Assignment # 3

[Find errors and complete the steps]

1. Implement a to-do list application where users can add, remove, and view tasks

1. Initialize an Empty List

```
tasks = []  # This list will store the tasks added by the user.
```

2. Define a Function to Add Tasks

```
def add_task(task):
    tasks.append(task)
```

3. Define a Function to Remove Tasks

```
def remove_task(task):
    if task in_ tasks:
        tasks.remove(task)
```

4. Define a Function to View Tasks

```
def view_tasks():
    for task in tasks
        print(task)
```

5. Example Usage

```
add_task("Finish homework")
add_task("Buy groceries")
view_tasks()
remove_task("Buy groceries")
```

[Example Output]

```
Finish homework
Buy groceries
Finish homework
```

Find Maximum and Minimum Values in a List

1. Define the Function

```
def find_max_min(lst):
```

2. Initialize max_val and min_val

```
    max_val = lst[0]
    min_val = lst[0]
```

3. Iterate Through the List

```
    for num in lst:
```

4. Compare Each Number to Find Maximum and Minimum

```
        if num > max_val:
            max_val = num
        if num < min_val
            min_val = num
```

5. Return the Maximum and Minimum Values

```
    return max_val, min_val
```

6. Define a List and Call the Function

```
numbers = [3, 9, 2, 8, 1]
max_val, min_val = find_max_min(numbers)
print(f"Max: {max_val}, Min: {min_val}")
```

[Example Output]

```
Max: 9, Min: 1
```

Create a Simple Phonebook Using Dictionaries

1. Initialize an Empty Dictionary

```
phonebook = {}
```

2. Define a Function to Add Contacts

```
def add_contact(name, number):
    phonebook[name] = number
```

3. Define a Function to Retrieve a Phone Number

```
def get_number(name):
    return phonebook.get(name, "Contact not found")
```

4. Example Usage

```
add_contact("Alice", "1234567890")
add_contact("Bob", "9876543210")
print(get_number("Alice"))
```

[Example Output]

```
1234567890
```

Inventory System (Using Dictionaries)

1. Initialize an Empty Dictionary

```
inventory = {}
```

2. Define a Function to Add Items

```
def add_item(item, quantity):
```

3. Check if the Item Exists in inventory and If the Item Does Not Exist, Add It

```
      if item in inventory:
         inventory[item] += quantity
      else:
         inventory[item] = quantity
```

4. Define a Function to Remove Items

```
def remove_item(item, quantity):
```

5. Check If the Item Exists and Has Enough Quantity and Decrease the Quantity

```
    if item in inventory and inventory[item] >= quantity:
        inventory[item] -= quantity
```

6. If the Quantity Becomes Zero, Remove the Item

```
        if inventory[item] == 0:
            del inventory[item]
    else
      print(f"Insufficient quantity of {item}.")
```

7. Define a Function to View Inventory

```
def view_inventory():
```

8. Loop Through the Inventory

```
    for item, quantity in inventory.items():
        print(f"{item}: {quantity}")
```

9. Example Usage

```
add_item("Apple", 10)
add_item("Banana", 5)
remove_item("Apple", 3)
view_inventory()
```

[Example Output]

```
Apple: 7
Banana: 5
```

ATM System Simulation with Global Variables

```
def deposit(balance, amount):
    balance += amount
    print(f"Deposited: {amount}. New balance: {balance}")
    return balance  # Return the updated balance

def withdraw(balance, amount)
    if amount <= balance:
        balance -= amount
        print(f"Withdrew: {amount}. New balance: {balance}")
    else
        print("Insufficient funds")
    return balance  # Return the updated balance

def check_balance(balance):
    print("Current balance: {balance}")

# Example usage
balance = 1000  # Initial balance

balance = deposit(balance, 500)  # Deposit money
balance = withdraw(balance, 200)  # Withdraw money
check_balance(balance)  # Check balance
```

[Example Output]

```
Deposited: 500. New balance: 1500
Withdrew: 200. New balance: 1300
Current balance: 1300
```