

Лекция 22

Перевод и адаптация материалов Джона ДеНиро (John DeNero). Используется с разрешения автора.

Списки

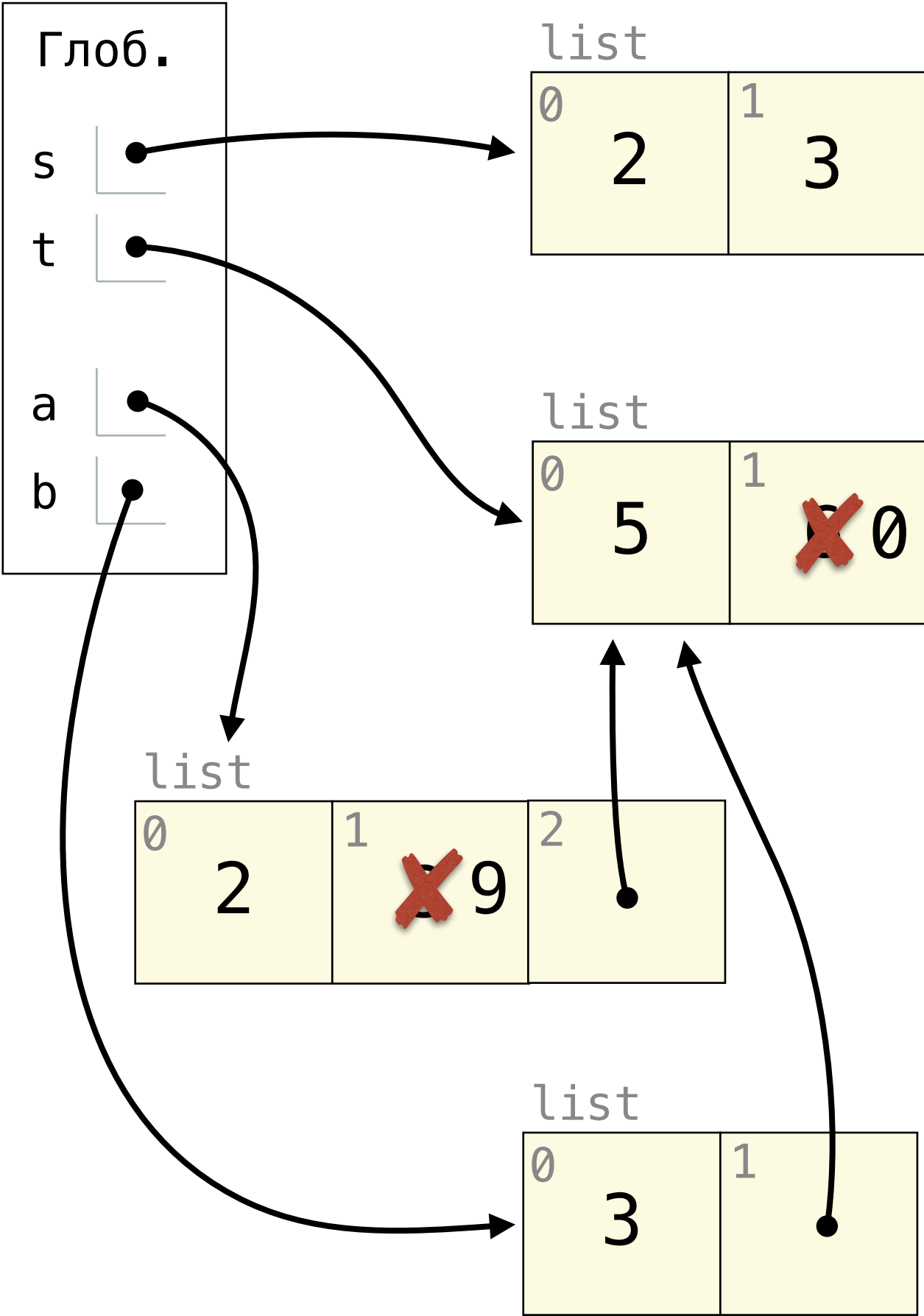
Списки на диаграммах окружения

Представь, что перед каждым примером выполняется вот что:

s = [2, 3]

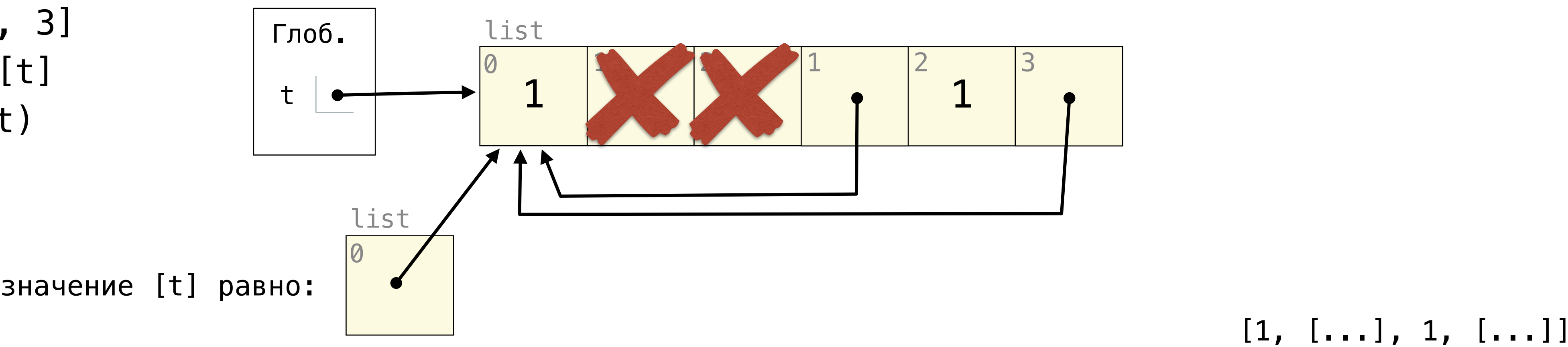
t = [5, 6]

Действие	Пример	Результат
append добавляет элемент в список	s.append(t) t = 0	s → [2, 3, [5, 6]] t → 0
extend добавляет все элементы одного списка в другой	s.extend(t) t[1] = 0	s → [2, 3, 5, 6] t → [5, 0]
Срезы и сложение создают новые списки содержащие существующие элементы	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]]
Функция list создаёт новый список, содержащий исходные элементы	t = list(s) s[1] = 0	s → [2, 0] t → [2, 3]
Присвоение срезу заменяет срез новыми значениями	s[0:0] = t s[3:] = t t[1] = 0	s → [5, 6, 2, 5, 6] t → [5, 0]

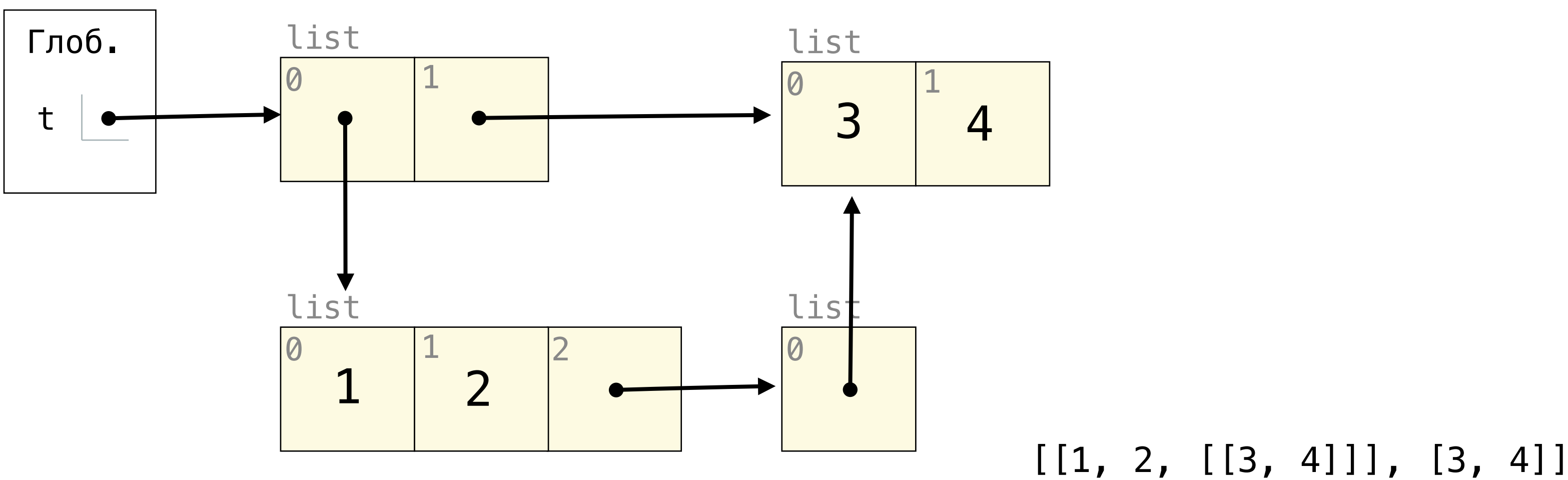


Списки в списках на диаграммах окружения

```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```



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



Объекты

Лэндлорды-с

Поиск имён сначала происходит в атрибутах экземпляра, затем в атрибутах класса; атрибуты класса наследуются.

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

```
class Bourgeoisie(Worker):
    greeting = 'Работник'
    def work(self):
        print(Worker.work(self))
        return 'Я богатею'
```

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

```
>>> Worker().work()
'Сэр, я работаю'
```

```
>>> jack
'Работник'
```

```
>>> jack.work()
'Мэм, я работаю'
```

```
>>> john.work()
Работник, я работаю
'Я богатею'
```

```
>>> john.elf.work(john)
'Работник, я работаю'
```

<class Worker>

greeting: 'Сэр'

<class Bourgeoisie>

greeting: 'Работник'

jack <Worker>

elf: ●
greeting: 'Мэм'

john <Bourgeoisie>

elf: ●

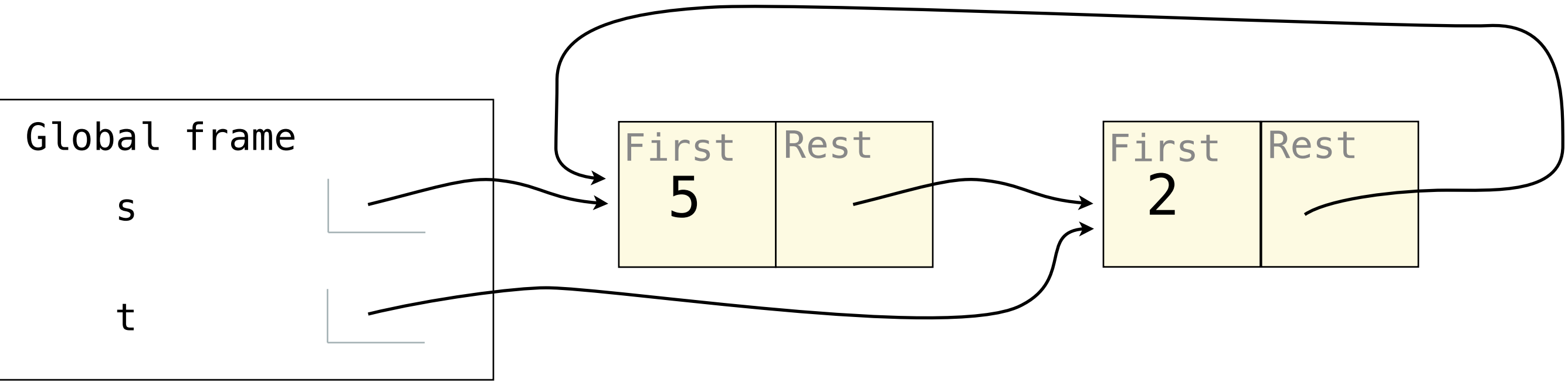
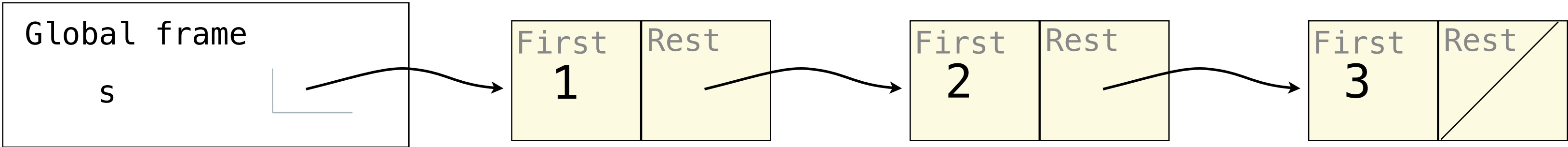
Связные списки

Рекурсивные списки могут изменяться

Инструкции присвоения атрибутам могут изменять атрибуты first и rest экземпляра класса Link.

Атрибут rest связанного списка может содержать связный список в качестве подписка.

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



Внимание: реальная диаграмма окружения куда как более сложна.

Деревья

Код Морзе

Код Морзе – способ кодирования – представление символов последовательностью сигналов: длинных («тире») и коротких («точек»).

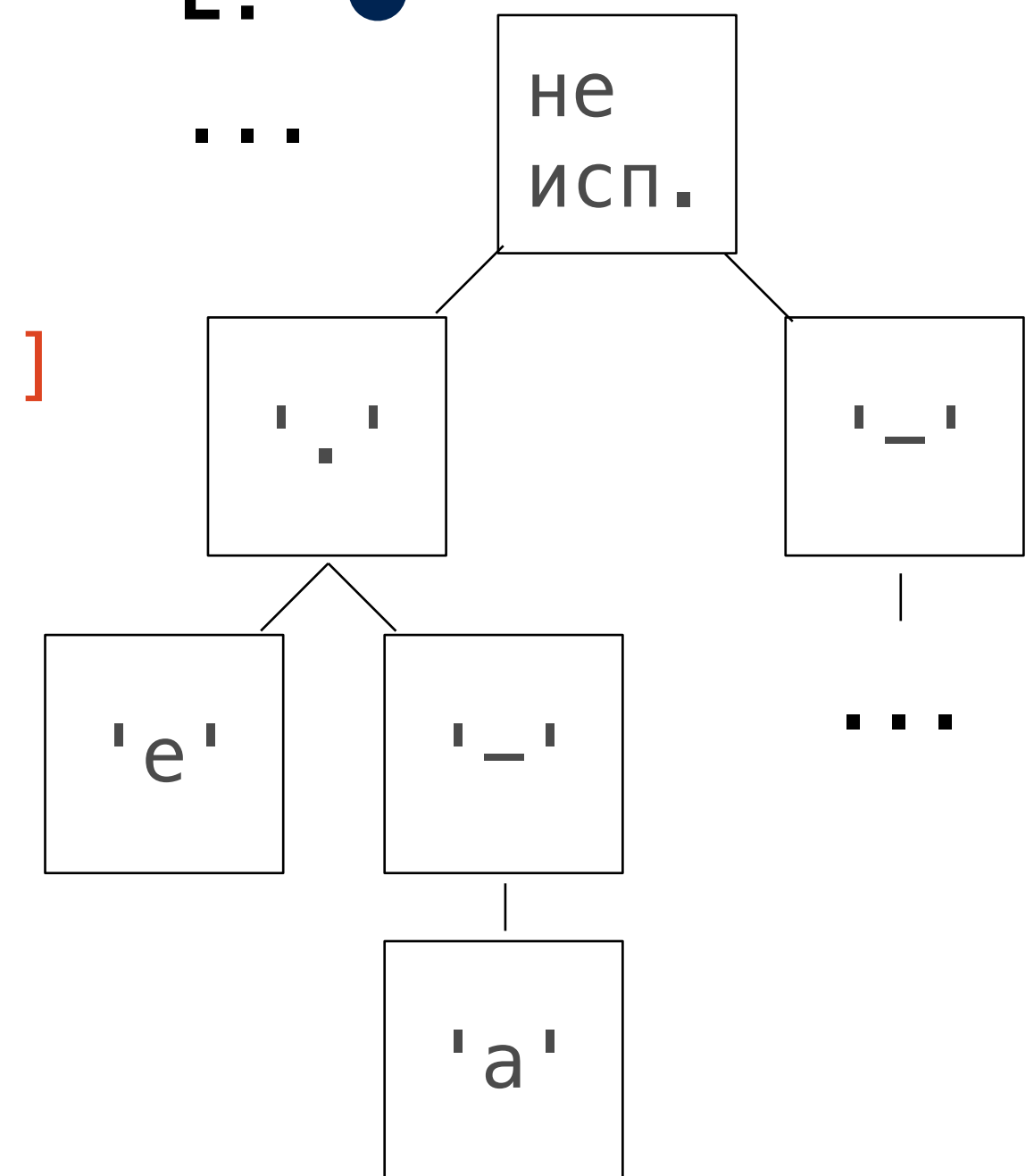
Задача: Создать функцию **morse**, чтобы **decode** верно работала

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

```
def decode(signals, tree):  
    """Декодирование сигналов в букву.  
    >>> 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.label == signal][0]  
    leaves = [b for b in tree.branches if b.is_leaf()]  
    assert len(leaves) == 1  
    return leaves[0].label
```

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

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



Вперед!

```
def dead(bolt):
    def any(stud):
        nonlocal bolt
        if bolt(stud) == 0:
            return [stud+1, stud-1]
        bolt = lambda ent: stud-ent
        return [stud, any(stud)]
    return any(2)

dead(abs)
```

