

## 61A Lecture 22

---

## Announcements

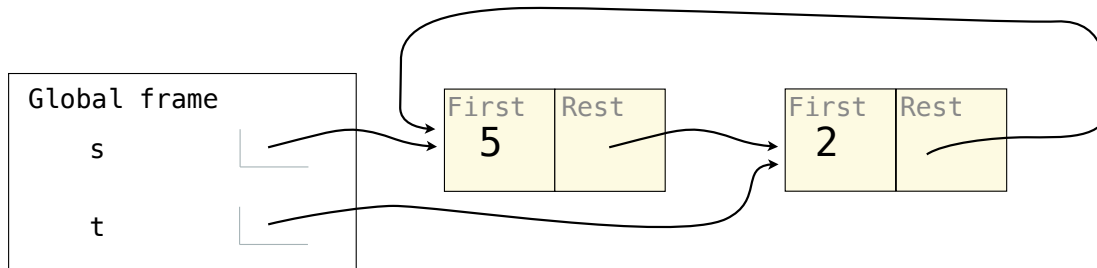
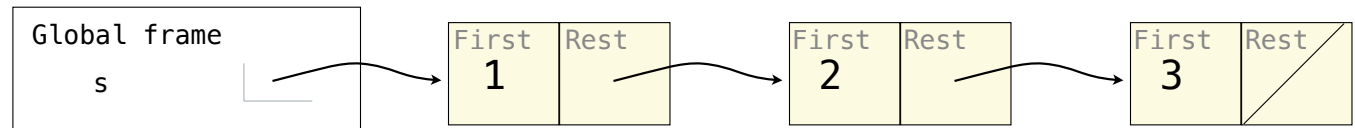
# Linked Lists

## Recursive Lists Can Change

Attribute assignment statements can change first and rest attributes of a Link

The rest of a linked list can contain the linked list as a sub-list

```
>>> s = Link(1, Link(2, Link(3)))
>>> s.first = 5
>>> t = s.rest
>>> t.rest = s
>>> s.first
5
>>> s.rest.rest.rest.rest.first
2
```

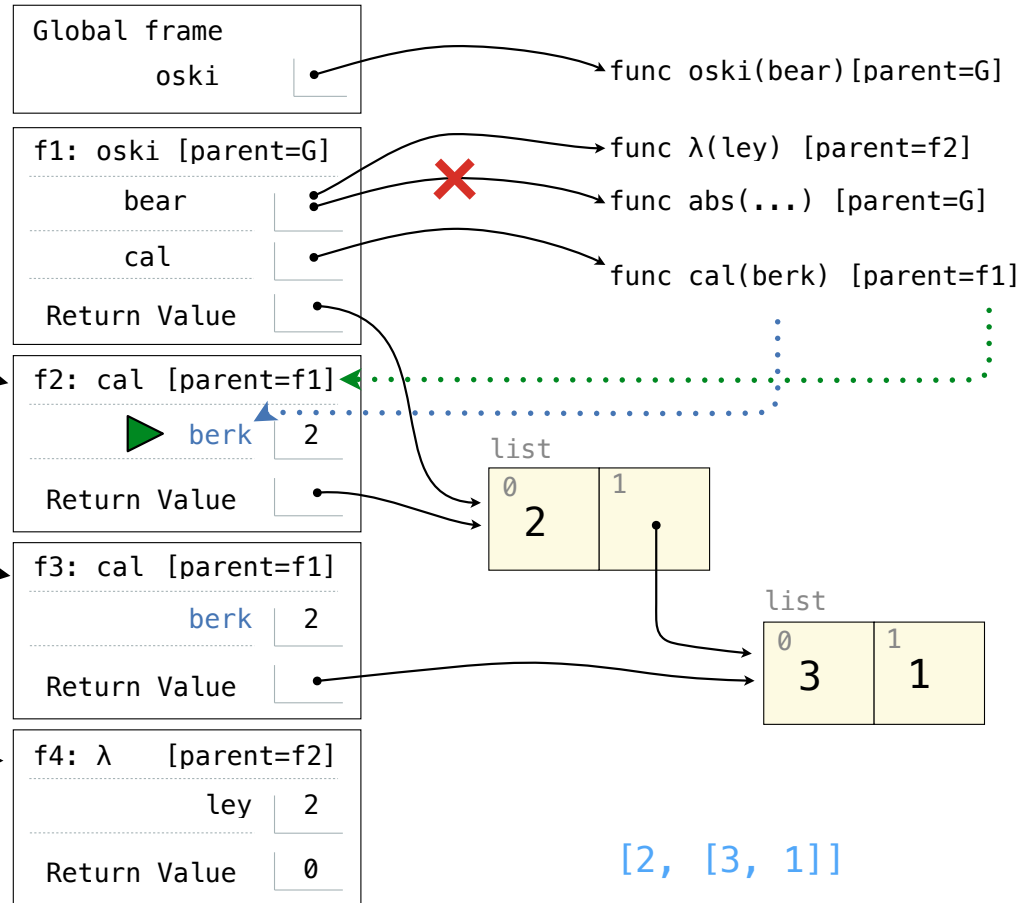


Note: The actual environment diagram is much more complicated.

# Environment Diagrams

## Go Bears!

```
def oski(bear):  
    def cal(berk):  
        nonlocal bear  
        if bear(berk) == 0:  
            return [berk+1, berk-1]  
        bear = lambda ley: berk-ley  
        return [berk, cal(berk)]  
    return cal(2)  
oski(abs)
```



Objects

## Land Owners

Instance attributes are found before class attributes; class attributes are inherited

```
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting
```

```
class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'
```

```
jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
```

```
>>> Worker().work()
'Sir, I work'
```

```
>>> jack
Peon
```

```
>>> jack.work()
'Maam, I work'
```

```
>>> john.work()
Peon, I work
'I gather wealth'
```

```
>>> john.elf.work(john)
'Peon, I work'
```

<class Worker>

greeting: 'Sir'

<class Bourgeoisie>

greeting: 'Peon'

jack <Worker>

elf: \_\_\_\_\_  
greeting: 'Maam'

john <Bourgeoisie>

elf: \_\_\_\_\_



Trees

## Morse Code

Morse code is a signaling protocol that transmits messages by sequences of signals

*Problem:* Implement `morse` so that `decode` works correctly

```
abcde = {'a': '.-', 'b': '-...', 'c': '-.-.', 'd': '-..', 'e': '.'}
```

```
def decode(signals, tree):  
    """Decode signals into a letter.  
    >>> t = morse(abcde)  
    >>> [decode(s, t) for s in ['-..', '.', '-.-.', '.-', '-..', '.']]  
    ['d', 'e', 'c', 'a', 'd', 'e']  
    """  
    for signal in signals:  
        tree = [b for b in tree.branches if b.root == signal][0]  
    leaves = [b for b in tree.branches if b.is_leaf()]  
    assert len(leaves) == 1  
    return leaves[0].root
```

```
def morse(code):  
    ....
```

```
decode('.', t)
```

(Demo)

A: ● ■  
B: ■ ● ● ●  
C: ■ ● ■ ●  
D: ■ ● ●  
E: ●  
...

