📝 Practice: Single List Operations

🎯 Learning Objectives

By completing these exercises, you will:

- Master working with lists using loops
- Learn to implement common list operations without built-in functions
- Develop problem-solving skills and algorithmic thinking
- Understand how built-in functions work internally

📋 Instructions

- Write your own code without using built-in functions (unless specified)
- Use loops and conditional statements to solve each problem
- Test your code with different test cases
- Add comments to explain your logic

🔢 Practice Problems

1️⃣ Sum of Numbers in a List

Write a Python program to find the sum of all numbers in a list **without using** `sum()` .

Example:

```
numbers = [10, 20, 30, 40, 50]
# Output: 150
```

💡 **Hint:** Start with `total = 0` and add each number using a loop.

2️⃣ Minimum Number in a List

Write a program to find the smallest number in a list **without using** `min()` .

Example:

```
numbers = [45, 12, 78, 23, 9, 56]
# Output: 9
```

💡 **Hint:**

- Assume the first number is the smallest
- Compare each number one by one and update if you find a smaller one

3️⃣ Maximum Number in a List

Write a program to find the largest number in a list **without using** `max()` .

Example:

```
numbers = [45, 12, 78, 23, 9, 56]
# Output: 78
```

💡 **Hint:**

- Assume the first number is the largest
- Compare each number one by one and update if you find a bigger one

4️⃣ Count Even and Odd Numbers

Write a program to count how many numbers in a list are even and how many are odd.

Example:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# Output: Even: 5, Odd: 5
```

💡 **Hint:** Use the `%` operator to check divisibility by 2.

5️⃣ Count Positive and Negative Numbers

Write a program to count how many numbers are positive and how many are negative.

Example:

```
numbers = [10, -5, 20, -15, 0, 30, -8]
# Output: Positive: 3, Negative: 3
```

💡 **Hint:** Check with `if num > 0` and `if num < 0` .

6️⃣ Find the Average of List Elements

Find the average value of all numbers in a list **without using** `sum()` **or** `len()` .

Example:

```
numbers = [10, 20, 30, 40, 50]
# Output: 30.0
```

💡 **Hint:** Use a loop to count elements and add them manually, then divide total by count.

7️⃣ Find Second Largest Number

Write a program to find the second largest number in a list **without using** `sort()` **or** `max()` .

Example:

```
numbers = [45, 12, 78, 23, 56, 89]
# Output: 78 (second largest after 89)
```

💡 **Hint:** Use two variables — one for largest and one for second largest. Update while looping.

8️⃣ Reverse a List

Write a program to reverse a list **without using** `reverse()` **or slicing** (`[::-1]`).

Example:

```
numbers = [1, 2, 3, 4, 5]
# Output: [5, 4, 3, 2, 1]
```

💡 **Hint:** Use a loop to build a new list by inserting each element at the beginning.

9️⃣ Find the Product of All Numbers

Write a program to find the product of all numbers in a list.

Example:

```
numbers = [2, 3, 4, 5]
# Output: 120 (2 × 3 × 4 × 5)
```

💡 **Hint:** Start with `product = 1` and multiply each number inside a loop.

🔟 Find the Difference Between Largest and Smallest

Write a program to find the difference between the largest and smallest number in the list — but do it **without using** `max()` **or** `min()` .

Example:

```
numbers = [45, 12, 78, 23, 9, 56]
# Output: 69 (78 - 9)
```

💡 **Hint:** First find both manually using loops, then subtract them.

1️⃣ 1️⃣ Find All Numbers Greater Than a Given Value

Ask the user for a number `x` and print all elements in the list greater than `x`.

Example:

```
numbers = [10, 25, 30, 15, 40, 5]
x = 20
# Output: [25, 30, 40]
```

💡 **Hint:** Use a loop and an `if` statement to check each number.

1️⃣ 2️⃣ Check if a Number Exists in the List

Ask the user for a number and check if it exists in the list using a loop (**not `in` keyword**).

Example:

```
numbers = [10, 25, 30, 15, 40]
search = 30
# Output: "30 exists in the list"
```

💡 **Hint:** Use a flag variable and a loop to search through the list.

1️⃣ 3️⃣ Count Occurrences of a Number

Ask the user for a number and count how many times it appears in a list **without using `.count()`**.

Example:

```
numbers = [5, 10, 5, 20, 5, 30]
search = 5
# Output: 3
```

💡 **Hint:** Use a counter and increment it each time the number matches.

1️⃣ 4️⃣ Find Both Smallest and Largest in One Loop

Write a program to find both smallest and largest numbers in a list using a **single loop**.

Example:

```
numbers = [45, 12, 78, 23, 9, 56]
# Output: Smallest: 9, Largest: 78
```

💡 **Hint:** Initialize both variables with the first element, then update both in the same loop.

1️⃣ 5️⃣ Swap First and Last Elements

Write a program to swap the first and last elements of a list.

Example:

```
numbers = [10, 20, 30, 40, 50]
# Output: [50, 20, 30, 40, 10]
```

💡 **Hint:** Use temporary variables or tuple unpacking to swap values.

🌟 Bonus Challenges

🎁 Bonus 1: Remove Duplicates

Write a program to remove all duplicate elements from a list **without using** `set()` .

Example:

```
numbers = [1, 2, 2, 3, 4, 4, 5]
# Output: [1, 2, 3, 4, 5]
```

🎁 Bonus 2: Find Common Elements Between Two Lists

Given two lists, find the elements that appear in both lists **without using** `set()` .

Example:

```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
# Output: [4, 5]
```

🎁 Bonus 3: Rotate List by N Positions

Rotate a list to the right by `n` positions.

Example:

```
numbers = [1, 2, 3, 4, 5]
n = 2
# Output: [4, 5, 1, 2, 3]
```

📊 Testing Your Solutions

For each problem, test with these cases:

- ✅ **Normal case**: Regular input
- ✅ **Edge case**: Empty list, single element, all same values
- ✅ **Large numbers**: Test with bigger values
- ✅ **Negative numbers**: Include negative values where applicable

💡 Learning Tips

1. **Start Simple**: Begin with problem 1 and work your way up
2. **Plan First**: Write pseudocode before coding
3. **Test Often**: Run your code after each problem
4. **Debug Smart**: Use print statements to see what's happening
5. **Compare Solutions**: After solving, think about other approaches
6. **Understand Why**: Don't just solve—understand how it works!

🎯 Success Criteria

You've mastered single list operations when you can:

- ✅ Traverse a list using loops
- ✅ Use conditional logic to filter/count elements
- ✅ Track multiple variables while looping
- ✅ Build new lists from existing ones
- ✅ Solve problems without relying on built-in functions
