# SICP

## God's Programming Book

Lecture-12 Mutable Values

# Mutable Values

Slides Adapted from cs61a of UC Berkeley

# Objects

# Objects

- Objects represent information
- They consist of data and behavior, bundled together to create abstractions
- Objects can represent things, but also properties, interactions, & processes
- A type of object is called a class; classes are first-class values in Python

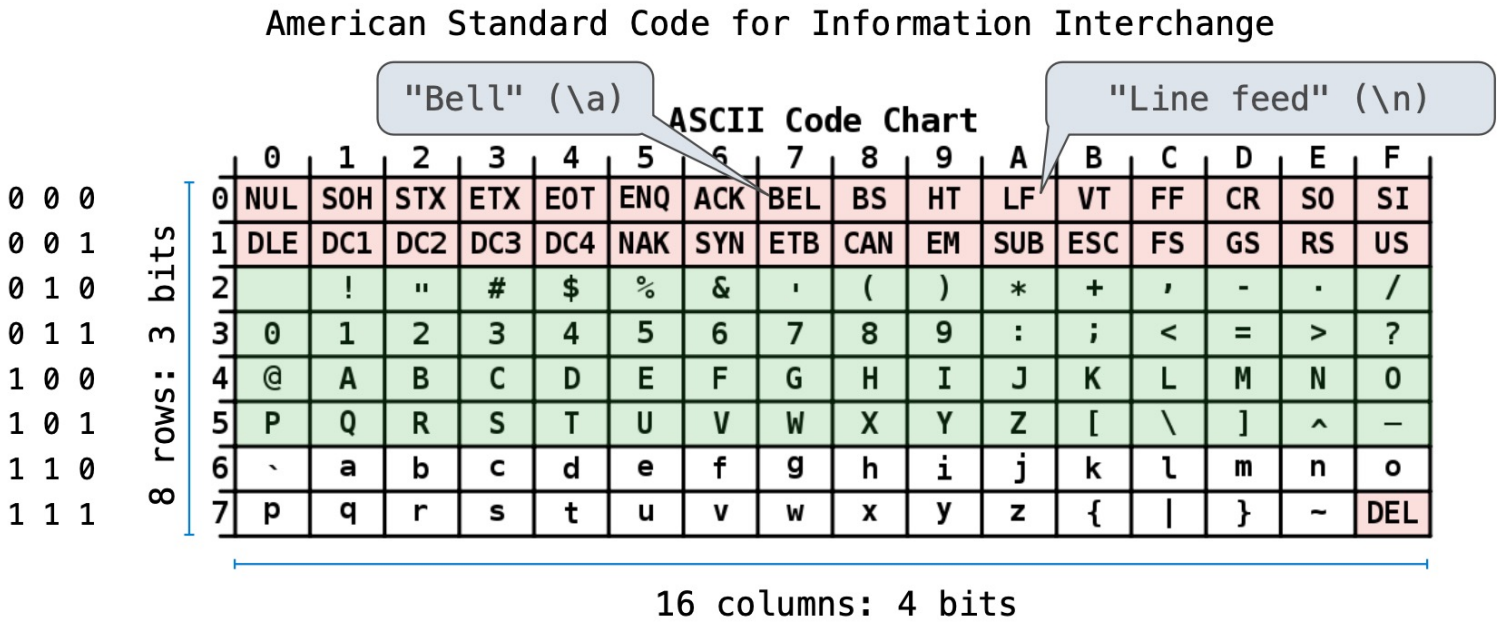
# Objects

- Object-oriented programming:
  - A metaphor for organizing large programs
  - Special syntax that can improve the composition of programs
- In Python, every value is an object
  - All objects have attributes
  - A lot of data manipulation happens through object methods
  - Functions do one thing; objects do many related things

# String

(Demo)

# Representing Strings: the ASCII Standard




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 0 | 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 0 0 1 | 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 0 1 0 | 2 | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 0 1 1 | 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 1 0 0 | 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 1 0 1 | 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 1 1 0 | 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 1 1 1 | 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | DEL |

- Layout was chosen to support sorting by character code
- Rows indexed 2-5 are a useful 6-bit (64 element) subset
- Control characters were designed for transmission

# Mutation Operations

# Some Objects Can Change

- The same object can change in value throughout the course of computation
- All names that refer to the same object are affected by a mutation
- Only objects of mutable types can change: lists & dictionaries

(Demo)

# Mutation Can Happen Within a Function Call

A function can change the value of any object in its scope

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

```
def mystery(s):
    s.pop()
    s.pop()
```

*or*

```
def mystery(s):
    s[2:] = []
```

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> another_mystery() # No arguments!
>>> len(four)
2
```

```
def another_mystery():
    four.pop()
    four.pop()
```

# Tuples

(Demo)

# Tuples are Immutable Sequences

- Immutable values are protected from mutation

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Next lecture: ooze can change turtle's binding

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Anything could be inside!']
```

# Tuples are Immutable Sequences

- The value of an expression can change because of changes in names or objects

**Name change:**

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

**Object mutation:**

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

# Tuples are Immutable Sequences

- An immutable sequence may still change if it contains a mutable value as an element

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
ERROR
```

```
>>> s = ([1, 2], 3)
>>> s[0][0] = 4
>>> s
([4, 2], 3)
```

# Mutation

# Sameness and Change

- As long as we never modify objects, a compound object is just the totality of its pieces
- A rational number is just its numerator and denominator
- This view is no longer valid in the presence of change
- A compound data object has an "identity" in addition to the pieces of which it is composed
- A list is still "the same" list even if we change its contents
- Conversely, we could have two lists that happen to have the same contents, but are different

# Sameness and Change

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a
[10, 20]
>>> b
[10, 20]
>>> a == b
True
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
>>> a
[10]
>>> b
[10, 20]
>>> a == b
False
```

# Identity Operators

---

**Identity**

`<exp0> is <exp1>`

evaluates to True if both <exp0> and <exp1> evaluate to the same object

**Equality**

`<exp0> == <exp1>`

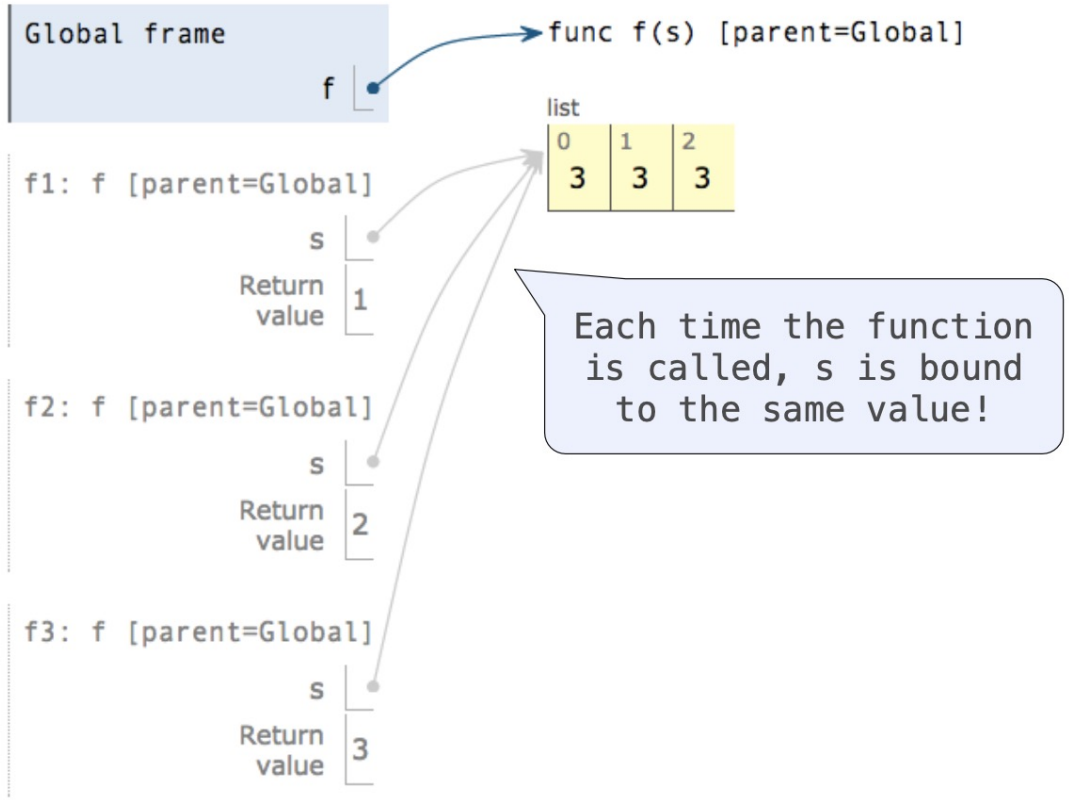
evaluates to True if both <exp0> and <exp1> evaluate to equal values

**Identical objects are always equal values**

# Mutable Default Arguments are Dangerous

A default argument value is part of a function value, not generated by a call

```
>>> def f(s=[]):
...     s.append(3)
...     return len(s)
...
>>> f()
1
>>> f()
2
>>> f()
3
```



# Lists

# Lists in Environment Diagrams

**Assume that before each example below we execute:**

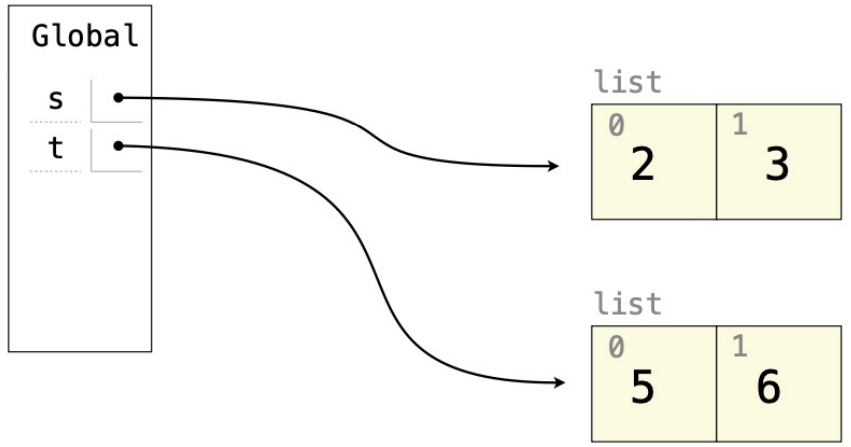
- s = [2, 3]
- t = [5, 6]

# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |

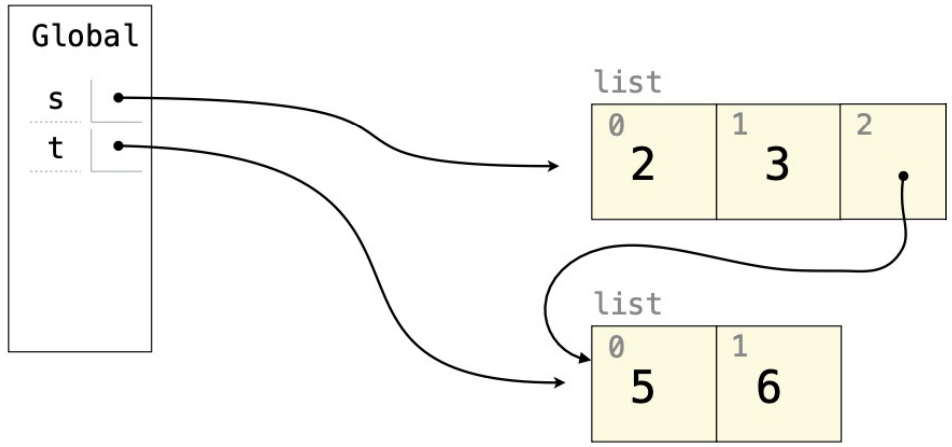
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | |



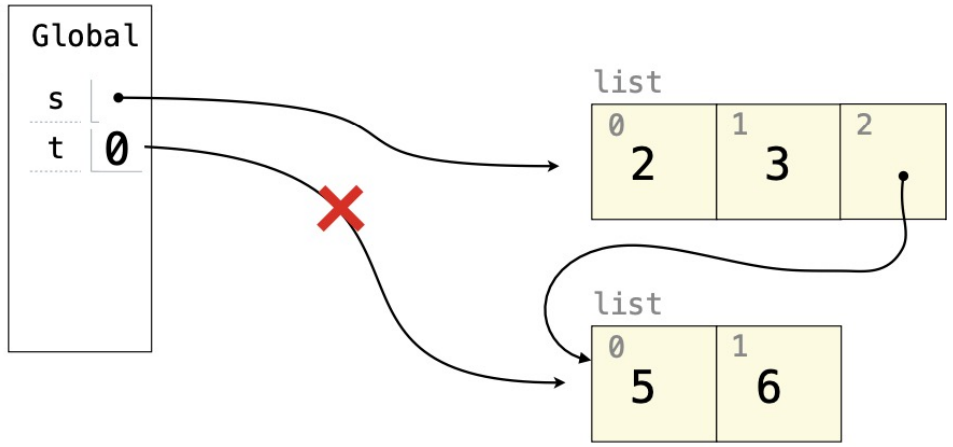
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | |



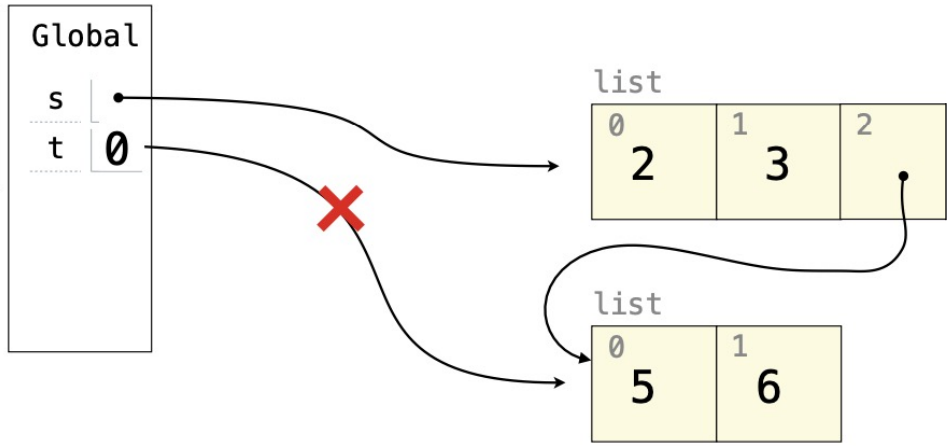
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | |



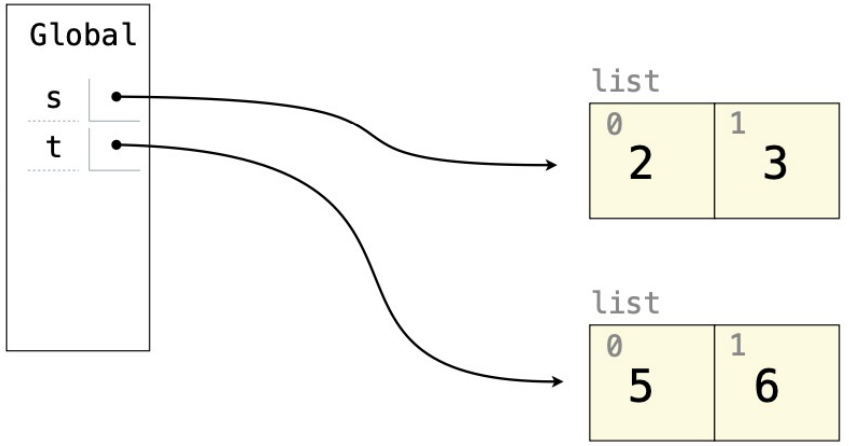
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | s → [2, 3, [5, 6]]<br>t → 0 |



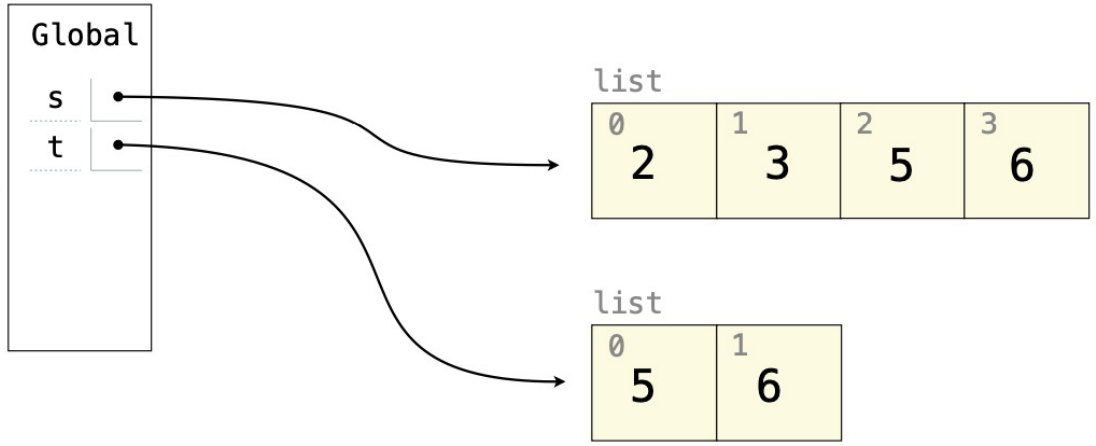
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | s → [2, 3, [5, 6]]<br>t → 0 |
| **extend** adds all elements in one list to another list | s.extend(t)<br>t[1] = 0 | |



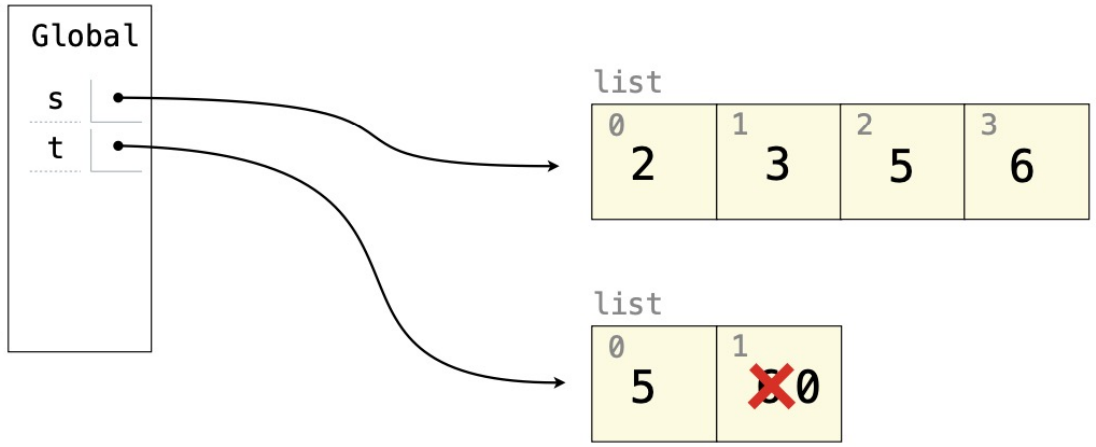
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | s → [2, 3, [5, 6]]<br>t → 0 |
| **extend** adds all elements in one list to another list | s.extend(t)<br>t[1] = 0 | |



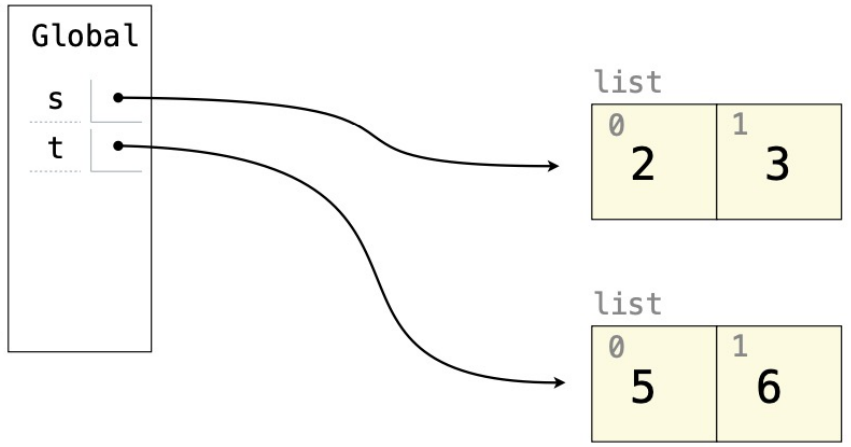
# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **append** adds one element to a list | s.append(t)<br>t = 0 | s → [2, 3, [5, 6]]<br>t → 0 |
| **extend** adds all elements in one list to another list | s.extend(t)<br>t[1] = 0 | s → [2, 3, 5, 6]<br>t → [5, 0] |



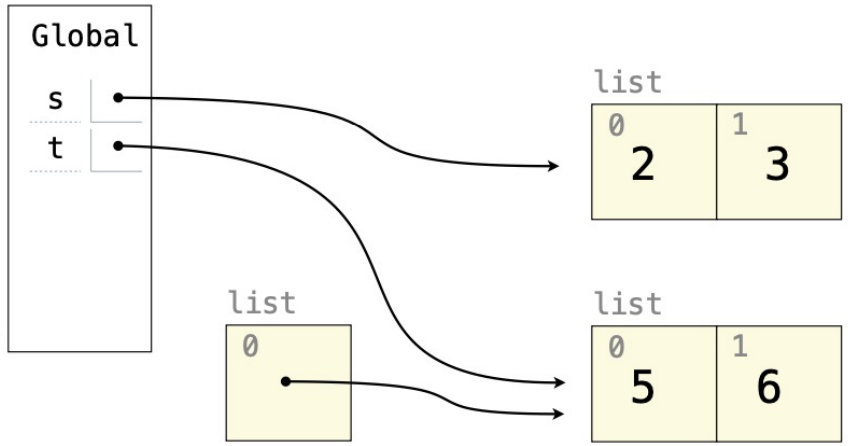
# Lists in Environment Diagrams

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



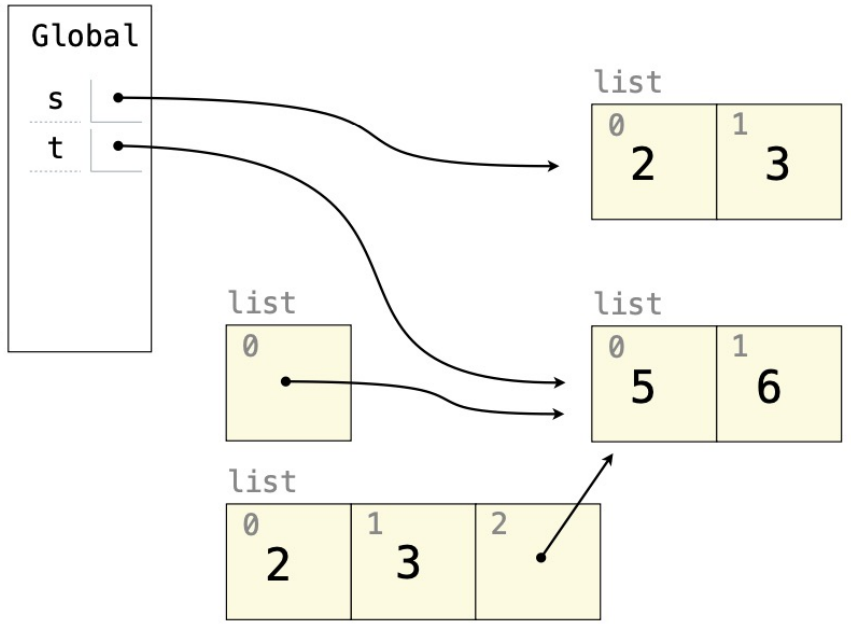
# Lists in Environment Diagrams

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



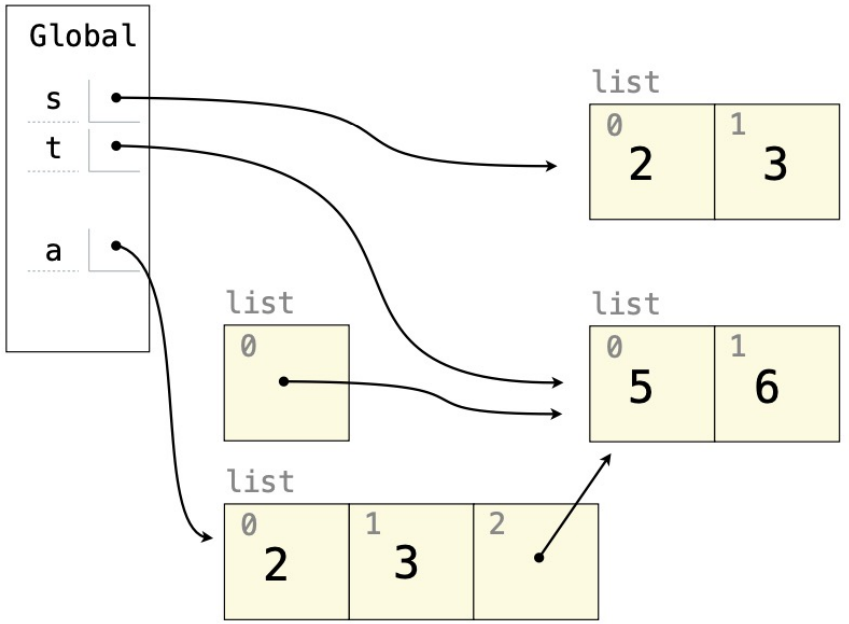
# Lists in Environment Diagrams

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



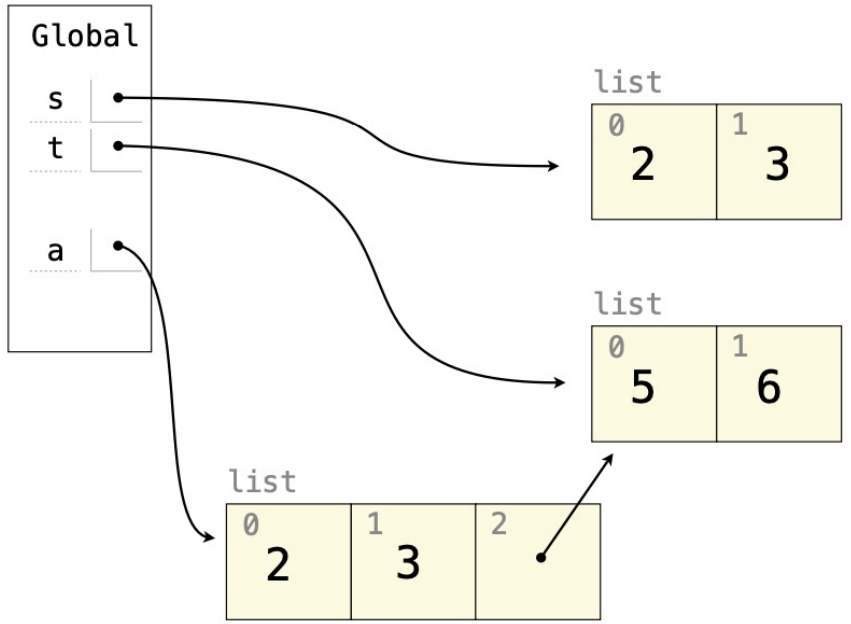
# Lists in Environment Diagrams

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



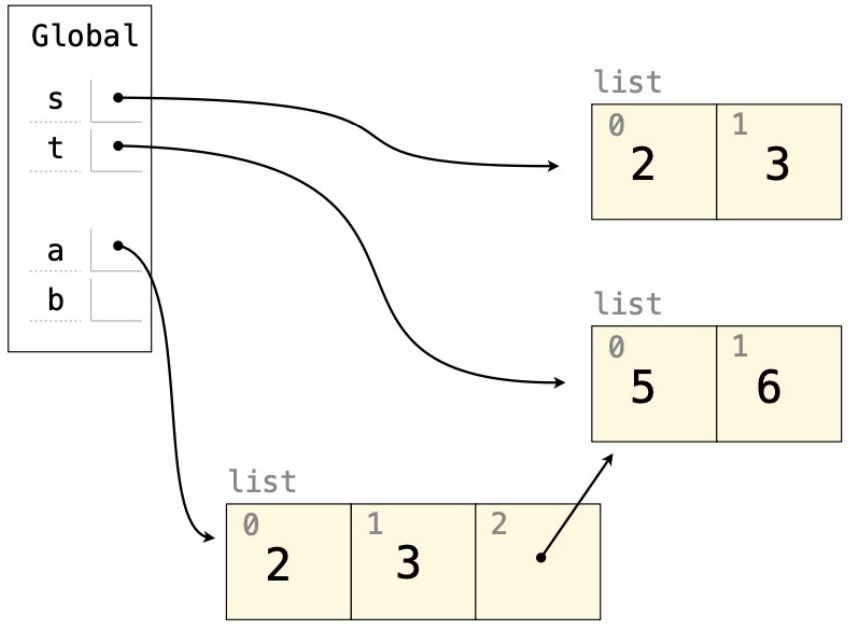
# Lists in Environment Diagrams

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



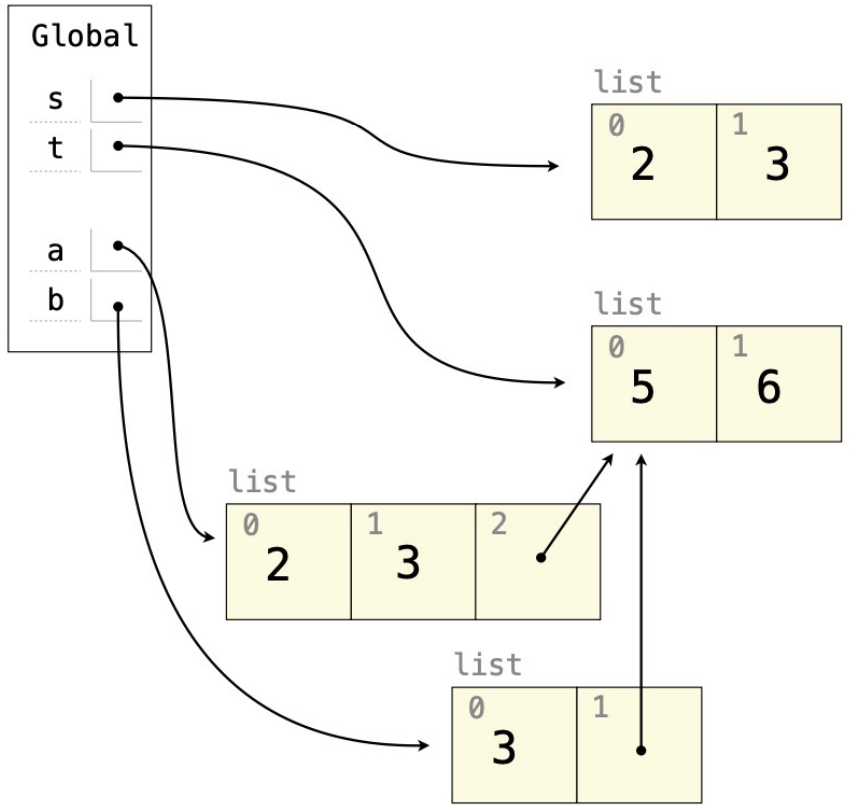
# Lists in Environment Diagrams

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



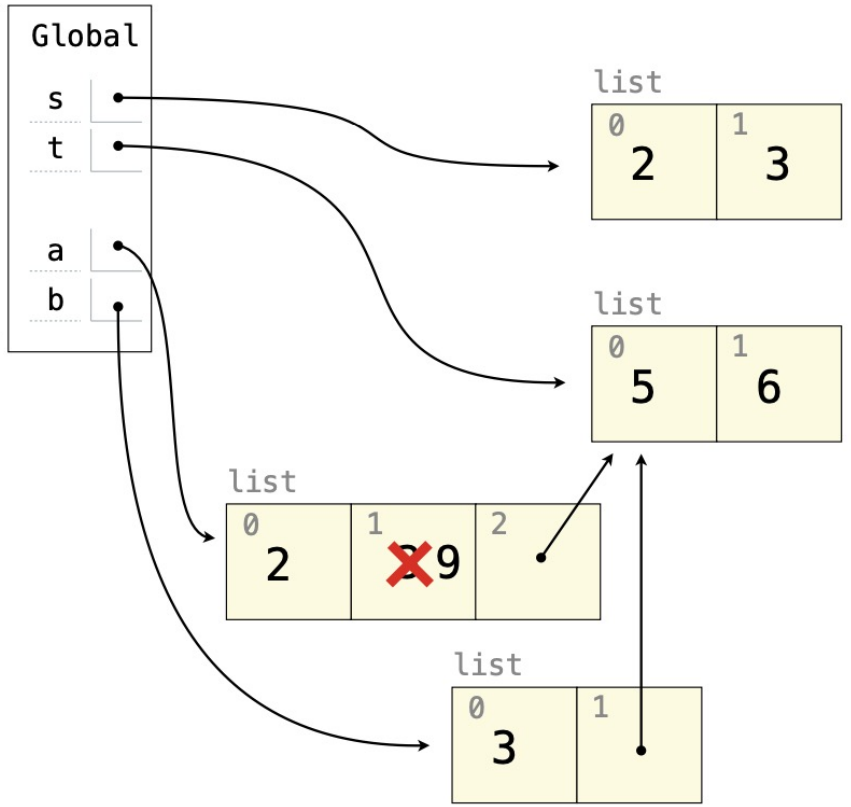
# Lists in Environment Diagrams

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



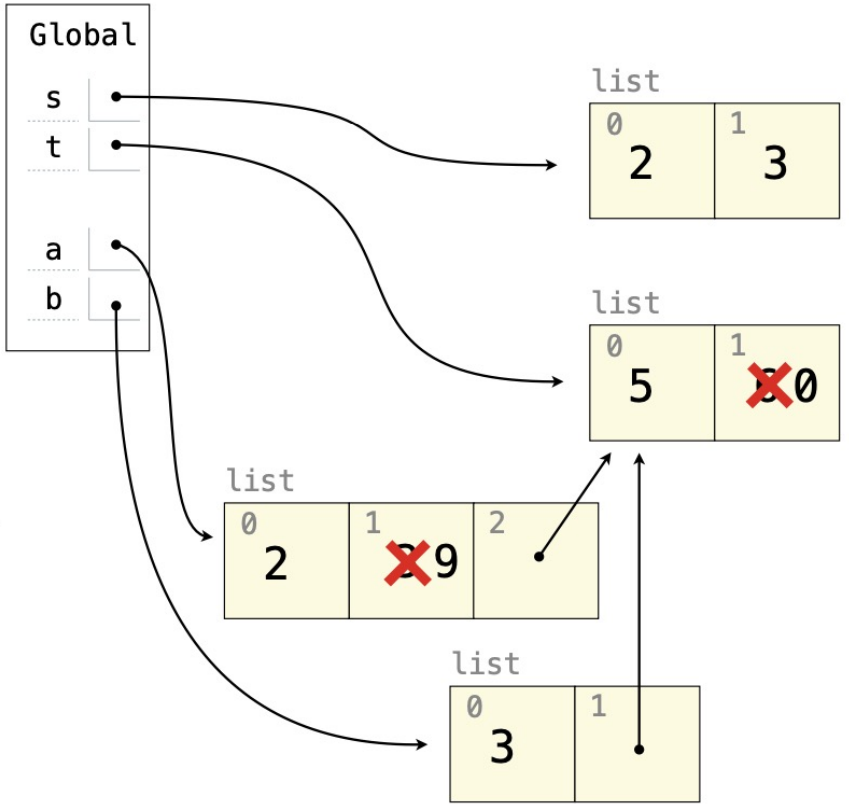
# Lists in Environment Diagrams

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



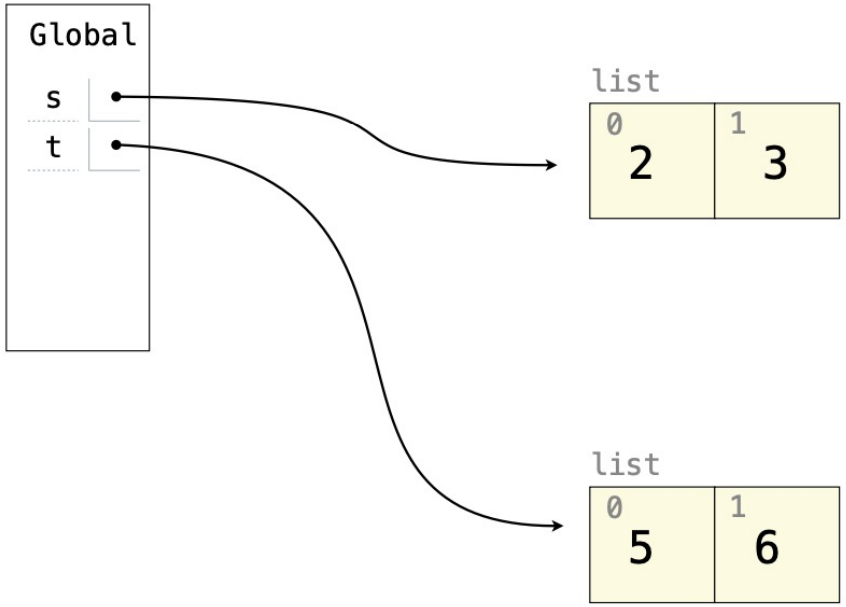
# Lists in Environment Diagrams

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



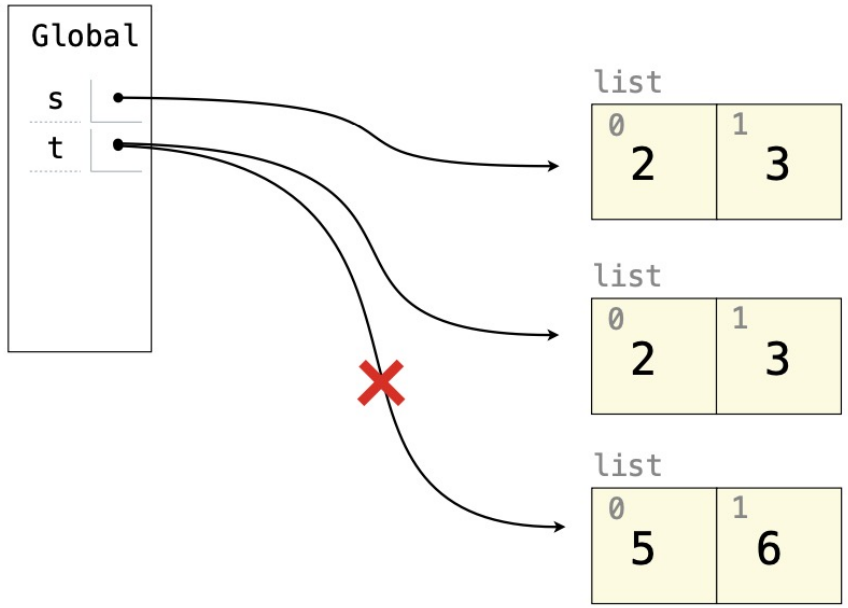
# Lists in Environment Diagrams

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



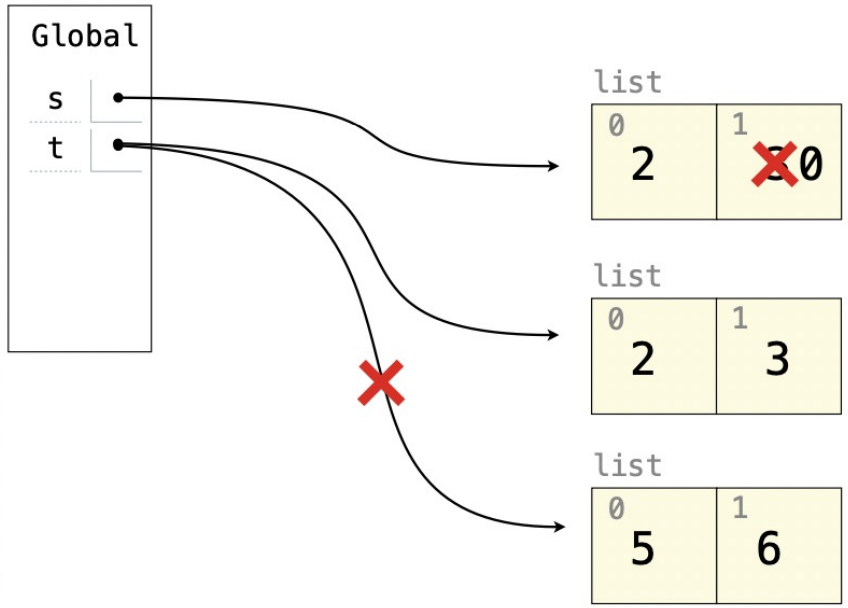
# Lists in Environment Diagrams

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



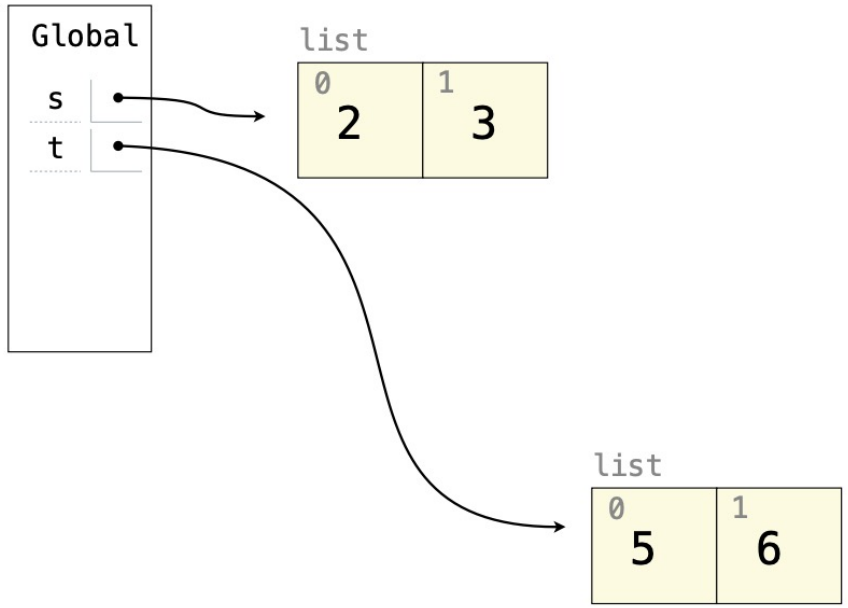
# Lists in Environment Diagrams

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



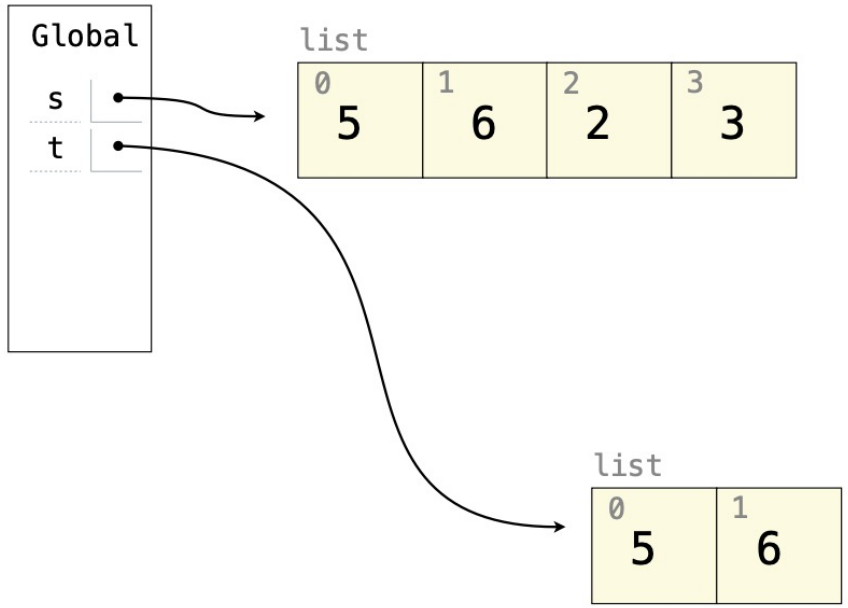
# Lists in Environment Diagrams

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



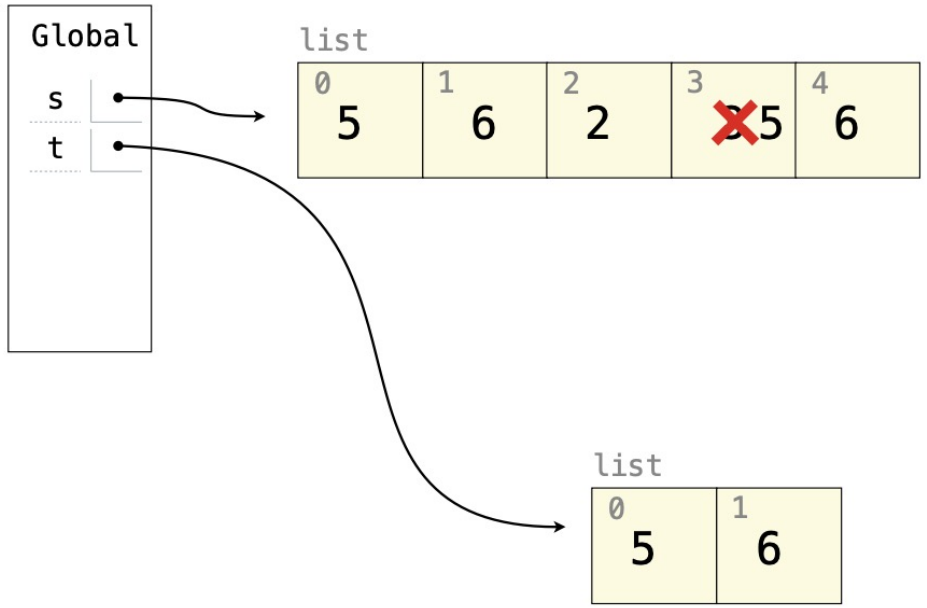
# Lists in Environment Diagrams

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



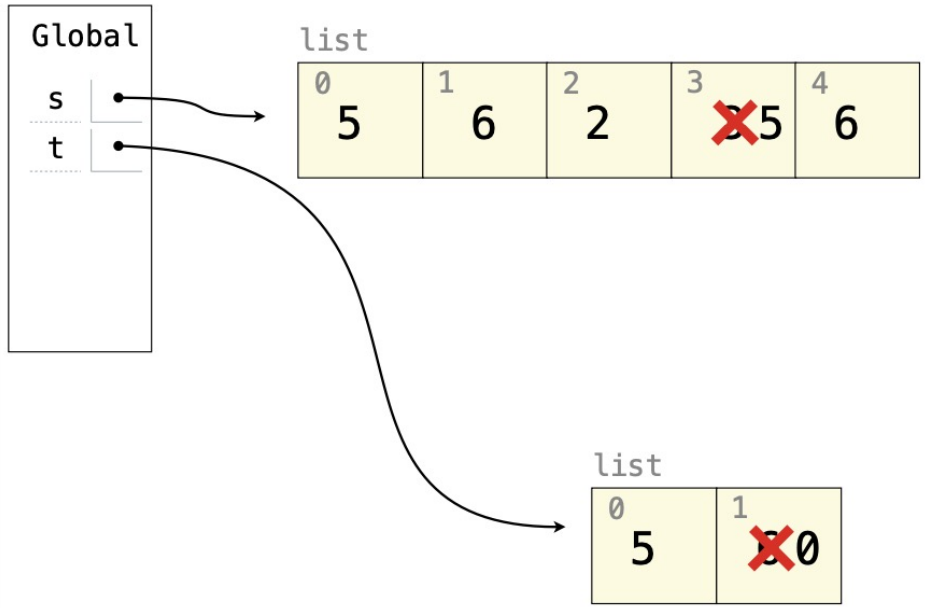
# Lists in Environment Diagrams

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



# Lists in Environment Diagrams

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

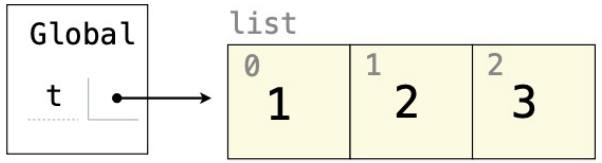


# Lists in Environment Diagrams

| Operation | Example | Result |
| --- | --- | --- |
| **pop** removes & returns the last element | `t = s.pop()` | `s → [2]`<br>`t → 3` |
| **remove** removes the first element equal to the argument | `t.extend(t)`<br>`t.remove(5)` | `s → [2, 3]`<br>`t → [6, 5, 6]` |
| **slice assignment** can remove elements from a list by assigning [] to a slice. | `s[:1] = []`<br>`t[0:2] = []` | `s → [3]`<br>`t → []` |

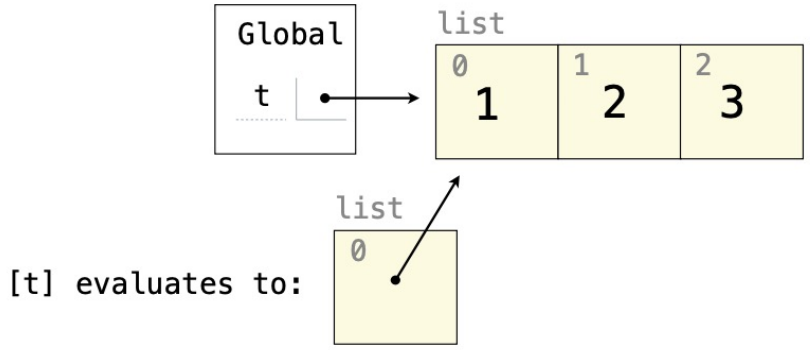
# Lists in Lists in Lists in Environment Diagrams

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



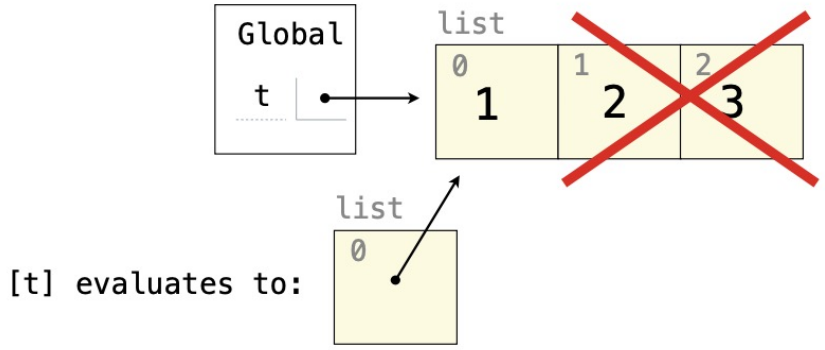
# Lists in Lists in Lists in Environment Diagrams

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



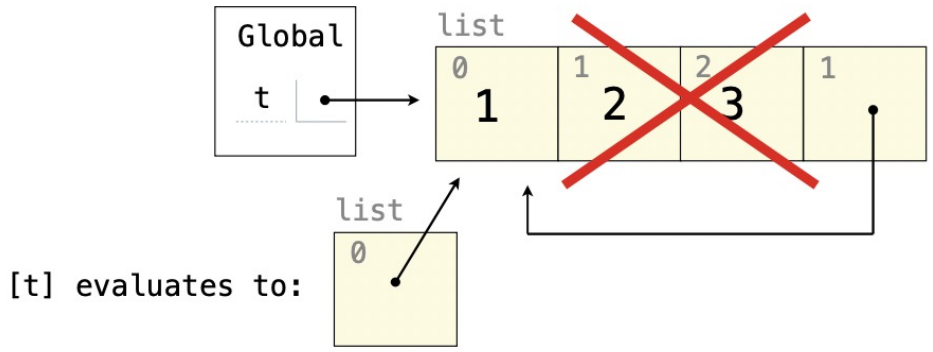
# Lists in Lists in Lists in Environment Diagrams

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



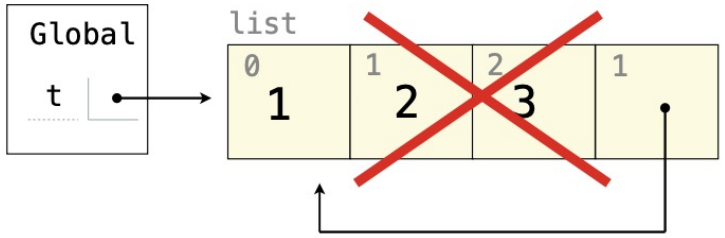
# Lists in Lists in Lists in Environment Diagrams

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



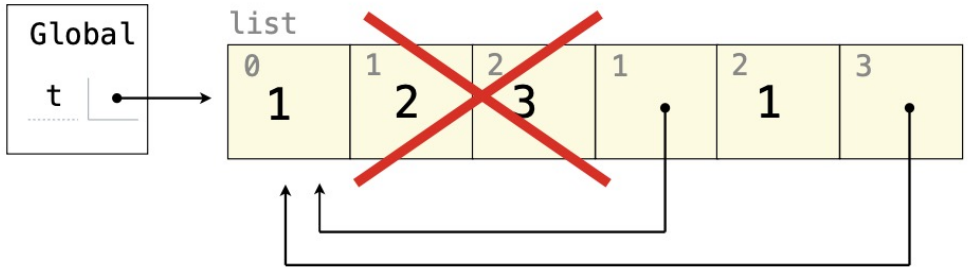
# Lists in Lists in Lists in Environment Diagrams

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



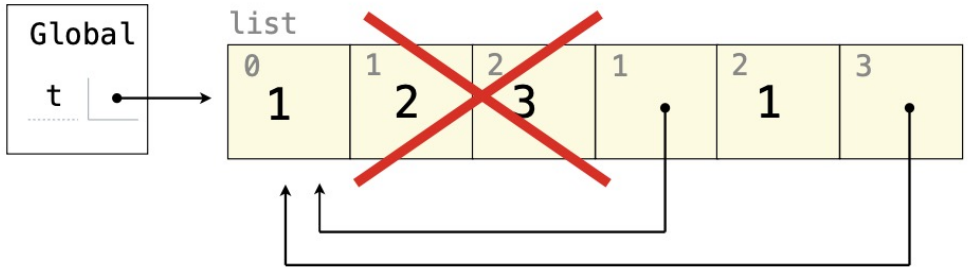
# Lists in Lists in Lists in Environment Diagrams

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



# Lists in Lists in Lists in Environment Diagrams

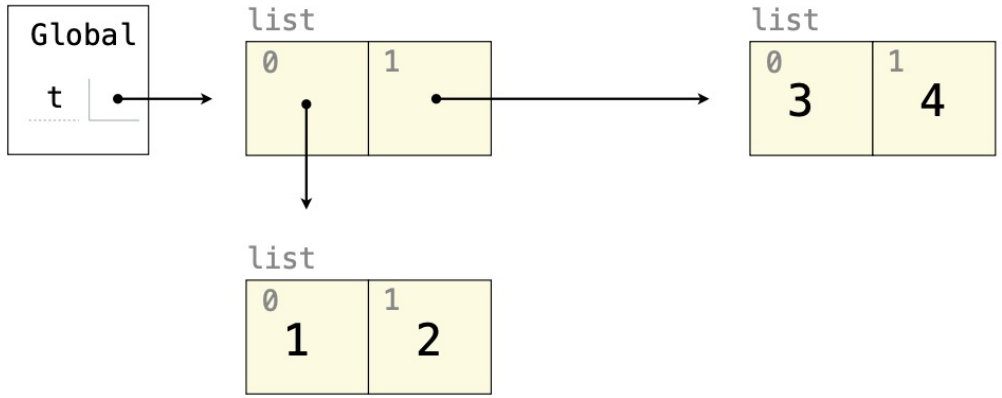
```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```




```
[1, [...], 1, [...]]
```

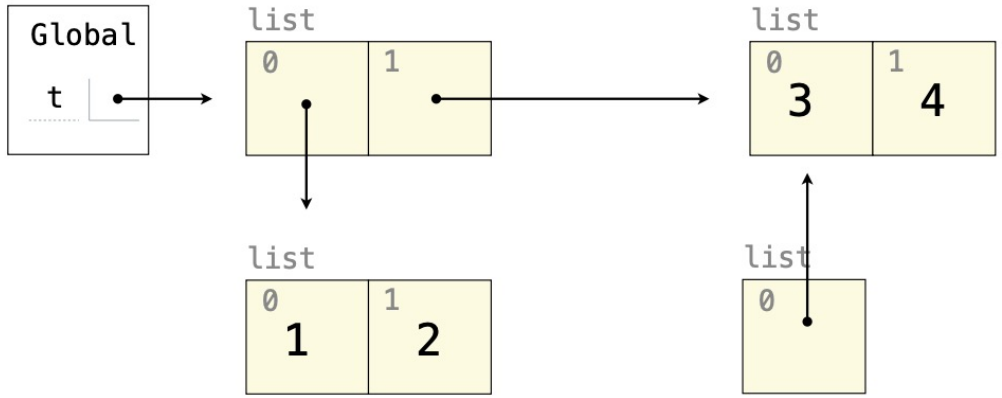
# Lists in Lists in Lists in Environment Diagrams

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



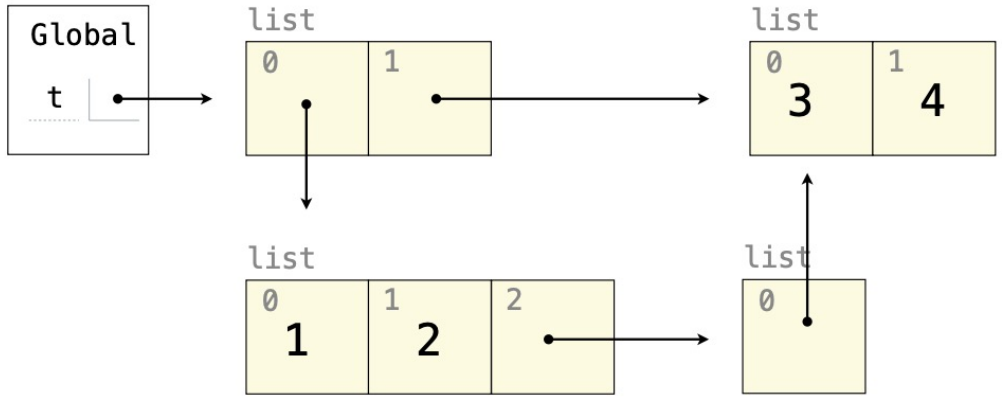
# Lists in Lists in Lists in Environment Diagrams

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



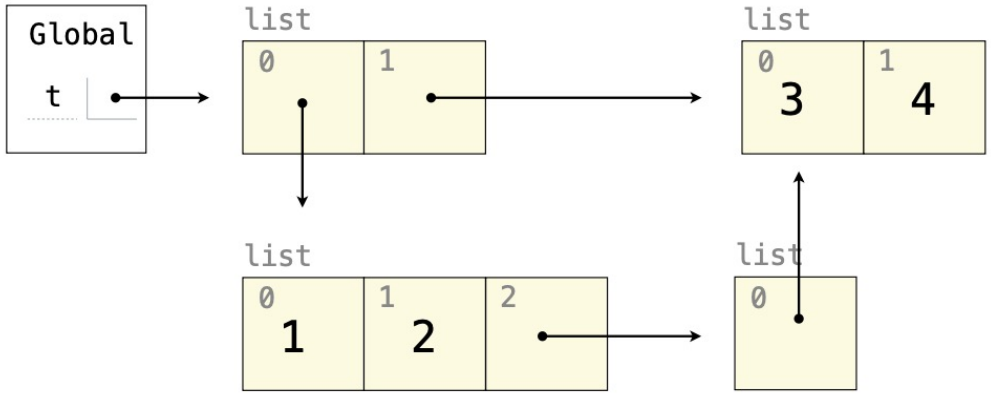
# Lists in Lists in Lists in Environment Diagrams

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



# Lists in Lists in Lists in Environment Diagrams

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




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

# Thanks for Listening