List Manipulation Exercises (Advanced)

Exercise 1: Create and Print List with Modifications

1. Define a list called `fruits` with the following elements: `apple`, `banana`, `cherry`, `date`, `elderberry`.
2. Print the first and last elements of the list.
3. Add two new fruits to the list, then print the updated list.
4. Remove the second element and print the updated list.

Expected Output:

```
apple
elderberry
['apple', 'banana', 'cherry', 'date', 'elderberry', 'grape', 'kiwi']
['apple', 'cherry', 'date', 'elderberry', 'grape', 'kiwi']
```

Exercise 2: Calculate List Length and Modify Elements

1. Create a list called `colors` with color names of your choice.
2. Print the length of the list.
3. Add a new color to the beginning of the list and another color to the end.
4. Print the updated list and its new length.

Expected Output:

```
5
['black', 'red', 'blue', 'green', 'yellow', 'purple']
7
```

Exercise 3: Iterate, Modify, and Print Elements

1. Define a list called `animals` with elements like `cat`, `dog`, `rabbit`, `hamster`, `lion`.
2. Use a `for` loop to print each animal in the list.

3. After the loop, replace `lion` with `tiger` and add a new animal, then print the list again.

Expected Output:

```
cat
dog
rabbit
hamster
lion
['cat', 'dog', 'rabbit', 'hamster', 'tiger', 'elephant']
```

Exercise 4: List Slicing with Modifications

1. Create a list called `books` with five book titles.
2. Print a slice of the first three elements and the last two elements.
3. Replace the last book title with `The Great Gatsby`.
4. Print the updated list and a slice that shows the middle three books.

Expected Output:

```
['book1', 'book2', 'book3']
['book4', 'book5']
['book1', 'book2', 'book3', 'book4', 'The Great Gatsby']
['book2', 'book3', 'book4']
```

Exercise 5: Print Elements with Step and Reverse the List

1. Create a list `cities` with names of cities.
2. Print every other city in the list using slicing.
3. Reverse the list order and print it.

Expected Output:

```
['city1', 'city3', 'city5']
['city5', 'city4', 'city3', 'city2', 'city1']
```

Exercise 6: Update List Elements with Conditions

1. Define a list called `vehicles` with elements like `car`, `bike`, `bus`, `train`.
2. Replace any vehicle with more than 3 letters with the word `truck`.
3. Print the updated list.

Expected Output:

```
['car', 'truck', 'bus', 'truck']
```

Exercise 7: Append, Extend, and Remove Elements

1. Create a list called `snacks` with two items of your choice.
2. Add a third item using `append()`.
3. Use `extend()` to add three more items at once.
4. Remove the first and last items, then print the final list.

Expected Output:

```
['snack2', 'snack3', 'snack4']
```

Exercise 8: Insert, Sort, and Remove Elements

1. Define a list called `hobbies` with four hobbies.
2. Insert a new hobby at index 2.
3. Sort the list alphabetically.
4. Remove the second hobby and print the updated list.

Expected Output:

```
['hobby1', 'hobby3', 'new_hobby', 'hobby4']
```

Exercise 9: Remove Elements and Clear the List

1. Create a list called `countries` with five country names.
2. Remove the last element using `pop()` and the element at index 1.
3. Clear the list completely and print it to verify it's empty.

Expected Output:

```
[]
```

Exercise 10: Sum of Elements with Conditions

1. Create a list called `numbers` with some integers.
2. Calculate the sum of all even numbers in the list.
3. Calculate the sum of all odd numbers in the list.
4. Print both sums.

Expected Output:

```
Sum of even numbers: 20
Sum of odd numbers: 15
```

Exercise 11: Counting Elements with Conditions

1. Define a list `ages` with several ages.
2. Count how many ages are above 18.
3. Count how many ages are between 13 and 19 (inclusive).
4. Print both counts.

Expected Output:

```
Number of ages above 18: 3
Number of ages between 13 and 19: 2
```

Exercise 12: Use While Loop to Find List Average

1. Create a list `scores` with some integers.
2. Use a `while` loop to calculate the average of all elements in the list.
3. Print the average.

Expected Output (based on list):

```
Average score: 15
```

Exercise 13: Find Maximum and Minimum Elements with Indexes

1. Create a list `heights` with different heights in centimeters.
2. Find the tallest and shortest heights.
3. Print each value along with its index in the list.

Expected Output:

```
Tallest height: 180 at index 2
Shortest height: 150 at index 0
```

Exercise 14: Merge, Sort, and Slice Lists

1. Define two lists: `list1` with integers from 1 to 5 and `list2` with integers from 6 to 10.
2. Merge the two lists into one.
3. Sort the merged list in descending order.
4. Print a slice of the largest three numbers in the list.

Expected Output:

```
[10, 9, 8]
```

Advanced Exercises (optional)

Exercise 1: Nested List Manipulation and Sum Calculation

1. Define a nested list called `grades` that contains the grades of students in different subjects:

   ```
   grades = [
       [85, 90, 88],  # Student 1
       [78, 81, 85],  # Student 2
       [92, 89, 94],  # Student 3
       [88, 76, 95],  # Student 4
       [79, 84, 80]   # Student 5
   ]
   ```

2. Calculate the average grade for each student and print it in the format: `Student 1 Average: 87.67`.
3. Find the highest grade across all students and print which student received it, along with the grade.
4. Calculate and print the average grade for each subject (the first element of each sublist is the grade for Subject 1, and so on).

Expected Output:

```
Student 1 Average: 87.67
Student 2 Average: 81.33
...
Highest Grade: 95 by Student 4
Subject 1 Average: 84.4
Subject 2 Average: 84.0
Subject 3 Average: 88.4
```

Exercise 2: Inventory Management with Multiple Lists

1. Define two lists: `products` containing product names and `quantities` containing the stock quantities for each product. Ensure they are aligned by index:

   ```
   products = ["Apples", "Bananas", "Oranges", "Grapes", "Strawberries"]
   quantities = [50, 30, 40, 20, 25]
   ```

2. Ask the user for a product name and a quantity to sell. If the product is in stock and has enough quantity, reduce the stock accordingly and print the updated stock for that product. If not, print an appropriate message.
3. After processing the sale, find the product with the lowest stock and print a warning if the stock is below 10.
4. Add a new product and quantity to the inventory lists based on user input, then print the updated lists.

Example User Interaction:

```
Enter product name: Apples
Enter quantity to sell: 15
Updated stock for Apples: 35
Low stock warning: Grapes (20 remaining)
Enter a new product to add: Blueberries
Enter quantity: 30
Updated inventory: ["Apples", "Bananas", "Oranges", "Grapes", "Strawberries",
Updated quantities: [35, 30, 40, 20, 25, 30]
```

Exercise 3: Transaction Logs with Filtering and Aggregation

1. Create a list of transactions represented as tuples, each containing a transaction ID, type (`"deposit"` or `"withdrawal"`), and amount. Example:

   ```
   transactions = [
       (1001, "deposit", 500),
       (1002, "withdrawal", 200),
       (1003, "deposit", 300),
       (1004, "withdrawal", 100),
       (1005, "deposit", 400)
   ]
   ```

2. Separate deposits and withdrawals into two separate lists. Print both lists.
3. Calculate and print the total amount for deposits and the total amount for withdrawals.
4. Filter the transactions to include only those with amounts over 250, then print this filtered list.

Expected Output:

```
Deposits: [(1001, "deposit", 500), (1003, "deposit", 300), (1005, "deposit", 4
Withdrawals: [(1002, "withdrawal", 200), (1004, "withdrawal", 100)]
Total Deposits: 1200
Total Withdrawals: 300
Transactions over 250: [(1001, "deposit", 500), (1003, "deposit", 300), (1005,
```