Intro to Recursive Structures & Processes

Let's code the `Node` class together in VS Code!

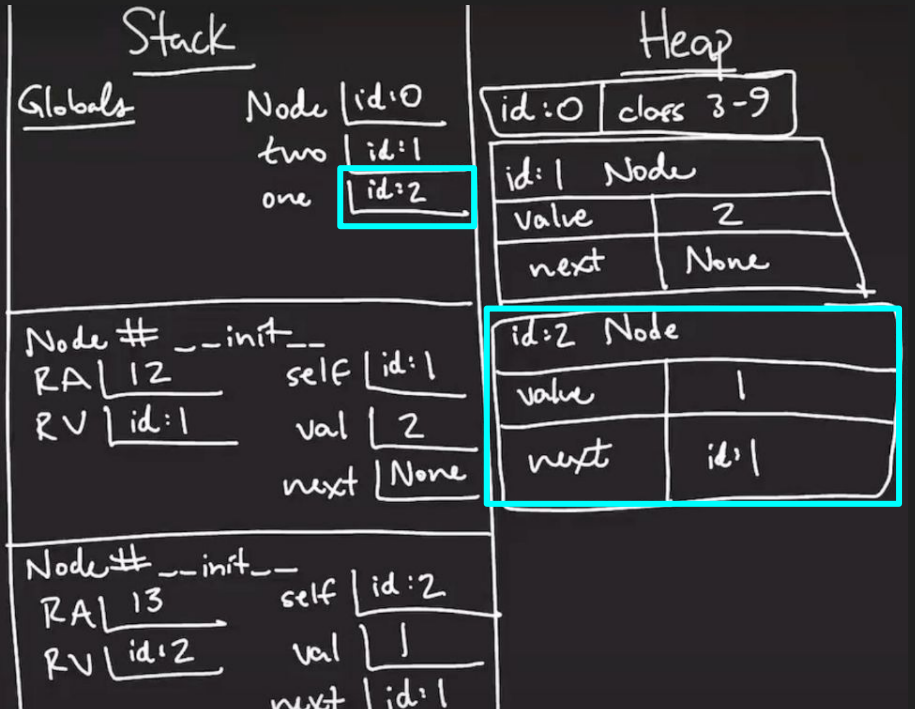
Create a new file named `cl/linked_list.py`

As a reminder, we want our `Node` class definition to have:

- Attributes:
  - `value: int`
  - `next: Node | None` "or"
- Initializer/constructor with parameters:
  - `self`
  - `val: int`
  - `next: Node | None`

What happens when you try to print a string representation of a `Node` object?

`print(one)` What does this print?




```
<__main__.Node object
at 0x100633950>
```

What does the `0x100633950` mean?!

If we had the power to write our own string representation of a Node object, what might we *want* `print(one)` to print?

What about the `to_str` function (from Monday's LS assignment)?

```
28  def to_str(head: Node | None) -> str:
29      if head is None:
30          return "None"
31      else:
32          rest: str = to_str(head.next)
33          return f"{head.value} -> {rest}"
34
35
36  print(to_str(one))
```

What does the function do?

Why is it useful?

Is it recursive? How do we know?

… but would this function change the output of `print(one)`?

Surely, there must be a better way…
Perhaps with some… ✨magic✨?

A Recursive `last` Algorithm Demo

1. When you are asked,
   "Are you the last Node?"

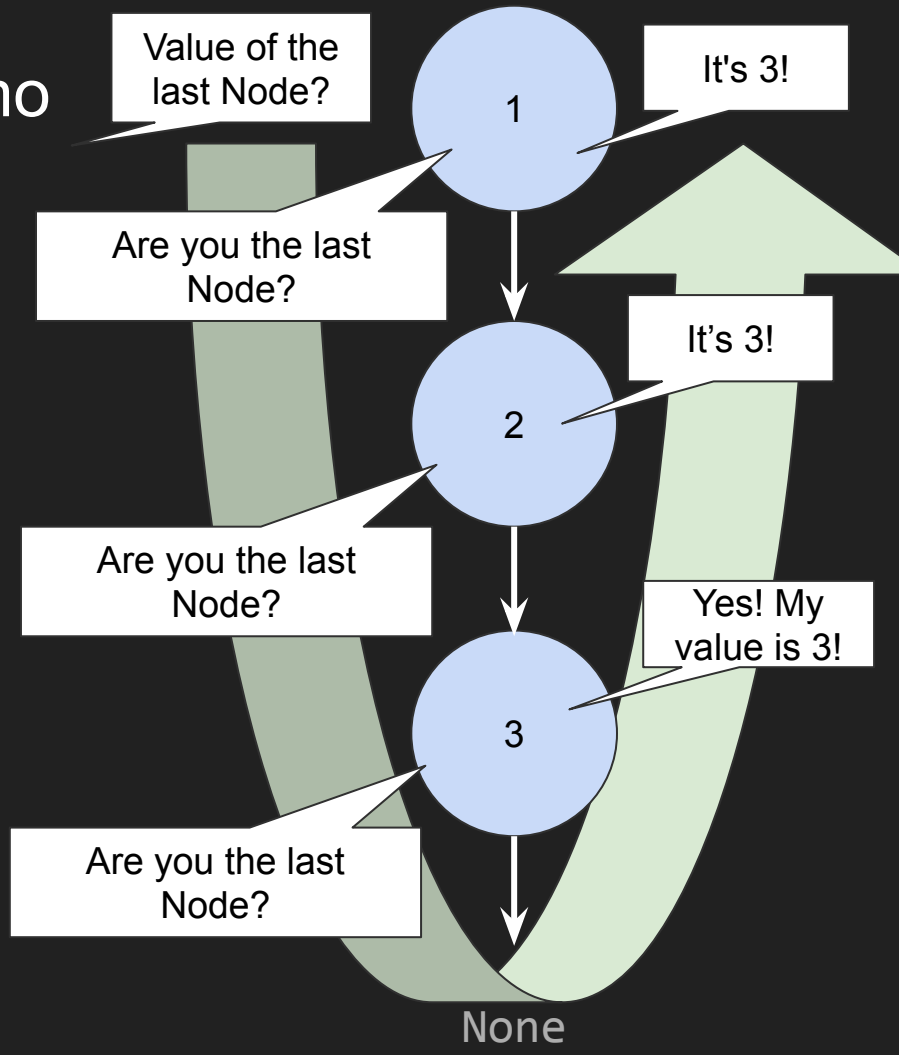
If you're ***not***:

2. Ask the ***next*** Node,
   "Are you the last Node?"
   Wait patiently for an answer!

3. Once the answer is returned back to you, turn to the person who asked you and give them this answer.

If it ***is***:

2. Tell them, "yes!" and share your value.



Let’s write the `last` function in VS Code!

Recursive function checklist:

Base case:

- ❏ Does the function have a clear base case?
  - ❏ Ensure the base case returns a result directly (without calling the function again).
- ❏ Will the base case *always* be reached?

Recursive case:

- ❏ Does the function have a recursive case that *progresses toward the base case*?
  - ❏ Does the recursive call have the right arguments? The function should call itself on a simpler or smaller version of the problem.
- ❏ Have you tested your function with multiple cases, including edge cases?