Assignment Questions

Part 1: Nested Lists

Q1: Basic Example

Create a nested list to store information about a group of students. Each student's information includes their name, age, and grade. Write a program to:

- Print the name of the second student in the list.
- Calculate the average grade of all students.

Sample Data:

```
students = [
  ["Alice", 14, 85],
  ["Bob", 15, 78],
  ["Charlie", 14, 92],
  ["David", 15, 88]
]
```

Q2: Real-Life Scenario

A company wants to organize its employees into teams. Each team has several members with their names and their roles. Represent this structure using a nested list and write a program to:

- Print all members of a specific team.
- Add a new member to one of the teams.
- Calculate the total number of employees across all teams.

Sample Data:

```
teams = [
  ["Team A", [["Alice", "Manager"], ["Bob", "Developer"]]],
  ["Team B", [["Charlie", "Tester"], ["David", "Developer"], ["Eve", "Designer"]]]
]
```

Part 2: Nested Dictionaries

Q3: Basic Example

Create a nested dictionary to store details about a library. Each book should have a title, author, and the number of copies available. Write a program to:

- Add a new book to the library.
- Update the number of copies of an existing book.
- Print all book titles and their authors.

Sample Data:

```
library = {
    "Book1": {"title": "Python Basics", "author": "John Doe", "copies": 5},
    "Book2": {"title": "Machine Learning", "author": "Jane Smith", "copies": 2},
    "Book3": {"title": "Data Science", "author": "Sam Lee", "copies": 3}
}
```

Q4: Real-Life Scenario

Create a nested dictionary to represent a class of students where each student has their name, marks in three subjects, and their attendance percentage. Write a program to:

- Find and print the details of the student with the highest average marks.
- Update attendance for a specific student.
- Print the names of students with attendance less than 75%.

Sample Data:

```
class_data = {
    "student1": {"name": "Alice", "marks": {"math": 85, "science": 92, "english": 78}, "attendance": 88},
    "student2": {"name": "Bob", "marks": {"math": 78, "science": 81, "english": 89}, "attendance": 72},
    "student3": {"name": "Charlie", "marks": {"math": 90, "science": 87, "english": 85}, "attendance": 95}
}
```

Part 3: List Comprehension

Q5: Basic Example

Given a list of numbers, use list comprehension to:

- Create a new list containing only the even numbers.
- Create a list of squares of all numbers.

Sample Data:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Q6: Real-Life Scenario

A store has a list of prices for products. Apply a discount of 10% to all products priced above $100 and generate a new list of discounted prices using list comprehension.

Sample Data:

prices = [50, 120, 200, 80, 300, 90, 150]

Part 4: Dictionary Comprehension

Q7: Basic Example

Given a list of numbers, use dictionary comprehension to:

- Create a dictionary where the keys are numbers and the values are their cubes.

Sample Data:

numbers = [1, 2, 3, 4, 5]

Q8: Real-Life Scenario

A company has a dictionary of employee names and their salaries. Generate a new dictionary containing only those employees who earn more than $50,000.

Sample Data:

```
employees = {
    "Alice": 45000,
    "Bob": 52000,
    "Charlie": 61000,
    "David": 47000,
    "Eve": 75000
}
```

Bonus Challenge:

Combine your knowledge of nested collections and comprehensions to solve the following:
A school has a nested dictionary representing students and their scores in various subjects. Write a program to generate a dictionary of students who have scored above 90 in math.

Sample Data:

```
students = {
    "Alice": {"math": 95, "science": 85, "english": 78},
    "Bob": {"math": 89, "science": 92, "english": 88},
    "Charlie": {"math": 91, "science": 87, "english": 85}
}
```

Simulate CRUD (Create, Read, Update, Delete) operations for a **Student Management System**.

Hints for Using Nested Data Structures

1. Use a **dictionary** where the keys are unique student IDs and the values are dictionaries containing student details like name, age, grade, and attendance.
   Example:

```
students = {
    "1": {"name": "Alice", "age": 14, "grade": "9th", "attendance": 85},
    "2": {"name": "Bob", "age": 15, "grade": "10th", "attendance": 92}
}
```

2. **WHY** dictionary structure???
   This structure allows you to:
   - Quickly look up a student by their ID.
   - Add, update, or delete student details.

Menu-Driven Program: Student Management System

```
def display_menu():
    print("\n--- Student Management System ---")
    print("1. Add a New Student")
    print("2. View All Students")
    print("3. Update Student Details")
    print("4. Delete a Student")
    print("5. Exit")

def add_student(students):
    student_id = input("Enter Student ID: ")
    if student_id in students:
        print("Student ID already exists!")
        return
    name = input("Enter Student Name: ")
    age = int(input("Enter Student Age: "))
    grade = input("Enter Student Grade: ")
    attendance = float(input("Enter Attendance Percentage: "))
    students[student_id] = {
        "name": name,
        "age": age,
        "grade": grade,
        "attendance": attendance
    }
    print("Student added successfully!")
```

```python
def view_students(students):
    if not students:
        print("No students found!")
        return
    print("\n--- Student Records ---")
    for student_id, details in students.items():
        print(f"ID: {student_id}, Name: {details['name']}, Age: {details['age']},
              Grade: {details['grade']}, Attendance: {details['attendance']}%")
```

```python
def update_student(students):
    student_id = input("Enter Student ID to Update: ")
    if student_id not in students:
        print("Student ID not found!")
        return
    print("\n--- Update Student Details ---")
    print("1. Update Name")
    print("2. Update Age")
    print("3. Update Grade")
    print("4. Update Attendance")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        students[student_id]["name"] = input("Enter New Name: ")
    elif choice == 2:
        students[student_id]["age"] = int(input("Enter New Age: "))
    elif choice == 3:
        students[student_id]["grade"] = input("Enter New Grade: ")
    elif choice == 4:
        students[student_id]["attendance"] = float(input("Enter New Attendance Percentage: "))
    else:
        print("Invalid choice!")
        return
    print("Student details updated successfully!")
```

```python
def delete_student(students):
    student_id = input("Enter Student ID to Delete: ")
    if student_id in students:
        del students[student_id]
        print("Student deleted successfully!")
    else:
        print("Student ID not found!")
```

```python
def main():
    students = {}  # Nested dictionary to store student details
    while True:
        display_menu()
        choice = int(input("Enter your choice: "))
        if choice == 1:
            add_student(students)
        elif choice == 2:
            view_students(students)
        elif choice == 3:
            update_student(students)
        elif choice == 4:
            delete_student(students)
        elif choice == 5:
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid choice! Please try again.")

if __name__ == "__main__":
    main()
```

Explanation of CRUD Operations

1. **Create (Add a New Student)**
    - Prompts the user for student details (ID, name, age, grade, attendance).
    - Adds a new entry in the students dictionary.
2. **Read (View All Students)**
    - Iterates through the students dictionary and prints each student's details.
    - Handles the case where no students exist.
3. **Update (Update Student Details)**
    - Allows the user to modify specific details of a student by selecting their ID.
    - Provides options to update individual fields (name, age, grade, or attendance).
4. **Delete (Delete a Student)**
    - Removes a student entry from the students dictionary based on their ID.

Sample Output

```
--- Student Management System ---
1. Add a New Student
2. View All Students
3. Update Student Details
4. Delete a Student
5. Exit
Enter your choice: 1

Enter Student ID: 1
Enter Student Name: Alice
Enter Student Age: 14
Enter Student Grade: 9th
Enter Attendance Percentage: 85
Student added successfully!

--- Student Management System ---
1. Add a New Student
2. View All Students
3. Update Student Details
4. Delete a Student
5. Exit
Enter your choice: 2

--- Student Records ---
ID: 1, Name: Alice, Age: 14, Grade: 9th, Attendance: 85%
```