Tasks – 2D and 3D Lists (Nested Lists)

Practice **2D lists** (list of lists) and **3D lists** (list of list of lists): creating, indexing `[i][j]` and `[i][j][k]`, looping with nested for, and building from input. Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

Part 1 – 2D lists: creating and accessing

Task 1.1 – Create and print a 2D list (`list_2d_basic.py`)

- Create `list_2d_basic.py`. Assign a 2D list, e.g. `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`. Print the whole list with `print(matrix)`. Then print the first row: `matrix[0]`. Then print the element at row 1, column 2: `matrix[1][2]` (value 6).

Expected output:

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[1, 2, 3]
6
```

Task 1.2 – len() and dimensions (`list_2d_len.py`)

- Create `list_2d_len.py`. Assign `grid = [[10, 20], [30, 40], [50, 60]]`. Print `len(grid)` (number of rows). Print `len(grid[0])` (number of columns in the first row). Use variables or direct print. Assume all rows have the same length.

Expected output:

```
3
2
```

Task 1.3 – Access first row, last row, first column element (`list_2d_access.py`)

- Create `list_2d_access.py`. Assign `data = [["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"]]`. Print the first row `data[0]`. Print the last row `data[-1]`. Print the element in the first row, first column: `data[0][0]`. Print the element in the last row, last column: `data[-1][-1]`.

Expected output:

```
['a', 'b', 'c']
['g', 'h', 'i']
a
i
```

Part 2 – 2D lists and loops

Task 2.1 – Print each row (`list_2d_loop_rows.py`)

- Create `list_2d_loop_rows.py`. Assign `matrix = [[1, 2], [3, 4], [5, 6]]`. Use `for row in matrix:` and print each row. Each row prints on one line.

Expected output:

```
[1, 2]
[3, 4]
[5, 6]
```

Task 2.2 – Print each element (nested loop) (`list_2d_loop_elements.py`)

- Create `list_2d_loop_elements.py`. Assign `grid = [[1, 2, 3], [4, 5, 6]]`. Use a nested loop: `for row in grid:` then `for cell in row:` and print each cell (one per line, or use `print(cell, end=" ")` and then `print()` after each row to get rows of numbers). Print so you see one line per row: 1 2 3, then 4 5 6.

Expected output:

```
1 2 3
4 5 6
```

Task 2.3 – Sum all elements in 2D list (`list_2d_sum.py`)

- Create `list_2d_sum.py` . Assign `nums = [[1, 2], [3, 4], [5, 6]]` . Use a variable total = 0 and nested loops (for row in nums, for cell in row) to add each cell to total. Print the total. Result: 21.

Expected output:

```
21
```

Task 2.4 – Access by index with range (`list_2d_index_loop.py`)

- Create `list_2d_index_loop.py` . Assign `matrix = [[10, 20], [30, 40]]` . Use `for i in range(len(matrix)):` and inside `for j in range(len(matrix[i])):` and print `matrix[i][j]` (or print the element). Print each element so you see 10, 20, 30, 40 (one per line or in order).

Expected output:

```
10
20
30
40
```

Part 3 – 2D lists and input

Task 3.1 – Build 2D list: N rows, M numbers per row (`list_2d_input.py`)

- Create `list_2d_input.py`. Ask "Number of rows?" and read n. Ask "Number of columns?" and read m. Start with an empty list `grid = []`. Use a for loop (n times): create an empty row, then an inner loop (m times) to read a number and append it to the row; then append the row to grid. Print the 2D list. Use variables. Test with 2 rows and 3 columns, entering e.g. 1,2,3 and 4,5,6.

Expected output (example):

```
Number of rows? 2
Number of columns? 3
Row 1 number 1: 1
Row 1 number 2: 2
Row 1 number 3: 3
Row 2 number 1: 4
Row 2 number 2: 5
Row 2 number 3: 6
[[1, 2, 3], [4, 5, 6]]
```

Task 3.2 – Sum each row, print row sums (`list_2d_row_sums.py`)

- Create `list_2d_row_sums.py`. Assign `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`. Use a for loop over rows. For each row, use a variable row_sum = 0 and a loop over the row to add each element; then print "Row X sum: Y" (X is row index + 1, Y is row_sum). Use str() and variables.

Expected output:

```
Row 1 sum: 6
Row 2 sum: 15
Row 3 sum: 24
```

Part 4 – 3D lists

Task 4.1 – Create and access 3D list (`list_3d_basic.py`)

- Create `list_3d_basic.py`. A 3D list is a list of 2D lists. Assign e.g. `cube = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]`. So `cube[0]` is the first "layer" (a 2D list), `cube[0][1]` is the second row of the first layer, `cube[0][1][0]` is the first element of that row (value 3). Print `cube[0]`. Print `cube[1][0][1]` (value 6). Print the last element: `cube[-1][-1][-1]` (value 8).

Expected output:

```
[[1, 2], [3, 4]]
6
8
```

Task 4.2 – len() in 3D (`list_3d_len.py`)

- Create `list_3d_len.py`. Assign `data = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]`. Print `len(data)` (number of "layers"). Print `len(data[0])` (rows in first layer). Print `len(data[0][0])` (elements in first row of first layer). Use variables or direct print.

Expected output:

```
2
2
2
```

Task 4.3 – Loop over 3D and sum all elements (`list_3d_sum.py`)

- Create `list_3d_sum.py`. Assign `cube = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]`. Use total = 0 and three nested for loops: for layer in cube, for row in layer, for cell in row: total += cell. Print the total. Result: 36.

Expected output:

```
36
```

Task 4.4 – Build simple 3D from input (2 layers, 2x2 each) (`list_3d_input.py`)

- Create `list_3d_input.py`. Build a 3D list with 2 layers, each layer 2 rows and 2 columns (so 8 numbers total). Use an outer loop (2 layers), then a loop for 2 rows, then a loop for 2 columns; read a number and append to the row, then append row to layer, then append layer to the main list. Print the 3D list. Use variables. Test with numbers 1 through 8.

Expected output (example – 1,2,3,4,5,6,7,8):

```
Layer 1 row 1 col 1: 1
Layer 1 row 1 col 2: 2
...
[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]
```

Done

You've used: **2D lists** (list of lists), **access** `[i][j]`, **len()** for rows and columns, **nested loops** to iterate and sum, **building 2D from input**; **3D lists** (list of 2D), **access** `[i][j][k]`, **three nested loops**, and building a simple 3D from input.