CS61A

LINKED LISTS, MUTABLE TREES

LOGISTICS AND REMINDERS

- Lab08 due **Today**
- Midterm 2 is in exactly **one week**
- Mid-Semester Feedback
  - Most seemed happy with my explanations :)
  - Mixed feedback about timing: too fast, too slow, just right
  - group work is meh?

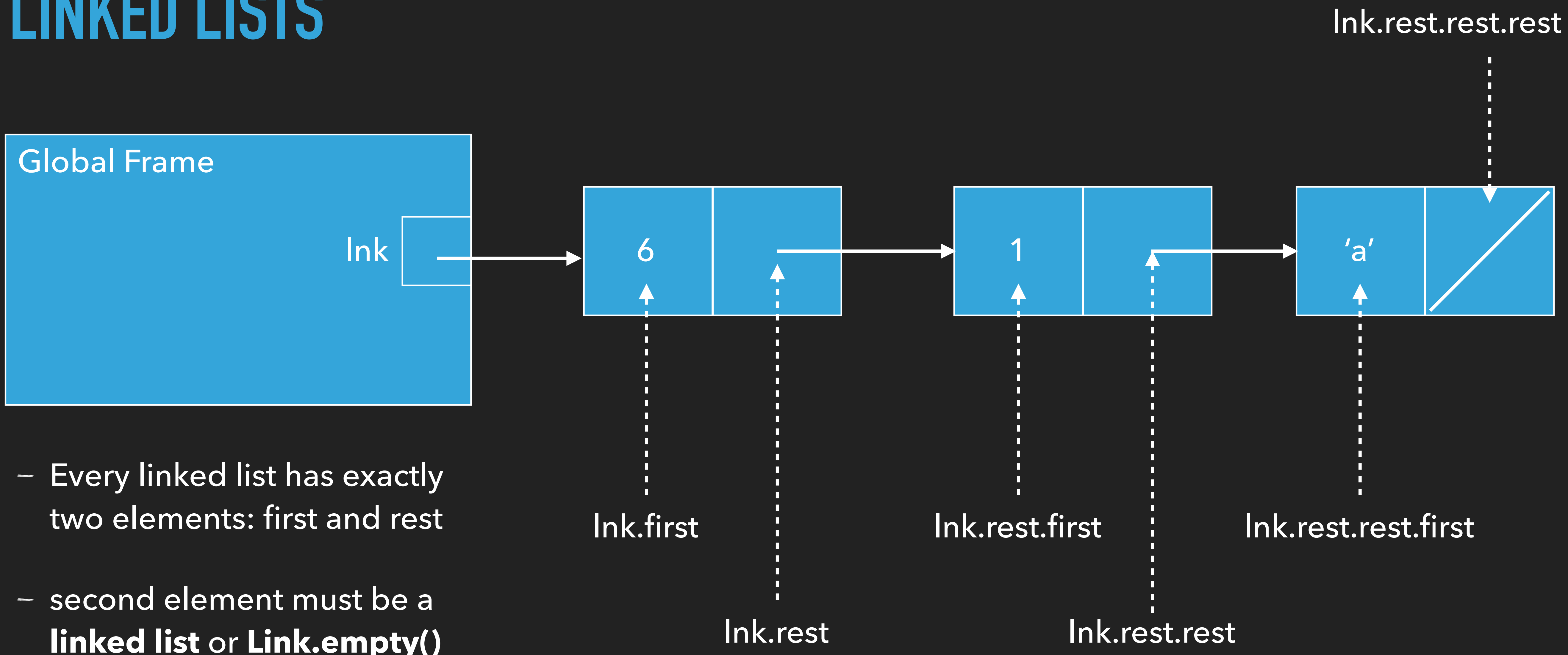
LOGISTICS AND REMINDERS

- Lab08 due **Today**
- Midterm 2 is in exactly **one week**
- Mid-Semester Feedback
  - Most seemed happy with my explanations :)
  - Mixed feedback about timing: too fast, too slow, just right
  - group work is meh?

AGENDA

- Linked Lists
- Mutable Trees

LINKED LISTS

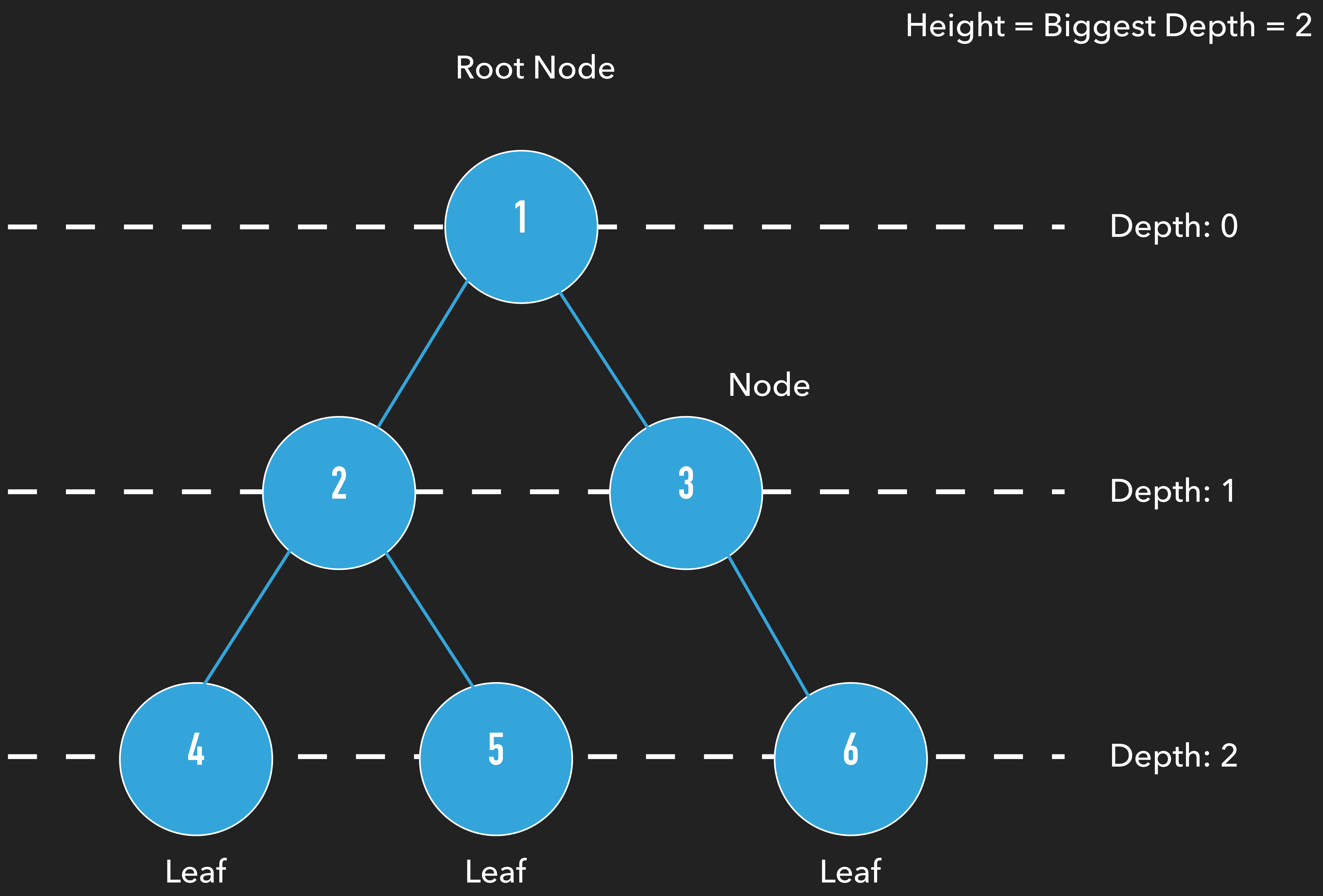


- Every linked list has exactly two elements: first and rest
- second element must be a **linked list** or **Link.empty()**
- Nothing new: just sideways trees

TIPS FOR SOLVING LINKED LIST PROBLEMS

- Recursion tends to be a common way to solve problems because you don't know how many times to call .rest
  - Makes for loops hard, but while loops let many problems be solved both **recursively** and **iteratively**
  - Try and see both ways
- Common base cases:
  - if lnk is Link.empty
  - if lnk is Link.empty or lnk.rest is Link.empty
    - short circuiting can matter!!
- Common mistakes, calling **.first** or **.rest** on Link.empty() which errors!

MUTABLE TREES—LOOK THE SAME



SO WHAT'S CHANGED? #1

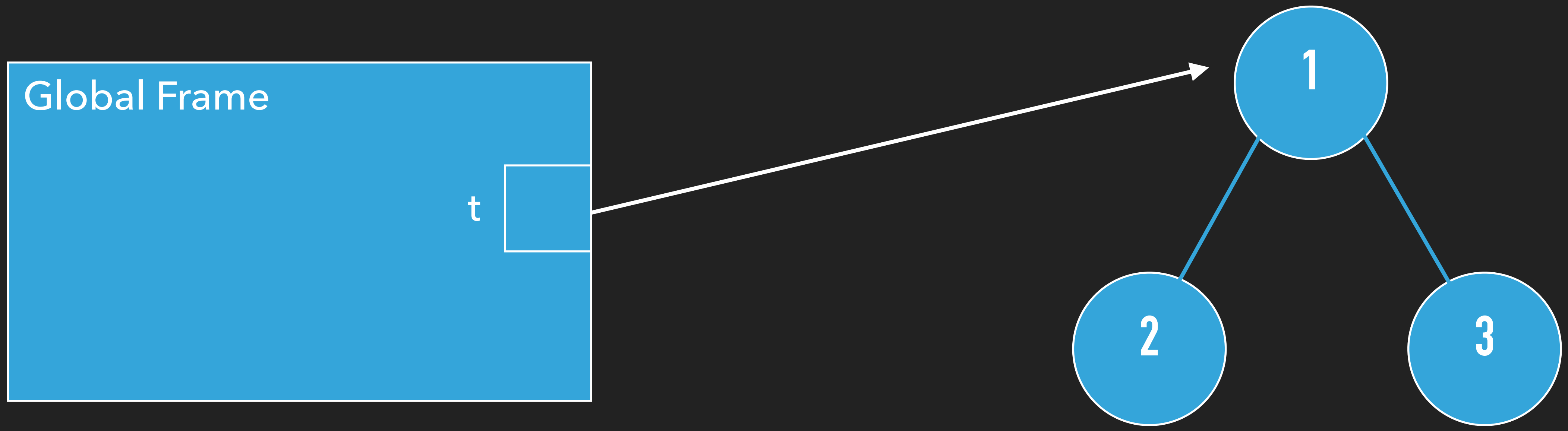
- Make trees - **tree(label, branches=[])**
- Get label of root node - **label(t)**
- Get branches - **branches(t)**
- Is the tree a leaf? - **is_leaf(t)**

- Make trees - **Tree(label, branches=[])**
- Get label of root node - **t.label**
- Get branches - **t.branches**
- Is the tree a leaf? - **t.is_leaf()**

**T is capital in Tree, and function selectors vs object attributes

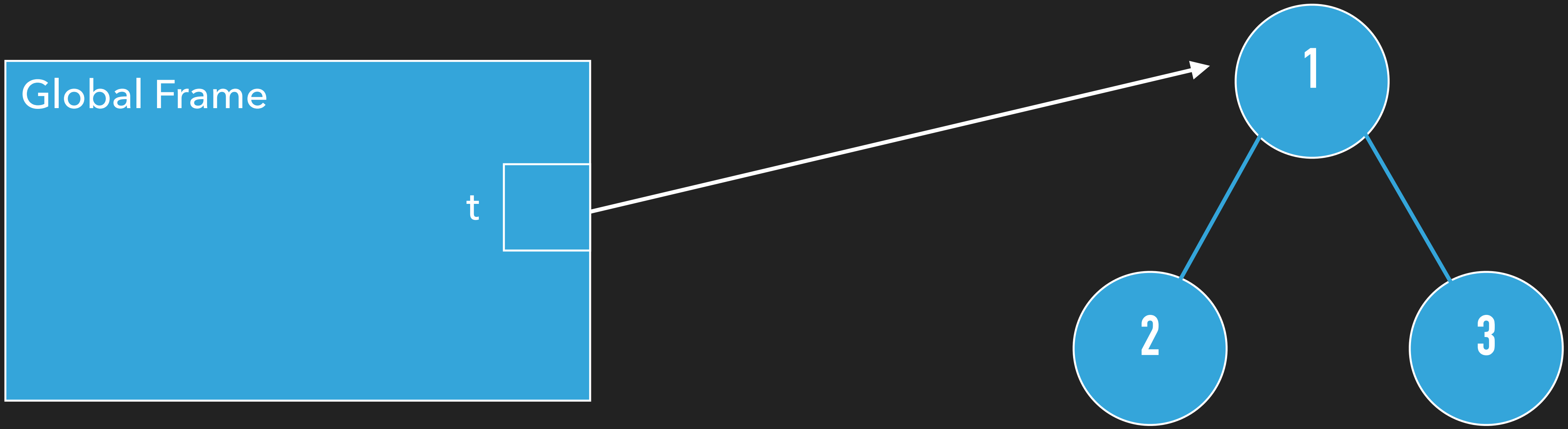
SO WHAT'S CHANGED? #2 MUTABILITY!!

SO WHAT'S CHANGED? #2 MUTABILITY!!



```
>>> t = Tree(1, [Tree(2), Tree(3)])
```

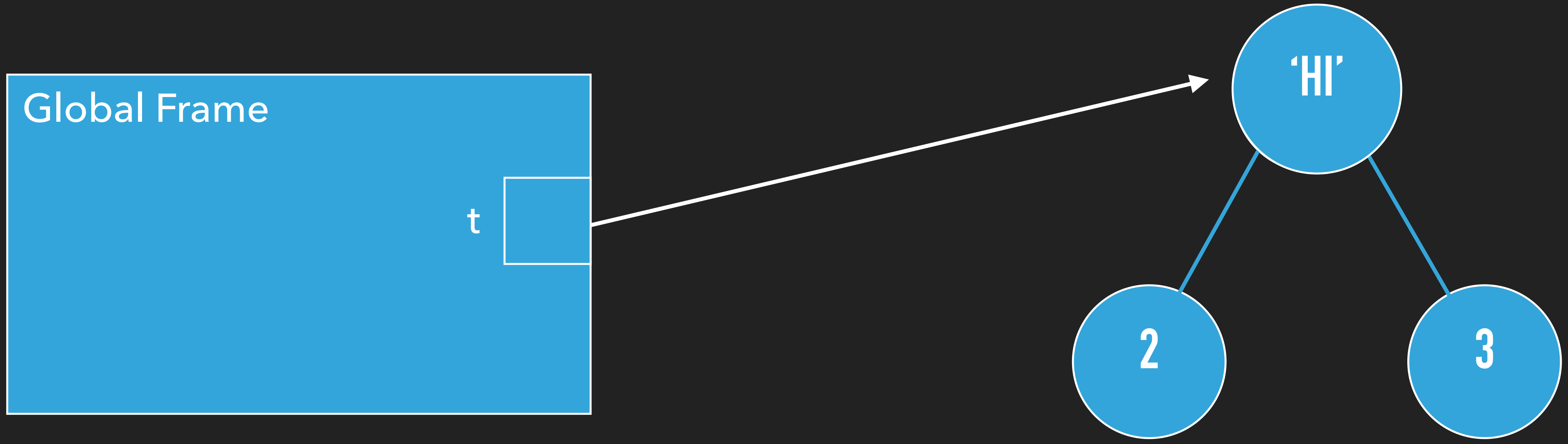
SO WHAT'S CHANGED? #2 MUTABILITY!!



```
>>> t = Tree(1, [Tree(2), Tree(3)])

>>> t.label = 'HI'
```

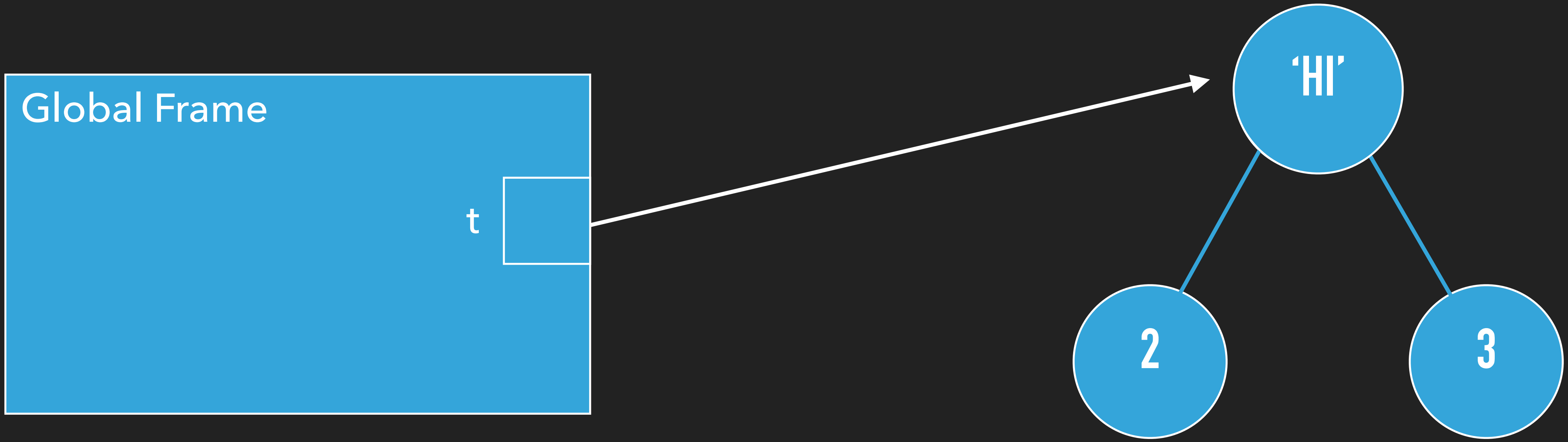
SO WHAT'S CHANGED? #2 MUTABILITY!!



```
>>> t = Tree(1, [Tree(2), Tree(3)])

>>> t.label = 'HI'
```

SO WHAT'S CHANGED? #2 MUTABILITY!!

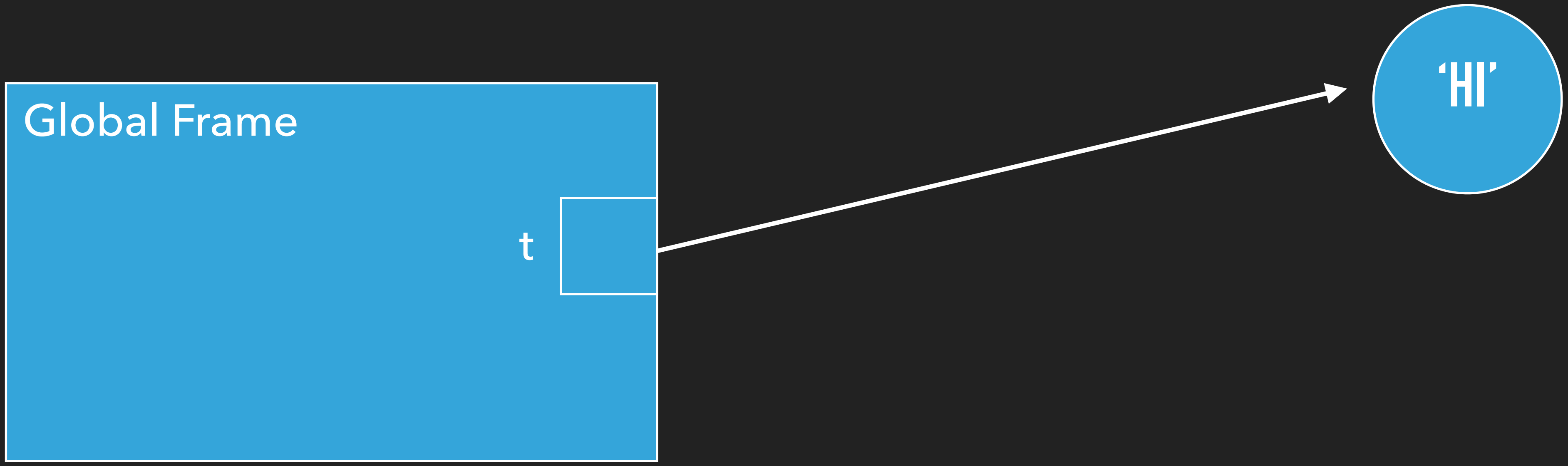


```
>>> t = Tree(1, [Tree(2), Tree(3)])

>>> t.label = 'HI'

>>> t.branches = []
```

SO WHAT'S CHANGED? #2 MUTABILITY!!



```
>>> t = Tree(1, [Tree(2), Tree(3)])

>>> t.label = 'HI'

>>> t.branches = []
```

CONSEQUENCES FOR PROBLEMS

- Recursion still your best bet to solve problems
- Now you don't always need to return values
  - If I was writing a function that squared every node in a tree mutatively, do I need a return?
  - You might hear the terms constructively and destructively to refer to creating a new tree, or modifying the input tree

SOME INTERESTING EXAM PROBLEMS

- Sp16 Final Q2
  - a) and b) are general mutability; c) is linked list – like today's discussion
- Sp16 Final Q5
  - Recursion on paths: like a child of Pascal's triangle and path_sum from lab
- Sp17 Final Q4 (Mutable trees) : conditionally destroy parts of trees.
- **Sp 17 Final Q2a) and b) for fun with linked lists**
  - What does odd/even dependence do for recursion?
- Su16 Midterm Q6) (Linked lists)
  - Classic recursion structure, have this pattern down–relevant outside of linked lists
- Fa16 Midterm 2 Q5
  - Combines Linked Lists, Trees, and Path problems; everyone's favorites trio!
  - Google the **any** function or **True in lst** syntax if this doesn't look familiar