Discussion 08

Linked Lists, Mutable Trees, Efficiency

Aditya Balasubramanian
aditbala [at] berkeley [dot] edu

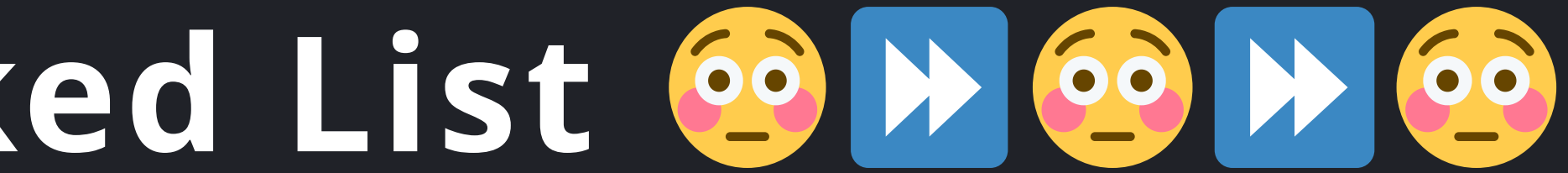
Slides available at `teaching.aditbala.com`

Announcements 📣

- Homework 6 is due Thursday 10/20.
- Ants is due Friday 10/21.
  - Early submission bonus Thursday 10/20.
- Project party 5pm-7:30pm Wed 10/19.
- Midterm 2 is 8pm-10pm Thursday 10/27.
- Read Midterm 2 logistics post
- Guest lecture on Web Apps by Pamela Fox on Monday 10/24.

Clarifications

- **Mutating** vs **Returning New**

Linked List 

Linked Lists

- Can be thought of as a queue waiting to enter a place (starting from the end of the line)
- Each person only knows themself and who is in front of them (rest of the line)

Linked Lists (In More Formal Terms)

- An object that either has a `first` and `rest` attribute or is empty
- `rest`
  - Must be `Link.empty` or another Linked List
  - Recursive data type!
- `first`
  - Any data type

Linked List Example

```
one = Link(1, Link(2, Link(3)))
>>> one.first
1
>>> one.rest
Link(2, Link(3))
>>> one.rest.rest
Link(3)
>>> one.rest.rest.first
3
>>> Link(Link(1), Link(2, Link(3)))
>>> Link(1, 2)
```

Worksheet 😃

Mutable Trees 

Tree

- Initializing a `Tree`
  - `Tree(label, branches=[])`
  - `t = Tree(1, [Tree(2), Tree(3)])`
- Accessing branches of a `Tree`
  - `t.branches` -> `[Tree(2), Tree(3)]`
- Checking if a `Tree` is a leaf
  - `t.is_leaf()` -> `False`
  - `t.branches[0].is_leaf()` -> `True`
- Getting label of a `Tree`
  - `t.label` -> `1`

`Tree` Implementation

```
class Tree:
    def __init__(self, label, branches=[]):
        for b in branches:
            assert isinstance(b, Tree)
        self.label = label
        self.branches = branches

    def is_leaf(self):
        return not self.branches
```

Worksheet

Thank you!

Anon Feedback -> https://tinyurl.com/adit-anon