Version: `0.1.0`

`run-liners`: User Manual

Contents

1. Introduction

This package provides a set of functions that create "run-in" lists and enumerations, following guidelines similar to the **Chicago Manual of Style**. You can produce run-in enumerations, bullet lists, terms/definitions, or verses, all joined in a single line with flexible separators and optional words like "and," "or," or *none* (i.e. omit a coordinator).

We'll illustrate each function using **the left column**: a `typst` block containing the Typst code you'd write and **the right column**: the rendered result.

2. `run-in-enum`: Run-in numbered lists

`run-in-enum` is typically used for run-in enumerations like "(1) apple, (2) banana, and (3) cherry." It offers these parameters:

- **`separator`**: Defaults to `"auto"` (which tries to detect commas and switch to semicolons).
- **`coordinator`**: The word or phrase before the last item—defaults to `"and"`, can be `"or"`, `"and/or"`, or `none`.
- **`numbering-pattern`**: A pattern like `"(1)"`, `"(A)"`, or any format the `numbering()` call supports.
- **`numbering-formatter`**: A function that can style or transform the generated number.

2.1. Example: Basic Three-Item Enumeration

```typst
#run-in-enum(
  [Apple],
  [Banana],
  [Cherry]
)
```

(1) Apple, (2) Banana, and (3) Cherry

This example uses **all defaults**:

- `separator: "auto"` → because there's no comma in the content, it uses commas.
- `coordinator: "and"` → puts "and" before the last item.
- `numbering-pattern: "(1)"` → "(1) (2) (3)"
- `numbering-formatter: (it) => [#it]` → no special styling.

2.2. Example: Coordinator = none, 4 Items, Comma in content

If one of your items contains `[text, with comma]`, the `auto` separator chooses `"; "`.

```typst
#run-in-enum(
  coordinator: none,
  [Item One],
  [Item Two, comma here],
  [Item Three],
  [Item Four]
)
```

(1) Item One; (2) Item Two, comma here; (3) Item Three; (4) Item Four

Notice it used semicolons because of comma detection, **and** no coordinator word (because `coordinator: none`).

2.3. Example: Custom Numbering Pattern & Styling

You can change `numbering-pattern` to `"(A)"` or `"(i)"` and style the numbers with `numbering-formatter`.

```typst
#run-in-enum(
  numbering-pattern: "(A)",
  numbering-formatter: (txt) => [#strong(txt)],
  [Alpha],
  [Beta],
  [Gamma]
)
```

(A) Alpha, **(B)** Beta, and **(C)** Gamma

Here, `(A)`, `(B)`, `(C)` appear in **bold**.

2.4. Example: Single Item & Two Items Edge Cases

When only one item is present, no separator or coordinator is used. With two items, you'll see just one separator plus "and/or/or" (depending on coordinator).

```typst
// Single item:
#run-in-enum([LoneItem])
```

(1) LoneItem

```typst
// Two items:
#run-in-enum([First], [Second])
```

(1) First and (2) Second

```typst
// Two items, no coordinator:
#run-in-enum(coordinator: none, [First],
[Second])
```

(1) First, (2) Second

```typst
// Three items:
#run-in-enum([First], [Second], [Third])
```

(1) First, (2) Second, and (3) Third

```typst
// Three items, no coordinator:
#run-in-enum(coordinator: none, [First],
[Second], [Third])
```

(1) First, (2) Second, (3) Third

3. `run-in-list`: Run-in bullet points

`run-in-list` behaves like `run-in-enum` but uses a **bullet (or any marker)** before each item. The rest of the logic is the same—`separator`, `coordinator`, etc. See the "run-in-enum" section for **how `auto` detection** and **`coordinator`** work.

3.1. Example1: Default Bullets

```typst
  #run-in-list(
    [Apple],
    [Banana],
    [Cherry]
  )
```

• Apple, • Banana, and • Cherry

You get bullets `[•]` for each item by default.

3.2. Example: Custom Marker, Coordinator = 'or'

```typst
#run-in-list(
  coordinator: "or",
  marker: [#sym.ballot.check],
  [First],
  [Second],
  [Third, comma item]
)
```

☑ First; ☑ Second; or ☑ Third, comma item

Because of the comma, it uses semicolons. The coordinator is now `"or"`.

4. `run-in-terms`: Run-in term definitions

`run-in-terms` is for run-in **terms and definitions**. By default, each term is **bold** (`[#strong(term)]`) followed by the definition, and you still get the same **separator** and **coordinator** logic as in `run-in-enum`.

4.1. Example: Three Terms, No Coordinator

```typst
#run-in-terms(
  coordinator: none,
  (
    ([Goal], [Summarize the prior research]),
    ([Methods], [Outline the approach]),
    ([Outcome], [Analyze the results])
  )
)
```

Goal: Summarize the prior research, **Methods**: Outline the approach, **Outcome**: Analyze the results

See how no "and" or "or" appears. It detects commas if they exist.

4.2. Example: Custom Term Formatter

```typst
#run-in-terms(
  term-formatter: (txt) => [#underline(txt)],
  ([Key], [Value 1]),
  ([Another Key], [Value 2])
)
```

Key: Value 1 and Another Key: Value 2

Each term is underlined, with a colon after the term.

5. `run-in-verse`: Run-in poetry lines

`run-in-verse` simply joins lines (verses) with a given **separator** (default: `[ /~]`), and no coordinator. It's much simpler since you typically don't want "and" in between lines of poetry.

5.1. Example: Default Verse Joining

```typst
#run-in-verse(
  [Line 1],
  [Line 2],
  [Line 3]
)
```

Line 1 / Line 2 / Line 3

5.2. Example: Custom Separator

```typst
// Overriding the default with a comma
+ space
#run-in-verse(
  separator: ", ",
  [Line A],
  [Line B],
  [Line C]
)
```

Line A, Line B, Line C

6. Direct `run-in-join`

All these functions ultimately call **`run-in-join`**, which you can use yourself if you want to create custom transformations. You supply:

1. **`separator`** (or `"auto"` for comma detection).
2. **`coordinator`** or `none`.
3. A **spread** of content items (`..content`).

6.1. Example: Using `run-in-join` Directly

```typst
#run-in-join(
  coordinator: "and/or",
  [First item],
  [Second item],
  [Third item, comma triggers semicolon],
  [Final item]
)
```

First item; Second item; Third item, comma triggers semicolon; and/or Final item

You can see that it inserts semicolons (auto detection) and “and/or” before the last item.

7. Conclusion

That’s your tour of the **run-in** functions! Remember:

- If you want enumerations with `(1)` or `(A)` prefixes, use **`run-in-enum`**.
- For bullet lists, use **`run-in-list`**.
- For term–definition pairs, **`run-in-terms`**.
- For verse lines, **`run-in-verse`**.
- Or just call **`run-in-join`** yourself for custom needs.

Explore different **separators** (like `[ /~]`, `", "`, or `"; "`) and **coordinators** (`"and"`, `"or"`, `"none"`, etc.) to get the exact run-in style your text requires.

Happy Typst-ing!