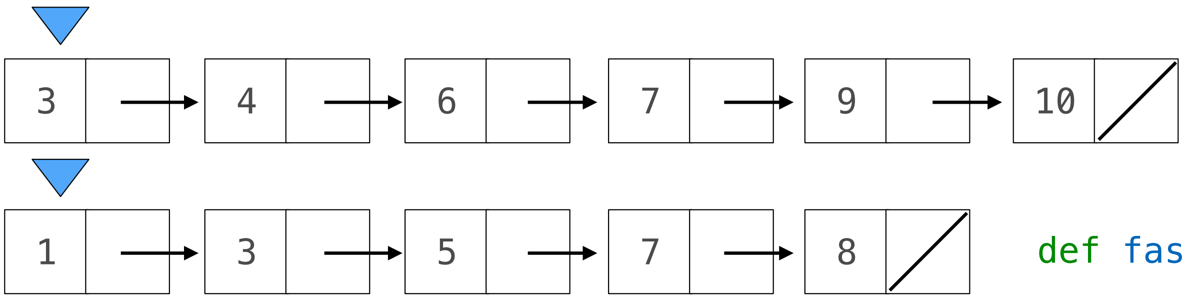
Data Examples

Announcements

Discussion 8

Linear-Time Intersection of Sorted Linked Lists

Given two sorted **linked lists** with no repeats, return the number of elements that appear in both.




```
def fast_overlap(s, t):
    if s is Link.empty or t is Link.empty:
        return 0
    if s.first == t.first:
        return 1 + fast_overlap(s.rest, t.rest)
    elif s.first < t.first:
        return fast_overlap(s.rest, t)
    elif s.first > t.first:
        return fast_overlap(s, t.rest)
```

```
def fast_overlap(s, t):
    k = 0
    while s and t:
        if s.first == t.first:
            k, s, t, = k + 1, s.rest, t.rest
        elif s.first < t.first:
            s = s.rest
        elif s.first > t.first:
            t = t.rest
    return k
```

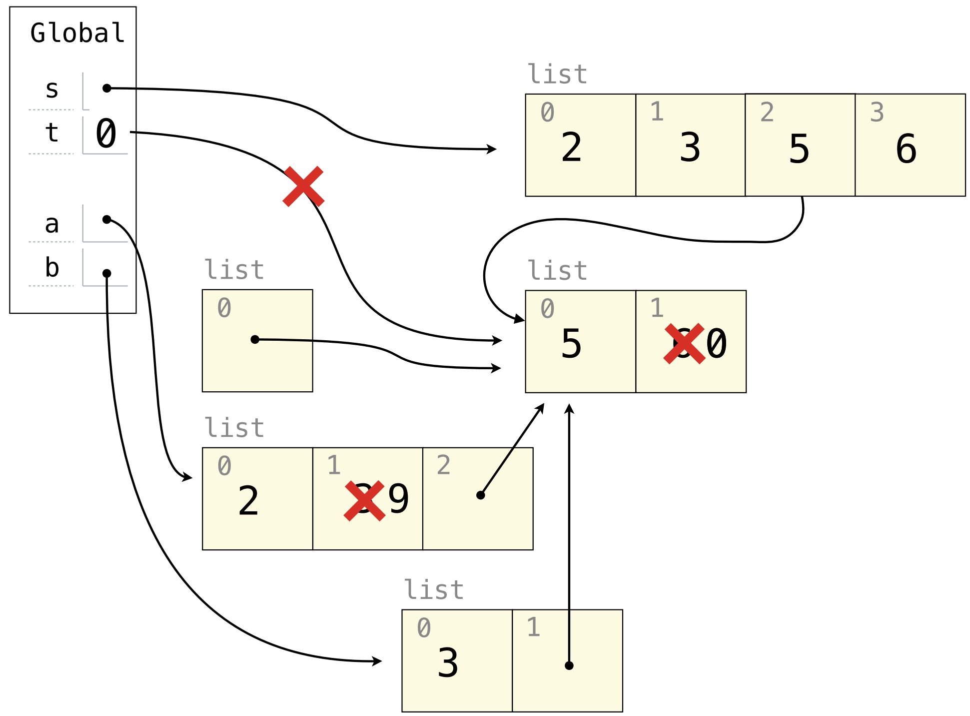
Lists

Lists in Environment Diagrams

Assume that before each example below we execute:

```
s = [2, 3]
t = [5, 6]
```

Operation	Example	Result
append adds one element to a list	s.append(t) t = 0	s → [2, 3, [5, 6]] t → 0
extend adds all elements in one list to another list	s.extend(t) t[1] = 0	s → [2, 3, 5, 6] t → [5, 0]
addition & **slicing** create new lists containing existing elements	a = s + [t] b = a[1:] a[1] = 9 b[1][1] = 0	s → [2, 3] t → [5, 0] a → [2, 9, [5, 0]] b → [3, [5, 0]]

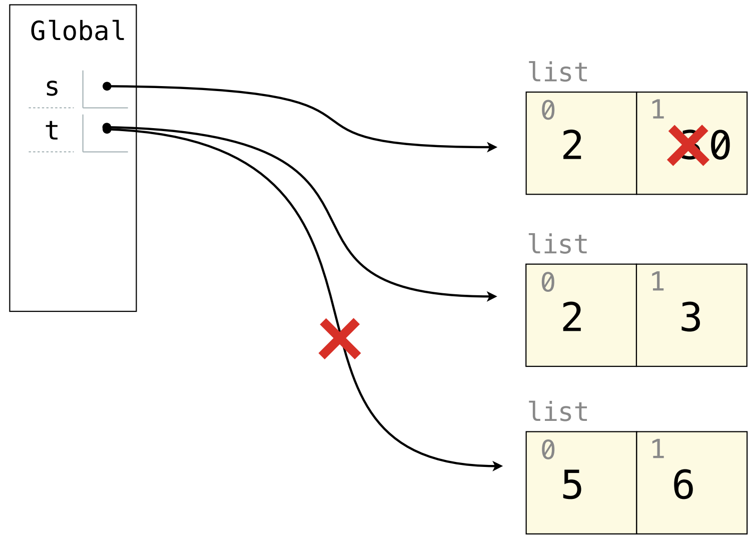


Lists in Environment Diagrams

Assume that before each example below we execute:

```
s = [2, 3]
t = [5, 6]
```

Operation	Example	Result
append adds one element to a list	s.append(t) t = 0	s → [2, 3, [5, 6]] t → 0
extend adds all elements in one list to another list	s.extend(t) t[1] = 0	s → [2, 3, 5, 6] t → [5, 0]
addition & **slicing** create new lists containing existing elements	a = s + [t] b = a[1:] a[1] = 9 b[1][1] = 0	s → [2, 3] t → [5, 0] a → [2, 9, [5, 0]] b → [3, [5, 0]]
The **list** function also creates a new list containing existing elements	t = list(s) s[1] = 0	s → [2, 0] t → [2, 3]

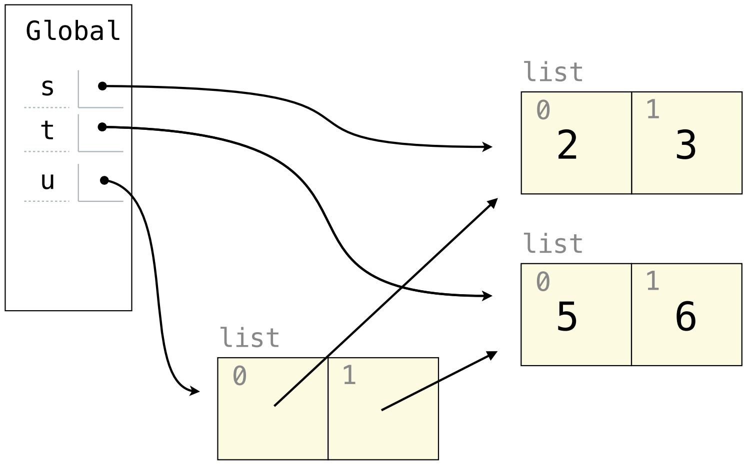


Lists in Environment Diagrams

Assume that before each example below we execute:

```
s = [2, 3]
t = [5, 6]
```

Operation	Example	Result
append adds one element to a list	s.append(t) t = 0	s → [2, 3, [5, 6]] t → 0
extend adds all elements in one list to another list	s.extend(t) t[1] = 0	s → [2, 3, 5, 6] t → [5, 0]
addition & **slicing** create new lists containing existing elements	a = s + [t] b = a[1:] a[1] = 9 b[1][1] = 0	s → [2, 3] t → [5, 0] a → [2, 9, [5, 0]] b → [3, [5, 0]]
The **list** function also creates a new list containing existing elements	t = list(s) s[1] = 0	s → [2, 0] t → [2, 3]
[...] creates a new list	u = [s, t]	s → [2, 3] t → [5, 6] u → [[2, 3], [5, 6]]



Lists in Environment Diagrams

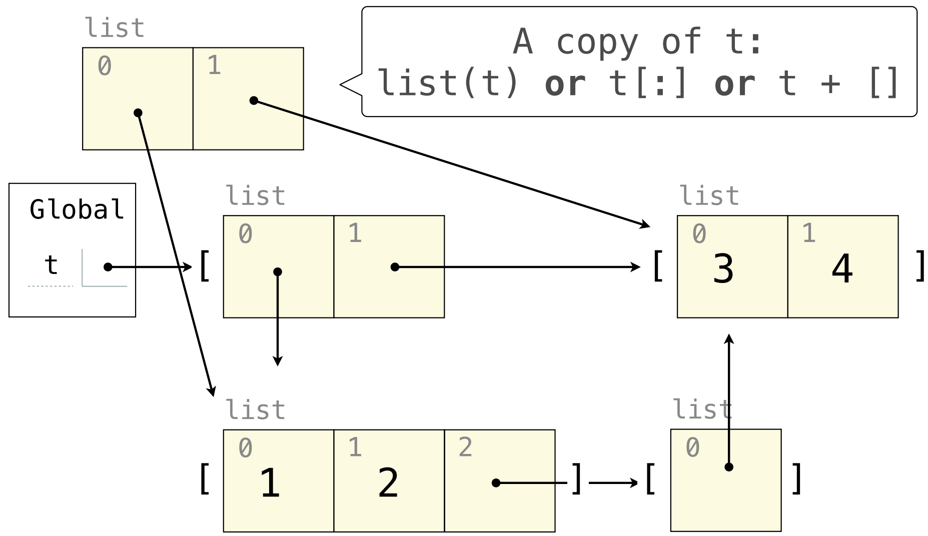
Assume that before each example below we execute:

```
s = [2, 3]
t = [5, 6]
```

Operation	Example	Result
pop removes & returns the last element	t = s.pop()	s → [2] t → 3
remove removes the first element equal to the argument	t.extend(t) t.remove(5)	s → [2, 3] t → [6, 5, 6]

Lists in Lists in Lists in Environment Diagrams

```
t = [[1, 2], [3, 4]]
list(t)
t[0].append(t[1:2])
print(t)
```




```
[[1, 2, [[3, 4]]], [3, 4]]
```

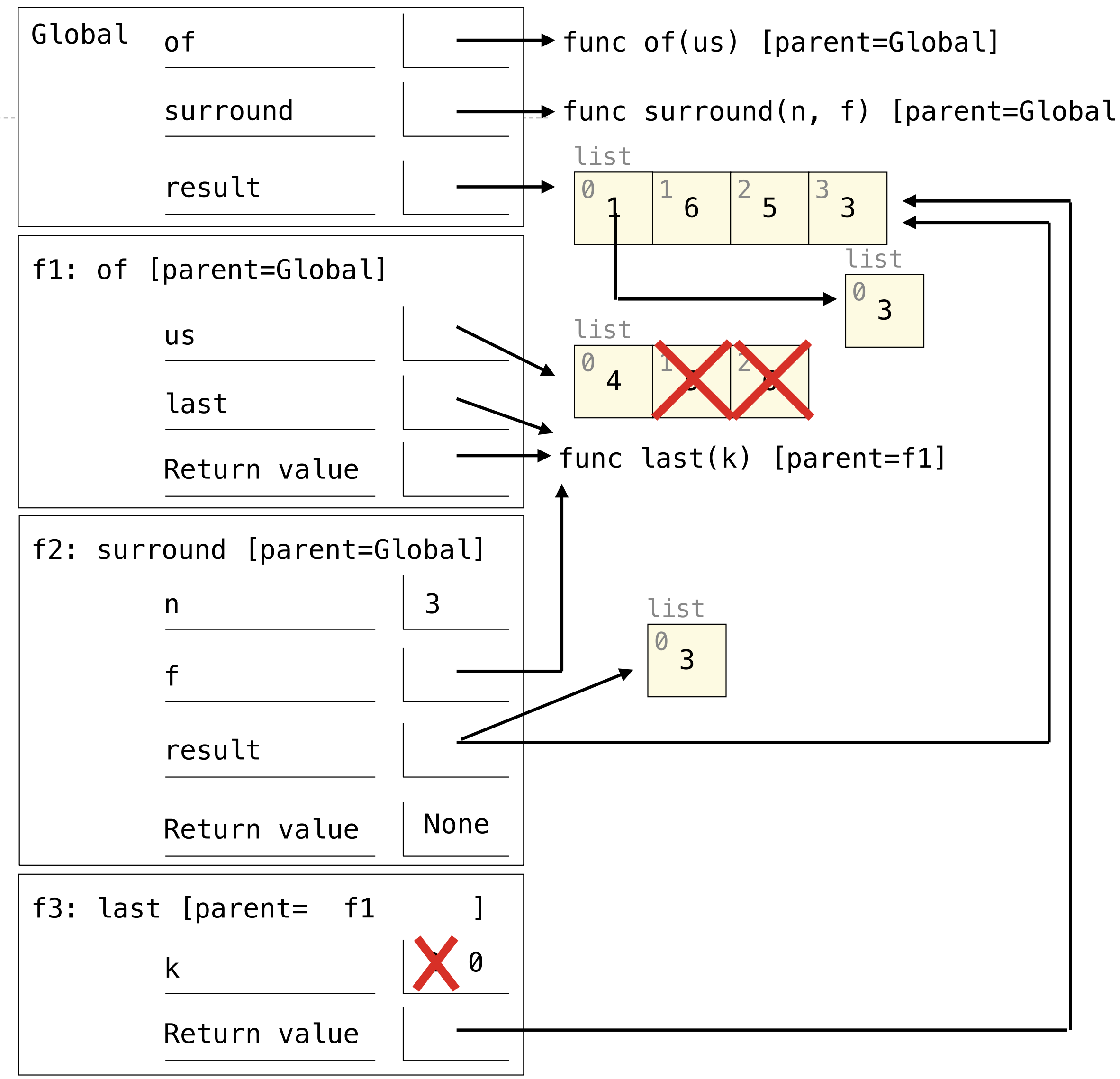
Fall 2022 Midterm 2 Question 2

```python
def of(us):
    def last(k):
        "The last k items of us"
        while k > 0:
            result.append(us.pop())
            k = k - 1
        return result
    return last

def surround(n, f):
    "n is the first and last item of f(2)"
    result = [n]
    result = f(2)
    result[0] = [n]
    return result.append(n)

result = [1]
surround(3, of([4, 5, 6]))
print(result)
```

[[3], 6, 5, 3]



Trees

Heracles, Iolaus and the Hydra, Paestan black-figure hydra C6th B.C., The J. Paul Getty Museum

Fall 2022 Midterm 2 Question 4(b)

A *hydra* is a Tree with a special structure. Each node has 0 or 2 children. All leaves are heads labeled 1. Each non-leaf body node is labeled with the number of leaves among its descendants.

Implement **chop_head(hydra, n)**, which takes a hydra and a positive integer n. It mutates hydra by chopping off the nth head from the left, which adds two new adjacent heads in its place. Update all ancestor labels.

```
def chop_head(hydra,  n):
    assert n > 0 and n <= hydra.label
    if hydra.is_leaf():
        hydra.label = 2
        hydra.branches = [Tree(1),  Tree(1)]
    else:
        hydra.label += 1
        left, right = hydra.branches
        if n > left.label :
            chop_head(right, n - left.label )
        else:
            chop_head(left, n)
```