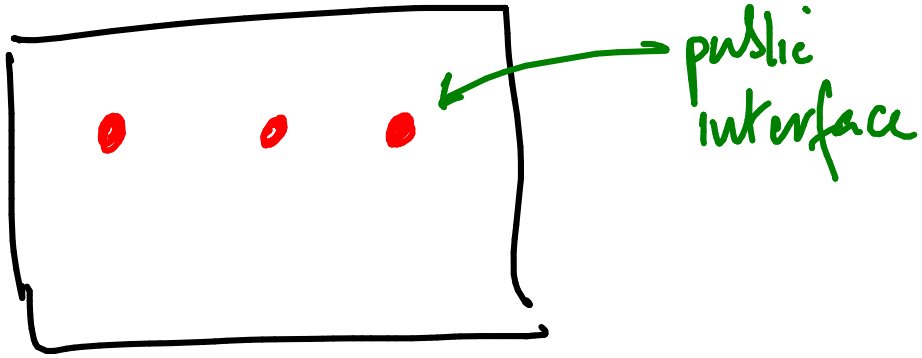
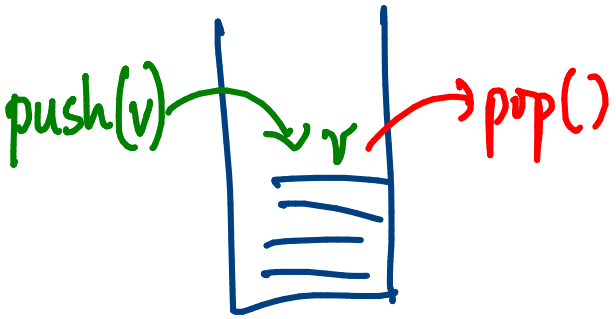
Abstract datatypes



Stack



Implement stack as list

```
def push(v):
    l.append(v)
def pop():
    v = l[-1]
    r
```

Stack as a list.

```
def push(v):
    l.append(v)
```

```
def pop():
    v = l[-1]
    del(l[-1])   # removes last elt
    return(v)
```

Public : push(v)
pop()

Should not allow l[6] = 22 etc

Creating an abstract data type

Template – "Class"

Internal variables used to store data

How public functions update/query this internal structure

Create instances of this template "Object"

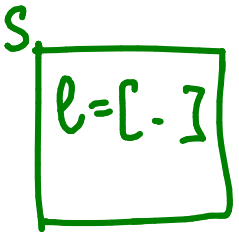
"Object Oriented Programming"

Example: Stacks

```
class Stack:
    def __init__(self):
        self.l = []

    def push(self, v):
        self.l.append(v)
    def pop(self):
        v = self.l[-1]
        del(self.l[-1])
        return(v)
```

first arg is always self

```
s = Stack()
```




```
push(s,v)
s.push(v)
```

Cannot rule out `s.l.reverse()`

Functions with optional arguments

By default, a new stack is empty
But you can initialize it at time of creation

```
def __init__(self, initval=[]):
    self.l = initval.reverse()
```

s = Stack() ⇝ s.l is []
s = Stack([4,3]) ⇝ s.l ⇝ [3,4]

Alternatively

```
def __init__(self, initval = []):
    self.l = []
    for v in initval:
        self.push(v)        # ← one function can call another

def isempty(self):
    return (self.l == [])
```

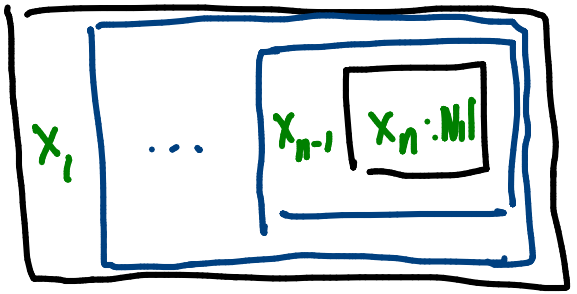
Recursive data types

A list is
- empty
- a value followed by a list

In Haskell

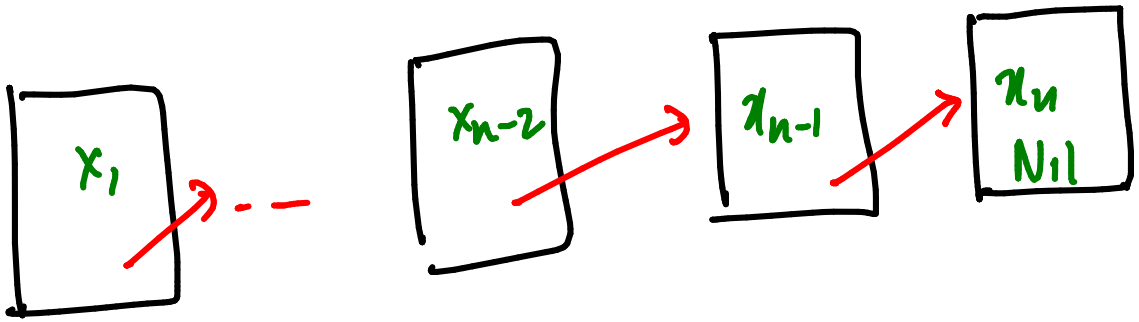
$x_1 : x_2 : \ldots : x_n : Nil$

$x_1 : (x_2 : \ldots x_{n-1} : (x_n : Nil))$



In Python.

Pull the nested boxes apart



"Linked" list

Sequence of identical boxes

Set this up as an abstract datatype

using Class

```
class Node:
    def __init__(self):
        self.value = None
        self.next = None
```

What should

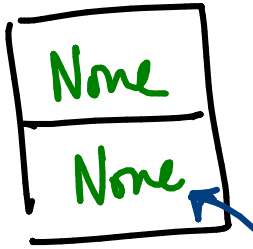
```
l = Node()
```

create ?

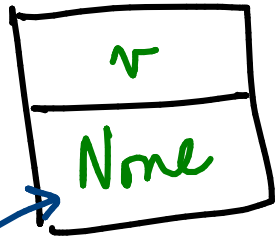
- should create empty list

last node in list

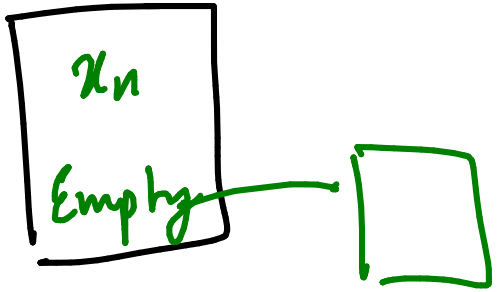
Empty list



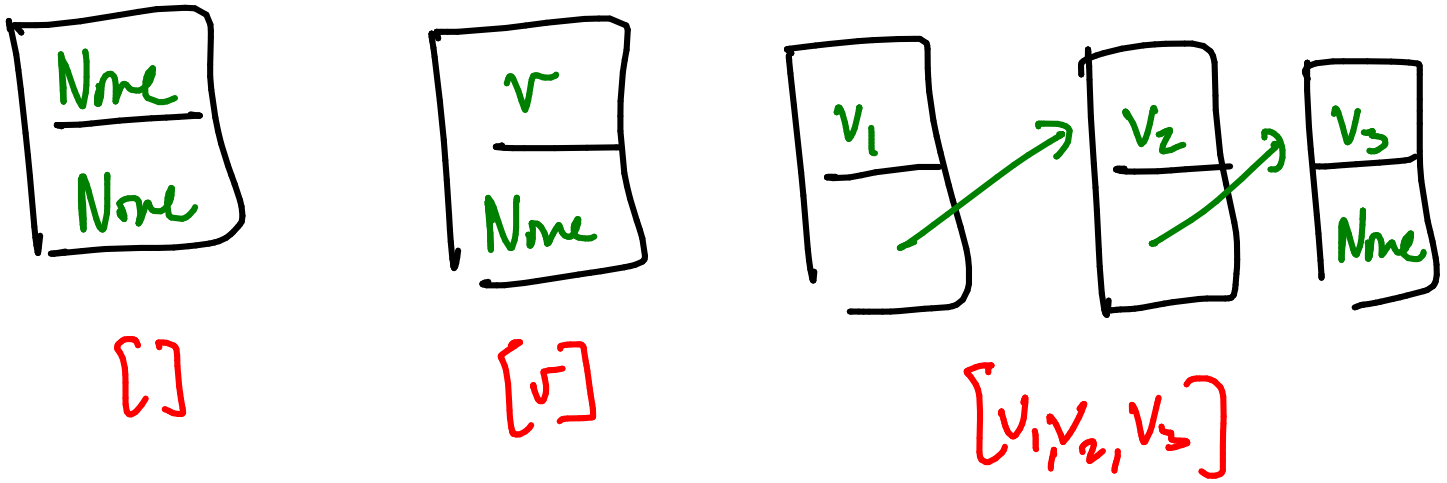
List [v]



this is the last node



Design decision:



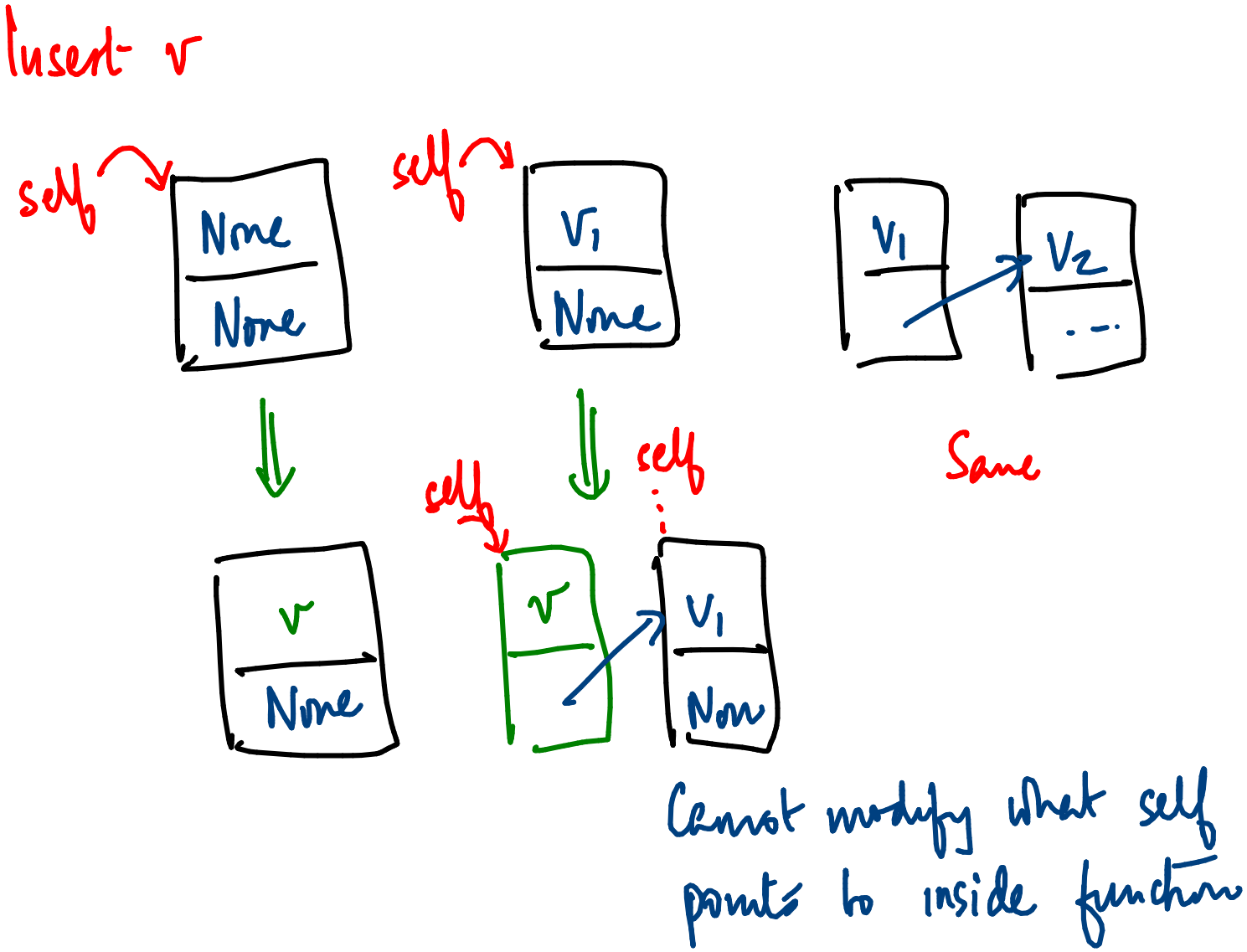
Value is never None except for []

Useful to be able to create Node with or without v

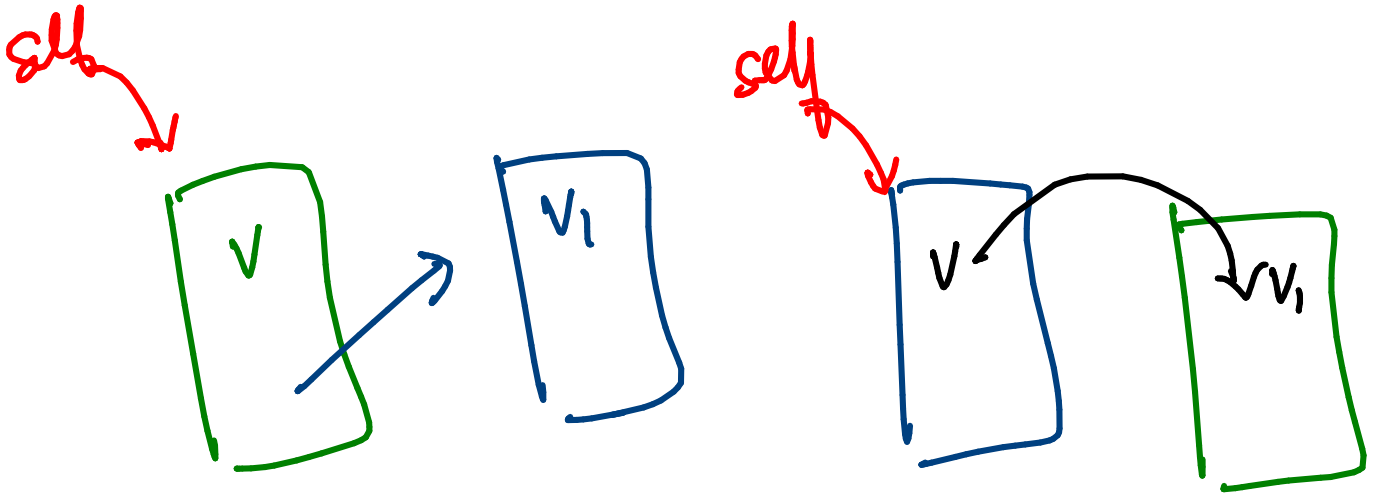
```
class Node:
    def __init__(self, initval=None):
        self.value = initval
        self.next = None

    def isempty(self):
        return (self.value == None)
```

define insert(v) — v:l

append(v) l.append(v)


insert- v
self
None
None
self
v_1
None
v_1
v_2
...
Same
self
self
v
None
v
v_1
Non
Cannot modify what self
points to inside function

Get around this?



Push new node to second position

Copy values from self to new node

Update values of self

Replumbing