Hack110 Sign-Up Form!

When: Saturday, April 18th from 10 AM - 12 AM (Midnight)

Where: Sitterson and Fred Brooks 1st floor (Floor 0)

Who can join? Anyone in COMP 110! No prior experience required.

Bring a partner, find one there, or go solo!

Come for a fun day of coding, workshops and events (**food and CLE credit will be provided**):

- Choose 1 of many tracks like web dev, game dev, etc - and WIN PRIZES too!
- Go to various **workshops & events** such as: Navigating the CS Major, Resume workshop, ice cream station, and kahoot trivia and MORE!
- Link: Sign-Up via the QR code
  - If you have a partner, **ONLY ONE OF YOU** has to sign up - you will just enter your partner's info in the form.

Instagram

Sign-Up Here!

Hack110 Pre Workshops

Food & Drinks Provided!

When: Thursday, April 9th; 5PM - 8PM

Where: Sitterson 014

Workshops:

Web Dev: 5 - 6PM

(HTML + CSS)

Game Dev: 6 - 7PM

(PyGame)

App (iOS) Dev: 7 - 8PM

(Swift - requires a Mac)

Practice with Recursive Structures & Processes

Announcements

Re: Assignments:

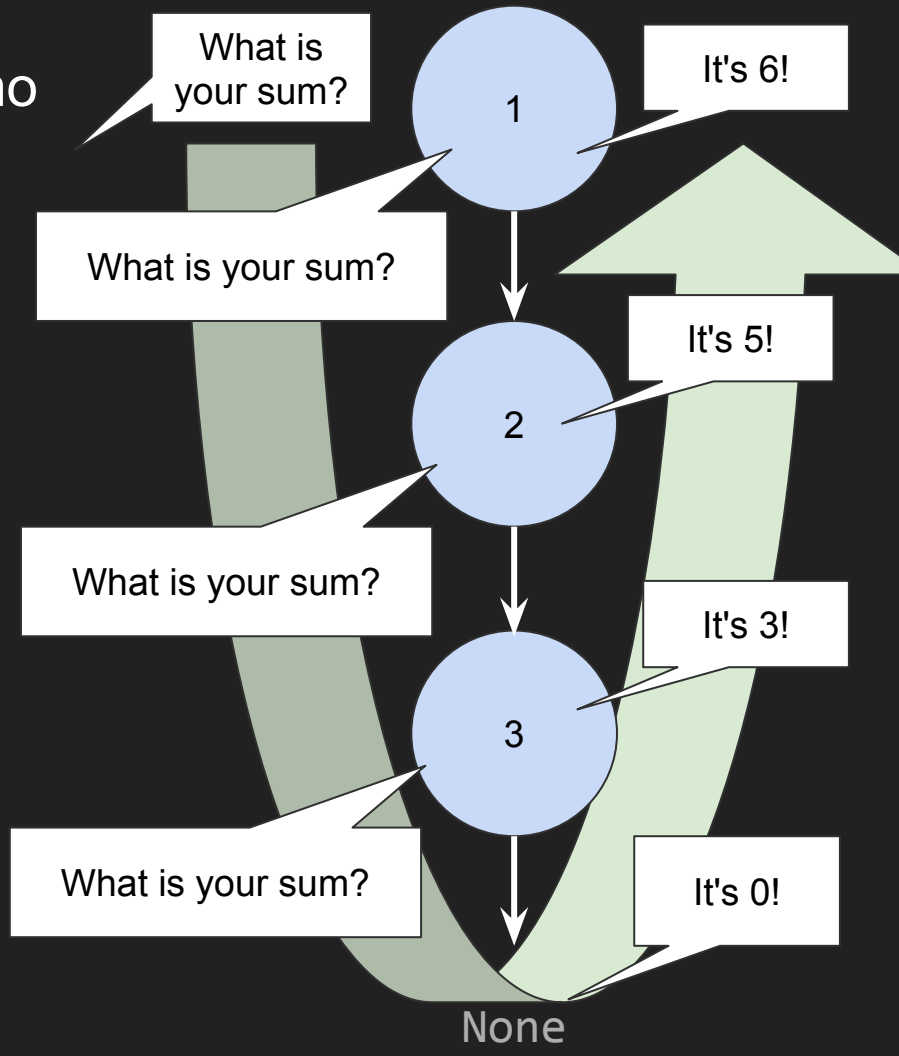
- **LS16: Recursive Structures** due today at 11:59pm
- **EX07: River Simulation** due tomorrow (April 9) at 11:59pm

Recall: A Recursive `sum` Algorithm Demo

1. When you are asked, "what is your sum?"

2. Ask the ***next*** Node, "what is your sum?"
   Wait patiently for an answer!

3. Once the answer is returned back to you, add ***your value to it***, then turn to the person who asked you and give them this answer.



Visualizing the `sum` function calls

Diagramming the `sum` function call

```python
from __future__ import annotations

class Node:
    value: int
    next: Node | None

    def __init__(self, val: int, next: Node | None):
        self.value = val
        self.next = next

# Note: There are no errors!
two: Node = Node(2, None)
one: Node = Node(1, two)

def sum(head: Node | None) -> int:
    if head is None:
        return 0
    else:
        rest: int = sum(head.next)
        return head.value + rest

print(sum(one))
```

A Recursive `last` Algorithm Demo

1. When you are asked,
   "What is the value of the last Node?"

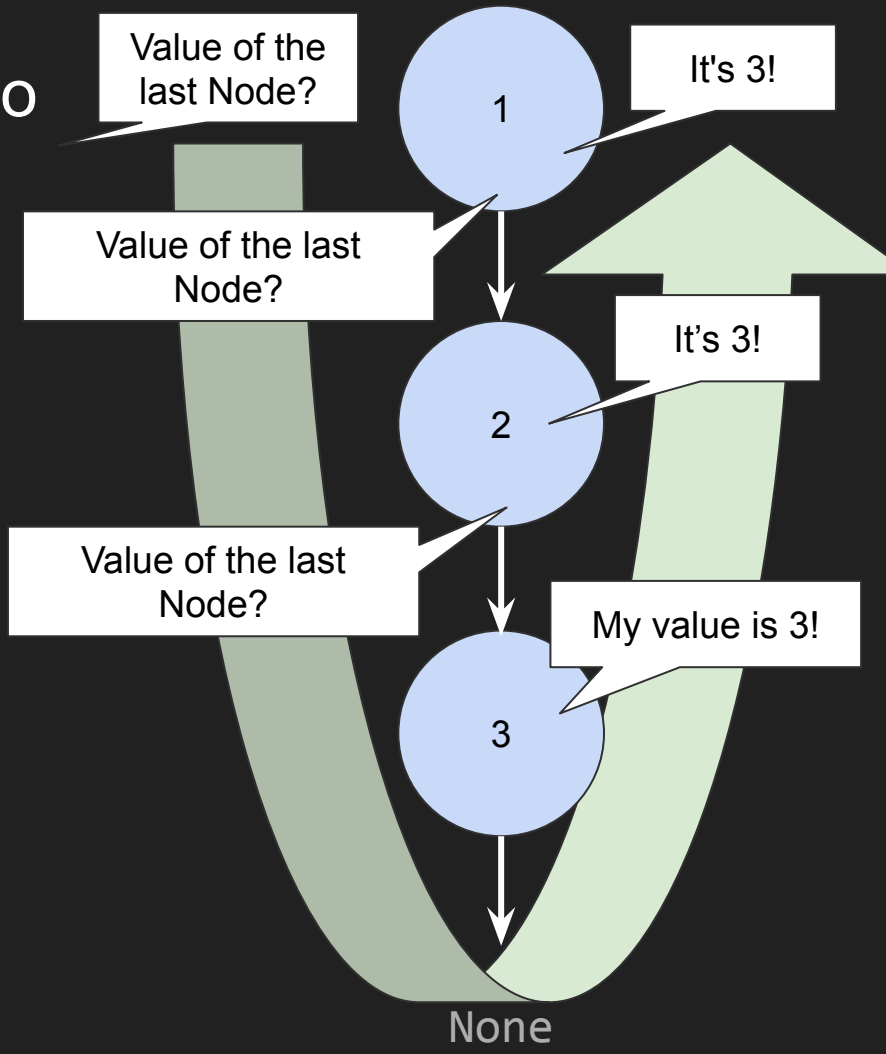
If you're ***not the last Node***:

2. Ask the ***next*** Node,
   "What is the value of 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 you ***are the last Node***:

2. Tell them, "my value is ____!" and share your value.



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

Visualizing the `last` function calls

More practice!

`insert_after` Algorithm Demo

1. When you are asked,
   "Can you add a Node with a value of 211 after the Node with value 210?"

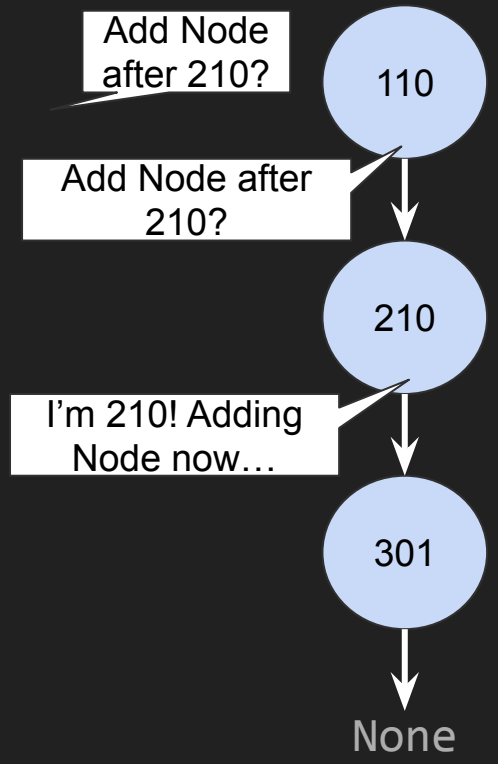
If your value ***is not 210***:

2. Ask the ***next*** Node,
   "Can you add a Node with a value of 211 after the Node with value 210?"
   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 your value **is 210**:

2. Invite a new friend to the list! You will now point to them, and they will point to the person you were previously pointing to. New Node, you'll say "I was added!!"



`insert_after` Algorithm Demo

1. When you are asked,
   "Can you add a Node with a value of 211 after the Node with value 210?"

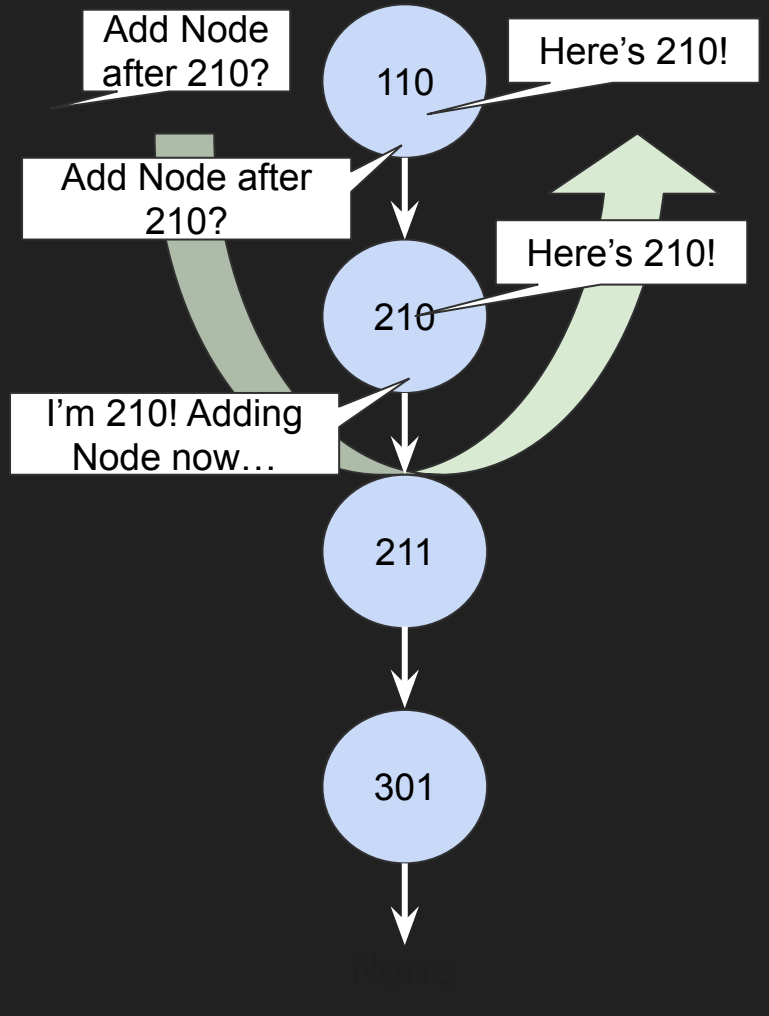
If your value ***is not 210***:

2. Ask the ***next*** Node,
   "Can you add a Node with a value of 211 after the Node with value 210?"
   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 your value **is 210**:

2. Invite a new friend to the list! You will now point to them, and they will point to the person you were previously pointing to. New Node, you'll say "I was added!!"



Let’s write pseudocode for the `insert_after` function

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

`recursive_range` Algorithm

Create a recursive function called `recursive_range` that will create a linked list of Nodes with values that increment from a start value up to an end value (exclusive). E.g.,

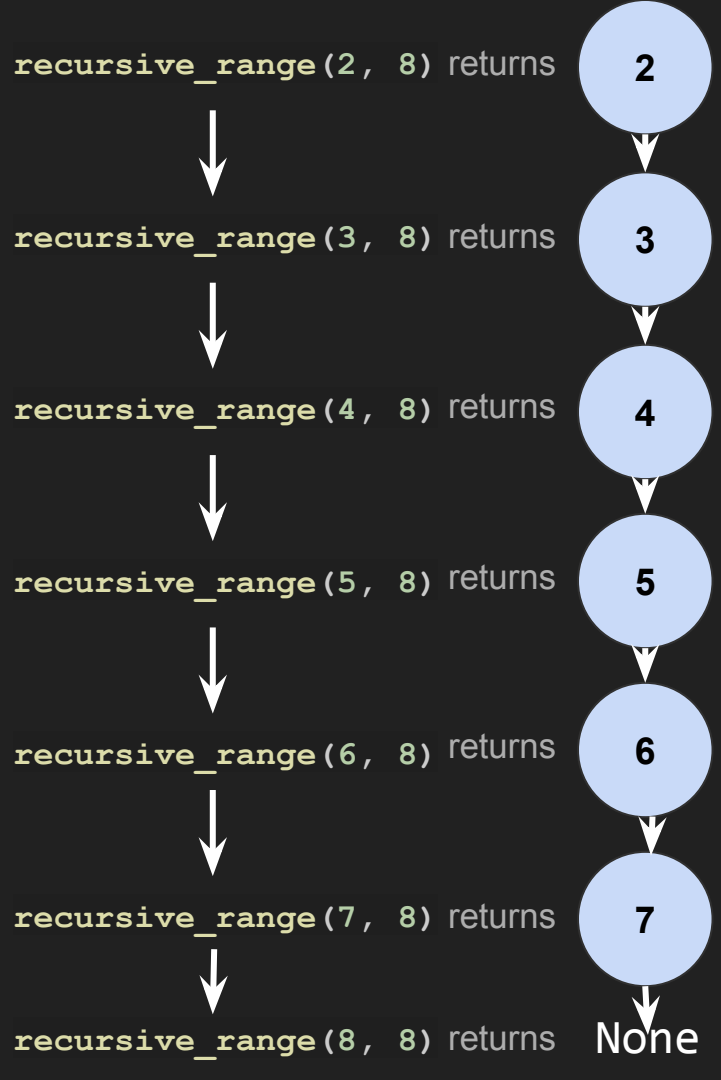
`recursive_range(start=2, end=8)` would return:

2 -> 3 -> 4 -> 5 -> 6 -> 7 -> None

Conceptually, what will our **base case** be?

What will our **recursive case** be?

What is an **edge case** for this function?
How could we account for it?



When "building" a new linked list in a recursive function:

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:

- ❏ Determine what the ***first*** value of the new linked list will be
- ❏ Then "build" the ***rest*** of the list by recursively calling the building function
- ❏ Finally, return a new ***Node(first, rest)***, representing the new linked list

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