

Recursion

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 35, Fall 2020

Friday November 13

Recursion

Ever wonder what would happen if you called a function from inside a function?

```
def fun(x):  
    fun(x)
```

- Called **recursion**. Yes, it is a valid thing we can do!
- Powerful design technique to produce elegant solutions to challenging problems that might otherwise require lots of code.
- When to use recursion? Solution to a complex problem can be expressed in terms of solutions to slightly simpler versions of the original problem.

Example: Factorial function and symbol table

Checklist for designing recursive solutions

1. Can the problem can be built up in terms of solutions to simpler versions of the problem.
 - In order words, will calling `f()` from within `f()` lead to a simpler version of the same problem.
 - `factorial()`: `factorial(n-1)` from within `factorial(n)`.
2. What's the stopping condition? How will the recursion stop?
 - For factorial: `n == 0`.
 - The stopping condition is also called the recursion **base case**: when the recursion "bottoms out" and gives us an actual hard value or closed form solution. **DOES NOT** involve calling the function again.
3. Does calling `f()` from within `f()` get us closer to the stopping condition?
 - If not, we'll end up in an infinite loop (**infinite recursion**).
 - Example: `factorial(n)` from within `factorial(n)`...the problem gets no simpler!
 - All recursive solutions need to have one more or more base cases.

Example: Summing elements in a list

Goal: Add up all the elements of a list *using recursion*. Example:

[1, 1, 1, 1] # 4

- **Checklist item #1:** Can the problem can be built up in terms of solutions to simpler versions of the problem.
 - Yes! Sum is entire list is the first element PLUS the rest of the list.
- **Checklist item #2:** What's the stopping condition? How will the recursion stop?
 - When the right-hand list is just 1 item long.
- **Checklist item #3:** Does calling `f()` from within `f()` get us closer to the stopping condition?
 - Yes! The remaining right-hand list is getting smaller.
- Let's write the function `lecture_35_sum_list.py`.