# Tutorial 07

## Guided Tutorial Exercises (Easy)

Easy Questions

1. **Creating and Modifying a List**
    - **Exercise**: Create a list of five numbers. Then, add a new number at the end and remove the second number from the list.
    - **Concepts Covered**: List creation, **append()**, **pop()** or **remove()**.
2. **Accessing Tuple Elements**
    - **Exercise**: Given a tuple **(4, 9, 10, 23)**, write a function that returns the sum of the first and last elements.
    - **Concepts Covered**: Tuple indexing and basic arithmetic operations.
3. **Using Common Operations on a List**
    - **Exercise**: Create a list of numbers. Write a function that returns the sum, maximum, and length of the list.
    - **Concepts Covered**: **sum()**, **max()**, **len()**.
4. **List to Tuple Conversion**
    - **Exercise**: Given a list **[1, 2, 3, 4]**, write a function that converts it into a tuple.
    - **Concepts Covered**: Type conversion using the **tuple()** function.
5. **Nested Lists and Access**
    - **Exercise**: Given a nested list **[[1, 2, 3], [4, 5, 6], [7, 8, 9]]**, write a function to flatten this list and return the sum of all elements.
    - **Concepts Covered**: Nested lists, list comprehension or loops, **sum()**.
6. **List Comprehensions with Conditional Logic**
    - **Exercise**: Given a list of integers, use a list comprehension to create a new list containing only the odd numbers from the original list multiplied by 2.
    - **Concepts Covered**: List comprehensions, conditional statements.

## Unguided Tutorial Exercises (Hard)

Hard Questions

1. **Find the Longest Sublist**
    - **Exercise**: Given a list of sublists, write a function to find the sublist with the most elements and return it. If there are multiple sublists of the same maximum length, return the first one encountered.
    - **Concepts Covered**: List iteration, length comparison.
2. **Tuple Pair Sum**
    - **Exercise**: Given a list of tuples, each containing two numbers, write a function that returns a new list of numbers, each being the sum of the numbers in the tuples. For example, given **[(1, 2), (3, 4)]**, the function should return **[3, 7]**.
    - **Concepts Covered**: Tuple unpacking, list comprehension.
3. **Concatenate All Strings in a Tuple**
    - **Exercise**: Given a tuple containing strings, write a function that concatenates all the strings into a single string.
    - **Concepts Covered**: Tuple iteration, string concatenation.
4. **Filter Negative Numbers**
    - **Exercise**: Given a list of numbers, write a function that returns a new list containing only the positive numbers from the original list.
    - **Concepts Covered**: List comprehension with a conditional.
5. **Nested List Element Sum**
    - **Exercise**: Given a list of lists, each containing integers, write a function that returns the total sum of all integers in all lists. For example, given **[[1, 2], [3], [4, 5]]**, the function should return **15**.
    - **Concepts Covered**: Nested iteration, sum calculation.
6. **List Elements Multiplication**
    - **Exercise**: Given a list of numbers, write a function that returns the product of all the numbers in the list. For example, given **[1, 2, 3, 4]**, the function should return **24**.
    - **Concepts Covered**: Iteration, multiplication.