

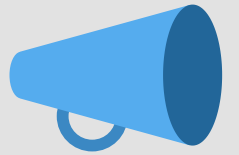
Discussion 04

Tree Recursion and Python Lists

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

Announcements



Tree Recursion



Tree Recursion

- What is Tree Recursion?
 - Recursion, but with more recursive calls
 - Can break down the problem in more than one way
 - With all of the options drawn out, looks like a tree of recursive calls
- When and Why?
 - Useful when the original problem can be broken down in multiple ways
 - Accumulate all sub-problems with multiple recursive calls

FIND EXAMPLE FOR FIB

Lists



Lists

- An indexed collection of any data type
- Examples of valid lists:
 - `[1, 2, 3]`
 - `[True, False, 'boo']`
 - `[[4], [3, 6, 7], [8]]`

Creation and Usage

- In order to access the values in our list, we have to use the index
- Python lists are zero indexed, so the first element is at index 0
- `n` element is at `n-1` index
- Can also access elements in reverse order through negative index
 - Last element is accessed through index `-1` or `len(list) - 1`

```
>>> a = [1, 2, [3, 4]]
>>> a[0]
1

>>> a[2]
[3, 4]

>>> a[2][0]
3
```


WWPD

List Slicing

- How do you access a subset of the list?
- List slicing: creating a copy of part of the list
 - Syntax: `list[start index: non inclusive end index: step size]`
 - step size by default is 1
 - negative step size means list is reversed

List Slicing Example

```
>>> a = [7, 89, True, ['cat']]
```

```
>>> a[1:3]  
[89, True]
```

```
>>> a[:3:2]  
[7, True]
```

```
>>> a[::-1]  
[['cat'], True, 89, 7]
```

```
>>> a[:3:-1]  
[]
```