Lists in Environment Diagrams

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

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>append add one element to a list</td>
<td>s.append(5)</td>
<td>s = [2, 3, 5, 6]</td>
</tr>
<tr>
<td>extend add all elements in one list to another list</td>
<td>s.extend(t)</td>
<td>s = [2, 3, 5, 6]</td>
</tr>
<tr>
<td>remove a slicing create new lists containing existing elements</td>
<td>s[3:] = t</td>
<td>s = [2, 3, 5, 6]</td>
</tr>
<tr>
<td>slice assignment also creates a new list containing existing elements</td>
<td>t[0] = s[0]</td>
<td>t = [5, 0]</td>
</tr>
</tbody>
</table>

**Example**

- `t[1] = 0`  
- `s.extend(t)`  
- `t = 0`  
- `s.append(t)`

**Result**

- `[2, 3]`  
- `[2, 0]`  
- `[3, [5, 0]]`  
- `[2, 9, [5, 0]]`  
- `[5, 0]`  
- `[2, 3, 5, 6]`  
- `[2, 3, [5, 6]]`  
- `[2, 3, [5, 0]]`  
- `[2, 3, [5, 6]]`  
- `[2, 3, [5, 0]]`  
- `[2, 3, [5, 6]]`

---

**Announcements**

- New values replaces a slice with slice assignment.
- List containing also creates a new list.
- The elements containing existing create new lists.
- Addition to another list.
- Extend element to a list.
- Append Operation.

---

**Data Examples**

1. `t = [5, 6]`
   - `s = [2, 3]`

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

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>append add one element to a list</td>
<td>s.append(5)</td>
<td>s = [2, 3, 5, 6]</td>
</tr>
<tr>
<td>extend add all elements in one list to another list</td>
<td>s.extend(t)</td>
<td>s = [2, 3, 5, 6]</td>
</tr>
<tr>
<td>remove a slicing create new lists containing existing elements</td>
<td>s[3:] = t</td>
<td>s = [2, 3, 5, 6]</td>
</tr>
<tr>
<td>slice assignment also creates a new list containing existing elements</td>
<td>t[0] = s[0]</td>
<td>t = [5, 0]</td>
</tr>
</tbody>
</table>

   **Example**

   - `t[1] = 0`  
   - `s.extend(t)`  
   - `t = 0`  
   - `s.append(t)`

   **Result**

   - `[2, 3]`  
   - `[2, 0]`  
   - `[3, [5, 0]]`  
   - `[2, 9, [5, 0]]`  
   - `[5, 0]`  
   - `[2, 3, 5, 6]`  
   - `[2, 3, [5, 6]]`  
   - `[2, 3, [5, 0]]`  
   - `[2, 3, [5, 6]]`  
   - `[2, 3, [5, 0]]`  
   - `[2, 3, [5, 6]]`
Lists in Lists in Lists in Environment Diagrams

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

9

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

Global

t
list
1
0
1
2
3

list
0
2
3

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

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

Objects

Land Owners

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

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

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

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

Mutable 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

```python
add(s, 0)
```

Linked List Mutation Example

Adding to an Ordered List

```python
add(s, 0)

def add(s, v):
    """Add v to an ordered list s with no repeats, returning modified s."""
    if v in s:
        return s
    else:
        if s.first:
            s.first = s.first.add(v)
        else:
            s.first = s.rest.add(v, 0)
    return s
```
Adding to an Ordered List

```python
def add(s, v):
    # Add v to an ordered list s with no repeats...
    if s is not List.empty:
        if s.first > v:
            Link(s.first, s.rest)
        elif s.first < v and empty(s.rest):
            Link(s)
        elif s.first < v:
            add(s.rest, v)
        return s
```

Adding to a Set Represented as an Ordered List

```python
def add(s, v):
    # Add v to s, returning modified s.
    s = Link(1, Link(3, Link(5)))
    add(s, 0)
    add(s, 3)
    add(s, 4)
    add(s, 6)
```

Example: Pruning Trees

Removing subtrees from a tree is called pruning.

```python
def prune(t, n):
    # Prune all sub-trees whose label is n.
    t.branches = [b for b in t.branches if b.label != n]
    for b in t.branches:
        prune(b, n)
```

Tree Mutation