Lists & Tuples Level-2

Practice Problem 25

📌 Description:

> - Write a Python program that prints (as a list) the elements of `listA` that are **not** in `listB` as a list.
> - If the lists have the same elements, print an empty list.
> - If `listA` is an empty list, print an empty list.

🔹 Expected Output:

ListA	ListB	Output
[1, 2, 3, 4]	[1, 2]	[3, 4]
[1, 2, 3, 4]	[1, 2, 3]	[4]
[1, 2, 3, 4]	[1, 2, 3, 4]	[]
[]	[1, 3]	[]

Practice Problem 26

📌 Description:

> - Write a Python program that calculates the distance between two 3D points.
> - The points are represented by two lists with three elements. The first element is the x-coordinate. The second element is the y-coordinate. The third element is the z-coordinate.

Formula to find the Distance:

$$AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Where: $A\,(x_1, y_1, z_1)$ and $B\,(x_2, y_2, z_2)$

🔹 Expected Output:

pointA	pointB	Output
[1, 2, 3]	[1, 2, 3]	0
[3, 4, 5]	[1, 3, 5]	2.23607
[-3, 4, -5]	[2, 0, -4]	6.48074

Important: The value 0 can be expressed as 0.0 in the output (a float).

🔸 **Hints:**

- The value of the distance must always be positive.

Practice Problem 27

📌 **Description:**

> - Write a Python program that prints a list with the elements that `listA` and `listB` have in common.
> - If they don't have any elements in common, print an empty list.
> - The program should **not** assume that the lists have the same length.
> - You may assume that each element will only appear once in each list.

🔹 **Expected Output:**

pointA	pointB	Output
[1, 2, 3]	[1, 2, 3]	[1, 2, 3]
[4, 5, 6]	[1, 4, 5]	[4, 5]
[3, 4, 5]	[1, 2, 3]	[3]
[4, 5, 6]	[1, 2, 3]	[]

Practice Problem 28

📌 **Description:**

> - Write a Python program that prints the **second largest** value in a list.
> - If the list is empty or only has one element, print `None`.

🔹 **Expected Output:**

List	Output
[1, 2, 3, 4]	3
[1, 2]	1
[2]	None
[]	None

🔸 **Hints:**

- You might want to sort the list in ascending order.

Practice Problem 29

📌 **Description:**

> - Write a Python program that prints the **second smallest** value in a list.
> - If the list is empty or only has one element, print `None`.

🔹 **Expected Output:**

List	Output
[1, 2, 3, 4]	2
[1, 3]	3
[2]	None
[]	None

Practice Problem 30

📌 **Description:**

- Write a Python program that creates and print a dictionary that maps **each element in a list** to its corresponding **frequency** (how many times it occurs in the list).
- The test should be **case-sensitive**. Therefore, `"A"` should not be considered the same element as `"a"`.

Expected Output:

List	Output
["a", "a", "b", "c", "a", "b"]	{"a": 3, "b": 2, "c": 1}
[1, 2, 3, 4, 3, 2, 1, 2]	{1: 2, 2: 3, 3: 2, 4: 1}

Practice Problem 31

Description:

- Write a Python program that prints a "flattened" version of a list that contains nested lists.
- "Flattened" means that all the elements in the nested lists should be added to a main list such that it doesn't contain any nested lists, just the individual.
- The list could contain other elements that are not lists or other sequences, so you must check the type of each element and act appropriately.
- If the list is empty, print an empty list.

Expected Output:

List	Output
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]	[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, [7, 8, 9]]	[1, 2, 3, 4, 5, 6, 7, 8, 9]
[["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"]]	["a", "b", "c", "d", "e", "f", "g", "h", "i"]

Hints:

- Nested loops can be helpful to write this program.
- If you are familiar with list comprehension in Python, this is one alternative.
- You can also implement the solution recursively.

Practice Problem 32

📌 Description:

- Write a Python program that generates and prints all the possible permutations of a list.
- A **permutation** is a possible arrangement of the elements of the list. For example, [2, 1, 3] is a permutation of [1, 2, 3].
- Print each permutation as a list on a separate line. You can print them as lists or tuples.
- Include the list itself as a permutation.

🔹 Expected Output:

List	Output
[1, 2, 3]	[1, 2, 3]
	[1, 3, 2]
	[2, 1, 3]
	[2, 3, 1]
	[3, 1, 2]
	[3, 2, 1]

🔸 Hints:

- The `permutations` function of the `itertools` module can be very helpful to solve this exercise. You can import this module with `import itertools`.