# list-tools

Extensions coded in Opusmodus by Arthur Stammet in November 2025

## Overview

The **List Tools Extensions** provide utility functions for slicing and reorganizing lists in structured ways. They are especially useful for algorithmic composition, rhythmic grouping, or any situation where you want to partition or traverse lists with regular steps and patterns.
Functions included:

- step-list → take every *n*-th element from a list
- step-list-loop → partition a list into *n* interleaved sublists
- pendulum-list → partition into *n* sublists, alternating forward/backward order

The set includes three functions:

1. **step-list** return every step-th element from lst, starting at index 0.
2. **step-list-loop** partitions lst into step sublists, each starting at offsets 0..(step-1).
3. **pendulum-list** partitions lst into step sublists, alternating forward and backward order.

## Functions

### 1. step-list

```
(step-list lst step)
```

- **Purpose:** Return every step-th element from lst, starting at index 0.
- **Arguments:**
  - lst → input list
  - step → integer stride
- **Returns:** A list of elements spaced by step.

Example:

```
(step-list '(a b c d e f g h i j) 2) ;; → (a c e g i)
```

### 2. step-list-loop

```
(step-list-loop lst step)
```

- **Purpose:** Partition lst into step sublists, each starting at offsets 0..(step-1).
- **Arguments:**
  - lst → input list
  - step → number of partitions
- **Returns:** A list of sublists, each containing every step-th element starting at its offset.

Example:

```
(step-list-loop '(a b c d e f g h i j) 2)
;; → ((a c e g i) (b d f h j))
```

## 3. pendulum-list

```
(pendulum-list lst step)
```

- **Purpose:** Partition lst into step sublists, alternating forward and backward order.
- **Arguments:**
  - lst → input list
  - step → number of partitions
- **Returns:** A list of sublists, where even offsets are forward, odd offsets are reversed.

Example:

```
(pendulum-list '(a b c d e f g h i j) 3)
;; → ((a d g j) (i f c) (b e h))
```

# Cheat Sheet

| Function | Behaviour | Example Input → Output |
|---|---|---|
| step-list | Take every step-th element | (step-list '(a b c d e f) 2) → (a c e) |
| step-list-loop | Partition into step sublists | (step-list-loop '(a b c d e f) 2) → ((a c e) (b d f)) |
| pendulum-list | Partition with alternating order | (pendulum-list '(a b c d e f g h) 3) → ((a d g) (h e b) (c f)) |

# Source Code

```
(in-package :Opusmodus)

;;; -----------------------------------------------------------
;;; List Tools Extensions
;;; Author: Arthur Stammet
;;; Created: November 2025
;;; Location: ~/Opusmodus/User Source/Extensions/
;;; -----------------------------------------------------------
;;; Overview:
;;; Utility functions for slicing and reorganizing lists in structured ways.
;;;   - step-list       → take every N-th element
;;;   - step-list-loop  → partition into N interleaved sublists
;;;   - pendulum-list   → partition into N sublists, alternating forward/backward
order
;;;
;;; Functions:
;;;   (step-list lst step)
;;;   (step-list-loop lst step)
;;;   (pendulum-list lst step)
;;;
;;; Usage Examples:
```

```
;;;   (setq my-list '(a b c d e f g h i j))
;;;
;;;   ;; Every 2nd element
;;;   (step-list my-list 2)
;;;   ;; → (a c e g i)
;;;
;;;   ;; Partition into 2 interleaved lists
;;;   (step-list-loop my-list 2)
;;;   ;; → ((a c e g i) (b d f h j))
;;;
;;;   ;; Partition into 3 lists with pendulum alternation
;;;   (pendulum-list my-list 3)
;;;   ;; → ((a d g j) (i f c) (b e h))
;;;
;;; ------------------------------------------------------------

(defun step-list (lst step)
  "Return every STEP-th element from LST, beginning with index 0."
  (loop for i from 0 below (length lst) by step
        collect (nth i lst)))

(defun step-list-loop (lst step)
  "Return lists of STEP-th elements from LST, starting at offsets 0..(STEP-1)."
  (loop for offset from 0 below step
        collect (loop for i from offset below (length lst) by step
                      collect (nth i lst))))

(defun pendulum-list (lst step)
  "Return lists of STEP-th elements from LST, alternating forward and backward
passes."
  (loop for offset from 0 below step
        collect (if (evenp offset)
                    (loop for i from offset below (length lst) by step
                          collect (nth i lst))
                    (reverse (loop for i from offset below (length lst) by step
                                   collect (nth i lst))))))
```

# Summary

- step-list → simple stride extraction.
- step-list-loop → interleaved partitioning.
- pendulum-list → pendulum alternation for expressive sequencing.
- Place the file in ~/Opusmodus/User Source/Extensions/ to load automatically.