Hi! Welcome to 61A Discussion :)

We will begin at **5:10**!

Attendance form and skeleton notes:

cs61a.bencuan.me

Announcements

- HW3 due tonight!
- Cats Phase 1 also due tonight!
- Try cats! cats.cs61a.org

Agenda

- Attendance
- Tree Recursion
- Lists
  - List Slicing
  - List Comprehensions

Tree Recursion

What is tree recursion? (Lab review 0)

- Definition: **making multiple recursive calls at one time**
  - Can be very challenging in practice!

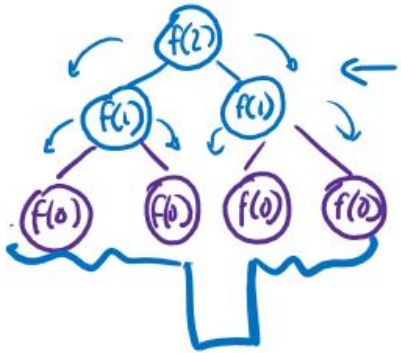
```
def f(x):
    if <base case condition>:
        return <result>

    else:
        return f(<recursive step>) + f(<another recursive step>)
```

(could be any operation)

Lab review (1)

- Creates a branching structure:



← # of recursive calls grows exponentially

← reaches base case multiple times!

```
def f(x):
    if <base case condition>:
        return <result>

    else:
        return f(<recursive step>) + f(<another recursive step>)
```

(could be any operation)

Partitions (Lab review 2)

- A very common tree recursion pattern:
  - You're given two options
  - You need to combine the two options together

- Example: HW03 count_coins
  - Option 1: try current coin size
  - Option 2: increase coin size
  - Combine: add total count of two options

Q1: Count Stair Ways

- **Note:** not *count stairways*!

Imagine that you want to go up a flight of stairs that has `n` steps, where `n` is a positive integer. You can either take 1 or 2 steps each time. In this question, you'll write a function `count_stair_ways` that solves this problem. Before you code your approach, consider these questions.

- **Base Case?** (simplest value of n?)
  - Hint: there might be more than one
- **Recursive Case?** (what do `count_stair_ways(n - 1)` and `count_stair_ways(n - 2)` represent?)

Count Stair Ways Tree Diagram

count_stair_ways(4) returns 5. Why?

Q2: Count K

Consider a special version of the `count_stair_ways` problem, where instead of taking 1 or 2 steps, we are able to take up to and including `k` steps at a time. Write a function `count_k` that figures out the number of paths for this scenario. Assume `n` and `k` are positive.

- **Hints:**
  - This is still a partition problem! Think about all of the options you need to add together (and how to compute them).
  - More hints coming soon

Lists

Q3: WWPD Lists

- Let’s try some polls!!
  - sli.do/616161

List Slicing


```
lst[ start : end : step]
```

You can omit any of the above and use default values!

- Start: 0
- End: len(lst)
- Step: 1

Example: `lst[3:]` returns all but the first 3 elements of lst

Q5: Max Product

Write a function that takes in a list and returns the maximum product that can be formed using nonconsecutive elements of the list. The input list will contain only numbers greater than or equal to 1.

Hints:

- **What are your options (partitions)?**
  - What do they return?
  - How do you combine them?
- **Use list slicing!**

List Comprehensions

The Syntax

```
[ op for el in lst if cond]
```

```
result = []
for el in lst:
    if cond:
        result += [op]
return result
```

Example: get all even numbers in list

```
[ x for x in lst if x%2==0]
```

```
result = []
for x in lst:
   if x%2==0:
      result += [x]
return result
```

Q4: Even Weighted

Write a function that takes a list `s` and returns a new list that keeps only the even-indexed elements of `s` and multiplies them by their corresponding index.

```
[ op for x in lst if cond]
```