (or 'done' to finish): Bob
Enter marks for Bob: 98
Enter student name (or 'done' to finish): Charlie
Enter marks for Charlie: 98
Enter student name (or 'done' to finish): done
Enter names of students to check marks, separated by commas:
Bob,Samuel
Marks for Bob: 98
Student Samuel not found in the records.
```

2.Given a list of names, group them into a dictionary where the key is the first letter of each name and the value is a list of names that start with that letter.

```
def group_names_by_first_letter(names):
    """Groups a list of names by their first letter.

    Args:
        names: A list of names.

    Returns:
```

```
        A dictionary where keys are the first letter of names,
        and values are lists of names starting with that letter.
    """
    name_groups = {}
    for name in names:
        first_letter = name[0].upper()  # Get the first letter and
convert to uppercase
        if first_letter not in name_groups:
            name_groups[first_letter] = []  # Create a new list if the
letter is not a key
        name_groups[first_letter].append(name)  # Add the name to the
list for that letter
    return name_groups

# Example usage
names = ["Alice", "Bob", "Charlie", "David", "Eve", "Jona", "Albert"]
grouped_names = group_names_by_first_letter(names)
print(grouped_names)
```

```
{'A': ['Alice', 'Albert'], 'B': ['Bob'], 'C': ['Charlie'], 'D':
['David'], 'E': ['Eve'], 'J': ['Jona']}
```

3.Build a simple inventory system where each item and its quantity is stored in a dictionary. Ask the user to enter an item name and quantity to sell. Update the dictionary and show the remaining stock or a message if there's not enough.

```
# Initialize the inventory dictionary
inventory = {
    "apple": 10,
    "banana": 15,
    "orange": 20
}

# Function to update inventory and sell items
def sell_item(item, quantity):
    """Updates the inventory after selling an item.

    Args:
        item: The name of the item to sell.
        quantity: The quantity of the item to sell.
    """
    if item in inventory and inventory[item] >= quantity:
        inventory[item] -= quantity
        print(f"Sold {quantity} {item}(s). Remaining stock:
{inventory[item]}")
    elif item not in inventory:
        print(f"{item} not found in inventory.")
    else:
        print(f"Not enough {item}(s) in stock. Current stock:
```

```
{inventory[item]}")

# Example usage
while True:
    item = input("Enter item name (or 'done' to finish): ")
    if item == 'done':
        break
    quantity = int(input(f"Enter quantity of {item} to sell: "))
    sell_item(item, quantity)

print("Final inventory:", inventory)
```

```
Enter item name (or 'done' to finish): apple
Enter quantity of apple to sell: 3
Sold 3 apple(s). Remaining stock: 7
Enter item name (or 'done' to finish): orange
Enter quantity of orange to sell: 5
Sold 5 orange(s). Remaining stock: 15
Enter item name (or 'done' to finish): mango
Enter quantity of mango to sell: 2
mango not found in inventory.
Enter item name (or 'done' to finish): banana
Enter quantity of banana to sell: 20
Not enough banana(s) in stock. Current stock: 15
Enter item name (or 'done' to finish): done
Final inventory: {'apple': 7, 'banana': 15, 'orange': 15}
```

4.Create a contact book using a dictionary where the name is the key and phone number is the value. Allow the user to input a name and phone number. If the name exists, update the number; otherwise, insert a new contact.

```
# Initialize the contact book dictionary
contact_book = {}

# Function to add or update contacts
def manage_contacts(name, phone_number):
    """Adds or updates contacts in the contact book.

    Args:
        name: The name of the contact.
        phone_number: The phone number of the contact.
    """
    contact_book[name] = phone_number  # Adds or updates the contact
    print(f"Contact {name} updated/added successfully.")

# Example usage
while True:
    name = input("Enter contact name (or 'done' to finish): ")
    if name == 'done':
        break
```

```
    phone_number = input(f"Enter phone number for {name}: ")
    manage_contacts(name, phone_number)

print("Contact book:", contact_book)
```

```
Enter contact name (or 'done' to finish): Anusha
Enter phone number for Anusha: 9878778808
Contact Anusha updated/added successfully.
Enter contact name (or 'done' to finish): Tera
Enter phone number for Tera: 6748234902
Contact Tera updated/added successfully.
Enter contact name (or 'done' to finish): done
Contact book: {'Anusha': '9878778808', 'Tera': '6748234902'}
```