

# DISCUSSION 03

---

## Recursion

Mingxiao Wei

[mingxiaowei@berkeley.edu](mailto:mingxiaowei@berkeley.edu)

Feb 09, 2023



# FROM LAST TIME... 🙄🙄

What's the best dish?

Salt and Pepper Shrimp	Pizza
butter chicken curry	Pizza
thai noodles	Any Korean food
sushi	PAD SEE EW
not currying	sushi
Curry	dishes
curry	sushi
pad thai	ramen
sushi!	Salad :p
Seafood	sushi
Tom Yum Soup	NOT curry I don't think I can eat curry after this
meatball	Noodles
curry	Roast potato
won ton noodle	mum made dishes are the best
fish soup	Ramen
curry	curry

# LOGISTICS

---

- Hog 
  - The entire project due Fri 02/10
  - Submit everything by tomorrow 02/09 for one extra point!
- Grading update - see [ed post #821](#) for more details
  - If you have issues with your lab or discussion score, please email me!
- Please remember to fill out the attendance form right after section! I'll close the form at midnight
- The deadline to add/drop classes was yesterday. Congrats on making it through 61A thus far! Let's continue the journey together 

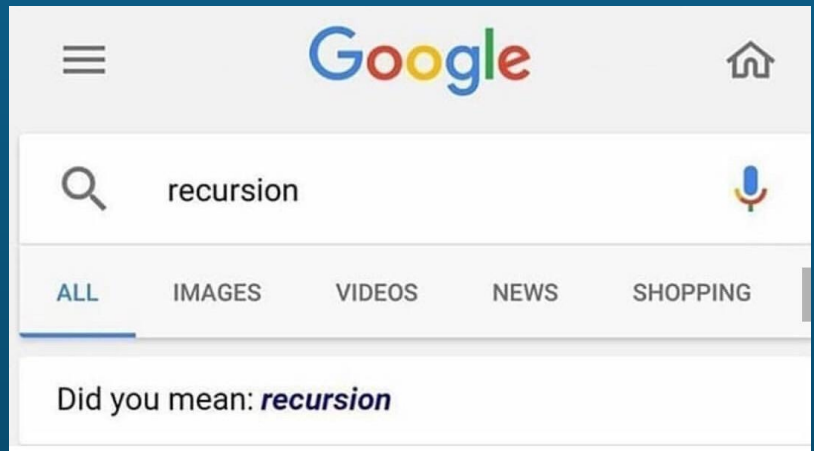
# SOME POST-MIDTERM WORDS...

---

- It's only 13% (40 pts) of your total grade!
- Keep in mind that we have 4 pts of extra credit (for example, early submission of projects) and the clobber policy!
- The first midterm is more of a learning experience to take into midterm 2
- Your grade is in no way a reflection of your worth as a programmer!
- Also a reminder that this class is not curved
- If you'd like to chat with me about non-content stuff (advising, etc.), feel free to email me!
- Other resources:
  - [Studying Guide](#)
  - [Advising OH from other experienced course staff](#)

# RECURSION





# INTRODUCTION

---

- A recursive function is a function that is defined in terms of itself - i.e., the function calls itself
- 3 main steps in a recursive function:
  1. Base case - the simplest function input, or the stopping condition for the recursion
  2. Recursive call on a smaller problem - calling the function on a smaller problem that our current problem depends on. We can assume that a recursive call on this smaller problem will give us the expected result as long as the we implement other parts correctly - the idea of the "recursive leap of faith".
  3. Solve the larger problem / Combination step - usually by using the result from the recursive call to figure out the result of our current problem

# RECURSION - ANALOGY

---

- Imagine that you were waiting in a line and wanted to know how many people are in front of you, you can ask the person before you how many people are in front of them, then they'll ask the same question to the person in front of them, etc.
- Base case: the first person in line, responds 0
- Recursive call: asking the person before you the same question
- Combination step: add one to the response from the person before

Note that the first person to ask is the last person to get their response - in recursion, the first frame to open is the last frame to close



# RECURSION - EXAMPLE

```
def factorial(n):  
    """Return the factorial of N, a positive integer."""  
    if n == 1:  
        return 1  
    else:  
        return n * factorial(n - 1)
```

- Base case: When  $n$  is 1, return 1
- Recursive call: `factorial(n - 1)`
- Combination step: By the recursive leap of faith, we can assume that `factorial(n - 1)` returns the correct result of  $(n - 1)!$ .  
Therefore, to obtain  $n!$ , we just need to multiply the result from the recursive call by  $n$  ( $n! = n * (n - 1)!$ )

# WORKSHEET - Q1,2,3

---

# PROBLEM-SOLVING STRATEGIES

---

- Base case:
  - Usually hinted in the doctests - make sure to read them!
  - Think about different possibilities of "when to stop"
  - If multiple arguments change throughout the recursive calls, make sure to include a base case for *each* argument - this is especially relevant in tree recursion
  - If there are multiple base cases, think about whether or not the order of them matters
- Recursive call:
  - break down the problem into smaller ones
  - Make sure that the base cases are *reachable* - that is, the argument changes toward the base case each time
- Combination step:
  - Assuming the implementation is correct (this is the recursive leap of faith), **what will the recursive call return?**

# HELPER FUNCTIONS FOR RECURSION

---

- When to use:
  - Need to keep track of more variables than the given parameters of the outer function
  - Need the original parameter from the outer function for each recursive call AND some other parameters that change throughout the recursive call
- Where - usually, though not necessarily, nested within the original function
- How - define the helper function, and return a call to it with appropriate initial arguments

# WORKSHEET - Q4,5,6

---

# ATTENDANCE! 🤠

---

[go.cs61a.org/mingxiao-att](https://go.cs61a.org/mingxiao-att)

- The attendance form and slides are both linked on our [section website!](#)
- Please leave any anonymous feedback here [go.cs61a.org/mingxiao-anon](https://go.cs61a.org/mingxiao-anon)
- Once again, please do remember to fill out the form by midnight today!!

