Python Data Structures Practice Exercises

Lists, Tuples, Sets Exercises

1. **List Manipulation**: Create a list of your favorite movies. Then write code to:
   - Add a new movie to the end
   - Insert a movie at the second position
   - Remove the third movie
   - Sort the list alphabetically
   - Print only the first three movies
2. **Tuple Operations**: Given the tuple of monthly expenses `(1200, 900, 1500, 1100, 1800)`:
   - Find the maximum and minimum expenses
   - Calculate the average monthly expense
   - Convert the tuple to a list, add a new expense, and convert back to a tuple
   - Try to modify an element directly (notice what happens)
3. **Set Operations**: Given two sets of student IDs, those who take Math: `{101, 103, 105, 107, 109}` and those who take Science: `{102, 103, 104, 105, 108}`:
   - Find students who take both subjects
   - Find students who take Math but not Science
   - Find students who take either Math or Science but not both
   - Find all unique student IDs

Dictionary Exercises

4. **Contact Manager**: Create a dictionary representing a contact list with names as keys and phone numbers as values. Write code to:
   - Add a new contact
   - Update an existing contact's number
   - Delete a contact
   - Print all contacts sorted by name
   - Search for a specific contact
5. **Frequency Counter**: Write a function that takes a string and returns a dictionary with each character as a key and its frequency as the value. Ignore spaces and make it case-insensitive.

6. **Nested Dictionaries**: Create a nested dictionary of students, where each student has a name, age, and a dictionary of subject grades. Write code to:
    - Add a new grade for an existing student
    - Calculate each student's average grade
    - Find the student with the highest grade in a specific subject
    - Print a formatted report for each student

Comprehension Exercises

7. **List Comprehension**: Use list comprehensions to solve the following:
    - Generate a list of squares for numbers 1 to 20
    - Filter out all odd numbers from a list of integers
    - Create a list of tuples containing numbers and their cubes for numbers 1 to 10
    - Convert a string to a list of ASCII values for each character
8. **Dictionary Comprehension**: Use dictionary comprehensions to:
    - Create a dictionary mapping numbers 1-10 to their squares
    - Invert a dictionary (swap keys and values)
    - Create a dictionary filtering out items with values less than 10 from an existing dictionary
    - Generate a word-length dictionary from a sentence
9. **Combined Data Structures**: Given a list of dictionaries representing products with 'name', 'price', and 'category' keys:
    - Use a list comprehension to filter products by category
    - Use a dictionary comprehension to create a price lookup by product name
    - Create a dictionary with categories as keys and lists of product names as values
    - Find the average price per category using comprehensions and dictionaries
10. **Data Transformation Challenge**: Given raw data about employees, use a combination of data structures and comprehensions to transform it into a structured format:
    - Parse a CSV-like string into a list of employee dictionaries
    - Group employees by department
    - Calculate average salary by department
    - Find the highest-paid employee in each department