Tasks – Lists Intermediate (final list lab)

This lab comes **after all list basics** (18–22). It uses **negative indices**, **del**, **valid index checks**, **not in**, and harder tasks that combine lists with loops, conditions, and input. Do labs **18_lists**, **19_lists_methods**, **20_lists_variables_copies**, **21_in_notin_combined**, and **22_lists_2d_3d** first. Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

Part 1 – Negative indices, del, and non-existing

Task 1.1 – Negative indices -1, -2, -3 (`list_neg_idx.py`)

- Create `list_neg_idx.py`. Assign `data = ["a", "b", "c", "d", "e"]`. Print the element at index -1 (last), then -2 (second to last), then -3. Then print the element at index -len(data) (same as first). Use variables or direct indexing.

Expected output:

```
e
d
c
a
```

Task 1.2 – del by index (`list_del.py`)

- Create `list_del.py`. Assign `vals = [10, 20, 30, 40, 50]`. Use `del vals[2]` to remove the element at index 2. Print the list (should be [10, 20, 40, 50]). Then use `del vals[-1]` to remove the last element and print again (should be [10, 20, 40]).

Expected output:

```
[10, 20, 40, 50]
[10, 20, 40]
```

Task 1.3 – Check index before access (avoid non-existing) (`list_safe_index.py`)

- Create `list_safe_index.py`. Assign `items = ["apple", "banana", "cherry"]`. Ask the user for an index (int). If the index is valid (0 <= index < len(items)), print the element at that index. Otherwise print "Invalid index". Use a variable for the index and an if/else. Test with 1 (valid) and 10 (invalid).

Expected output (example – input 1):

```
Enter index: 1
banana
```

(Run with 10: "Invalid index".)

Task 1.4 – Value not in list (`list_not_in.py`)

- Create `list_not_in.py`. Assign `allowed = ["yes", "y", "ok"]`. Read a string from the user. If the string is **not** in the list (`if answer not in allowed:`), print "Not allowed". Else print "Allowed". Use a variable for the input.

Expected output (example – input "n"):

```
Enter response: n
Not allowed
```

Task 1.5 – del only if index valid (`list_del_safe.py`)

- Create `list_del_safe.py`. Start with `nums = [5, 10, 15, 20]`. Ask the user for an index. If the index is valid (0 <= index < len(nums)), use `del nums[index]` and print the

new list. If invalid, print "Invalid index" and do not change the list. Print the list in both cases. Use if/else.

Expected output (example – input 2):

```
Enter index to remove: 2
[5, 10, 20]
```

(Run with 10: "Invalid index" and list stays [5, 10, 15, 20].)

Part 2 – Harder combined scenarios

Task 2.1 – Find index of first occurrence (`list_find_index.py`)

- Create `list_find_index.py`. Assign `words = ["apple", "banana", "cherry", "banana", "date"]`. Ask the user for a word. Use a for loop with index: `for i in range(len(words)):` and if `words[i] == target`, print "Found at index " + str(i) and break. If the loop ends without finding (use a variable like found = False, set True when found), print "Not found". Use variables for the target and found flag.

Expected output (example – "banana" then "mango"):

```
Enter word: banana
Found at index 1
```

(Run with "mango": "Not found".)

Task 2.2 – New list with only positive numbers (`list_filter_positive.py`)

- Create `list_filter_positive.py`. Ask how many numbers, read n, then read n numbers into a list. Create a second list (empty). Loop over the first list: if the number is positive (n > 0), append it to the second list. Print the original list and the new list. Use variables and if inside the loop.

Expected output (example – 4 numbers: 3, -1, 5, -2):

```
How many numbers? 4
Number 1: 3
Number 2: -1
Number 3: 5
Number 4: -2
Original: [3, -1, 5, -2]
Positives only: [3, 5]
```

Task 2.3 – Menu: add, list, remove by index, quit (`list_menu.py`)

- Create `list_menu.py`. Start with an empty list. Use a loop (while True or for with large range + break). Each time: print menu "1=Add 2=List 3=Remove by index 4=Quit", read choice (int). If 1: read a number (or string) and append to the list. If 2: print the list. If 3: ask for an index; if valid (0 <= index < len(list)), use del list[index] and print "Removed"; else print "Invalid index". If 4: print "Bye" and break. Use variables for choice, index, and the list.

Expected output (example – Add 10, Add 20, List, Remove index 0, List, Quit):

```
1=Add 2=List 3=Remove by index 4=Quit
Choice: 1
Value: 10
...
Choice: 2
[10, 20]
...
Choice: 3
Index: 0
Removed
...
Choice: 2
[20]
...
Choice: 4
Bye
```

Task 2.4 – Remove all occurrences (new list) (`list_remove_all.py`)

- Create `list_remove_all.py`. Assign `nums = [1, 2, 3, 2, 4, 2, 5]`. Ask the user for a value to remove (int). Create a new list (empty). Loop over nums: if the element is **not**

equal to the value, append it to the new list. Print the original list and the new list. Use variables.

Expected output (example – remove 2):

```
Enter value to remove: 2
Original: [1, 2, 3, 2, 4, 2, 5]
After removing 2: [1, 3, 4, 5]
```

Task 2.5 – List stats (sum, min, max, positive count) (`list_stats_full.py`)

- Create `list_stats_full.py`. Ask how many numbers, read n, then read n numbers into a list. If the list is empty, print "Empty list". Otherwise: use a loop to compute sum; use a loop to find min and max (no built-in min/max); use a loop to count how many are positive. Print "Sum: X, Min: Y, Max: Z, Positive count: W". Use variables and if for min/max and positive check.

Expected output (example – 4 numbers 3, -1, 5, 2):

```
How many numbers? 4
Number 1: 3
Number 2: -1
Number 3: 5
Number 4: 2
Sum: 9, Min: -1, Max: 5, Positive count: 3
```

Task 2.6 – Simple todo (add, list, remove by number, quit) (`list_todo.py`)

- Create `list_todo.py`. Start with an empty list of task names (strings). Loop: print "1=Add 2=List 3=Remove 4=Quit", read choice. If 1: read a task name and append. If 2: loop over the list and print each as "1. [task]", "2. [task]" (1-based). If 3: ask "Which number? (1-based)"; if 1 <= num <= len(list), remove the element at index (num - 1) with del and print "Removed"; else "Invalid". If 4: break. Use variables and check index before del.

Expected output (example – Add "Buy milk", Add "Call mom", List, Remove 1, List, Quit):

```
1=Add 2=List 3=Remove 4=Quit
Choice: 1
Task: Buy milk
...
Choice: 2
1. Buy milk
2. Call mom
...
Choice: 3
Which number? 1
Removed
...
Choice: 2
1. Call mom
...
Choice: 4
Bye
```

Done

You've used: **negative indexing**, **del**, **valid index checks**, **not in**, and combined lists with loops, conditions, input, and menu-style programs.