CISC320 Algorithms

Recursion

Austin Cory Bart
AlgoTutorBot
University of Delaware

Recursion

Definition

- a process in which the result of each repetition is dependent upon the result of the next repetition.
- If you still don't understand, go see the definition of **Recursion**

Hofstadter's Law

- *“It always takes longer than you expect, even when you take into account Hofstadter's Law.”*

Sesquipedalian

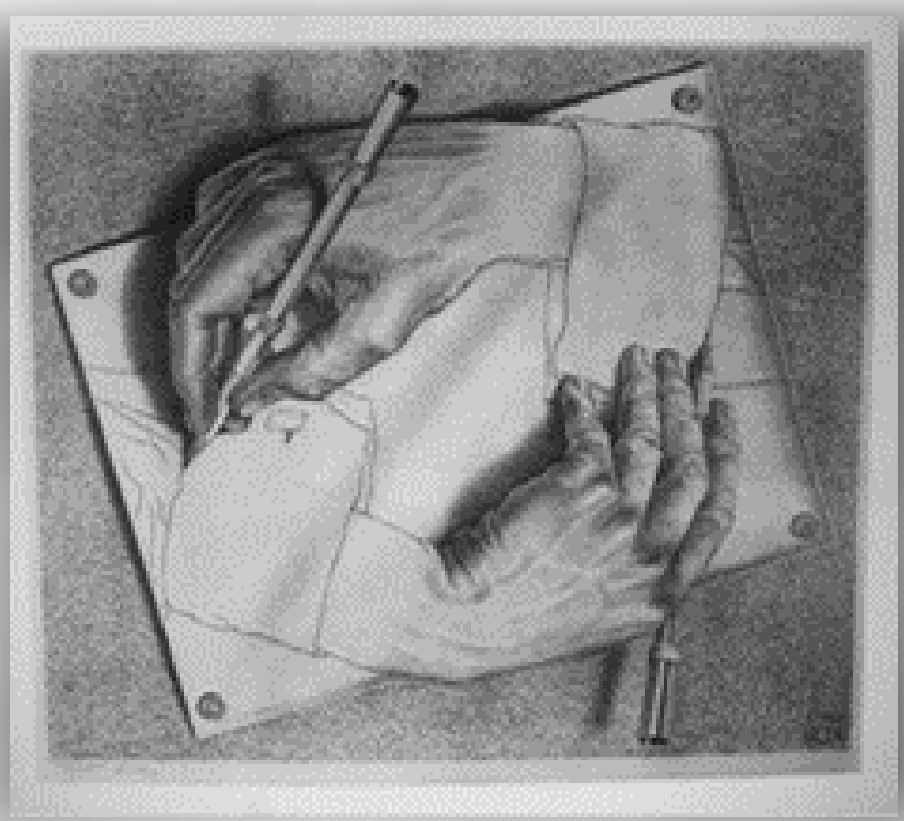
- *A person who uses words like sesquipedalian.*

Yogi Berra

- *“Its déjà vu all over again.”*

Art

- MC Escher



What Is Recursion?

- Consider hiring a contractor to build something
  - He hires a subcontractor for a portion of the job
  - That subcontractor hires a sub-subcontractor to do a smaller portion of job
- The last sub-sub- … subcontractor finishes
  - Each one finishes and reports “done” up the line

Example: The Countdown

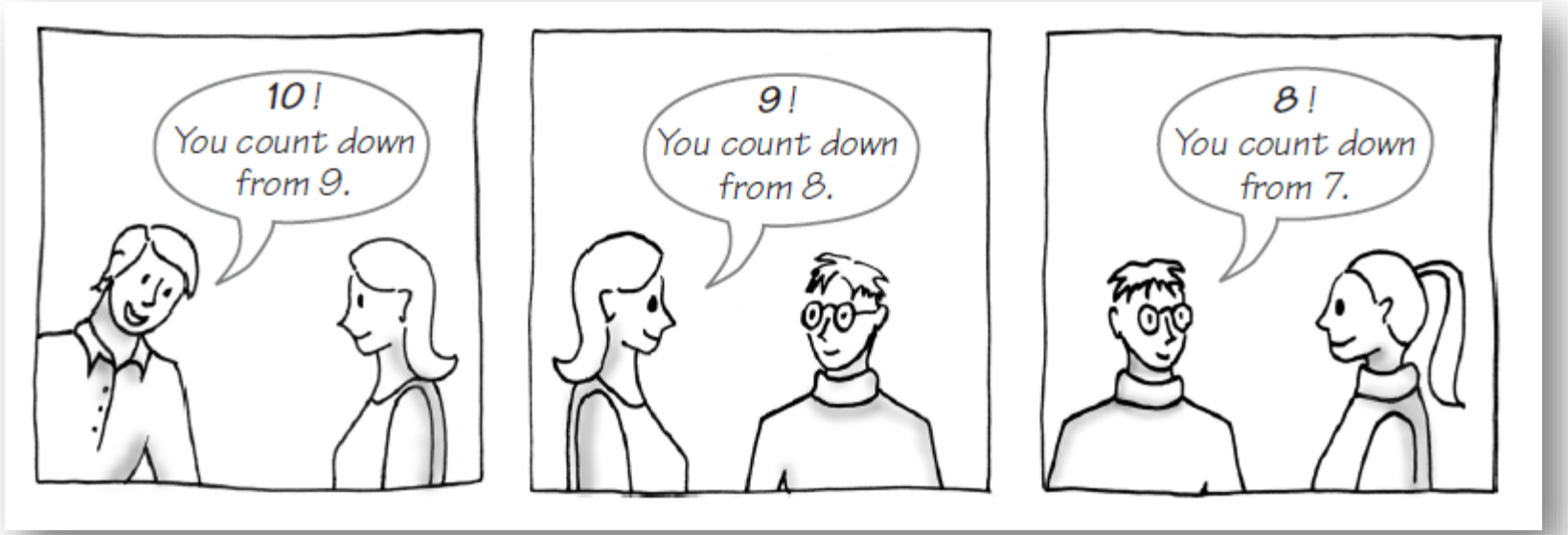


FIGURE 7-1 Counting down from 10

Example: The Countdown

FIGURE 7-1 Counting down from 10

Example: The Countdown

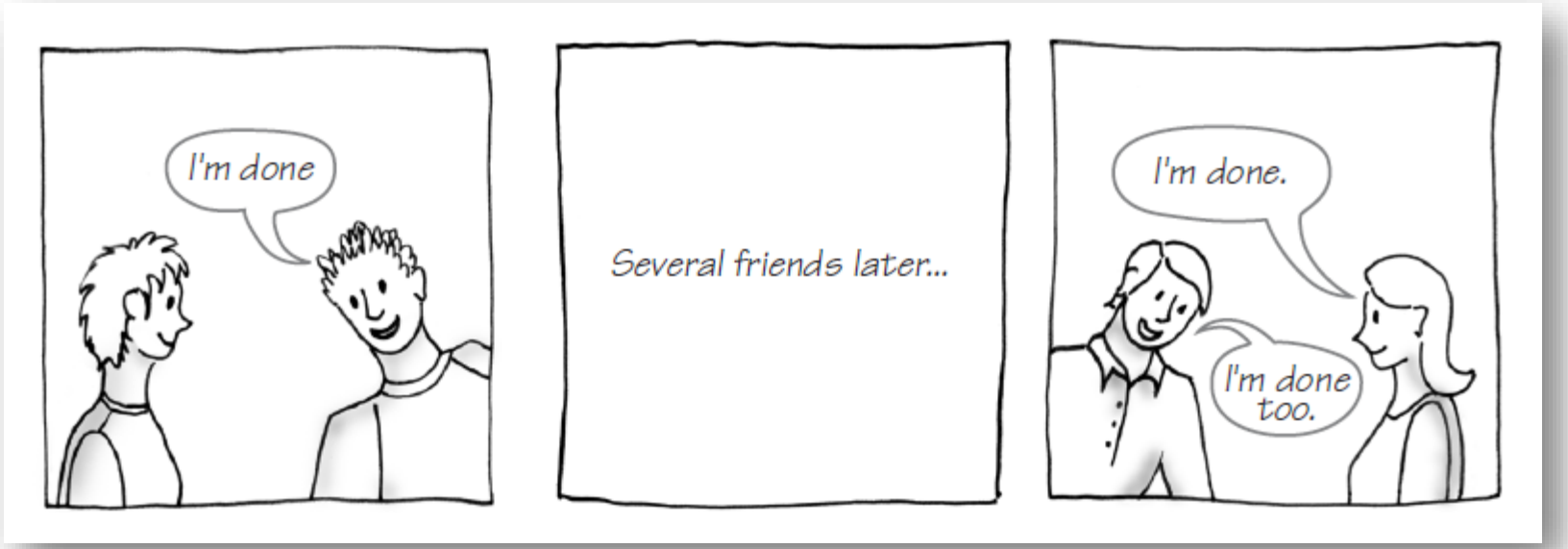


FIGURE 7-1 Counting down from 10

Recursion in Python

```
def count_down(value):
    print(value)
    if value == 0:
        return "I'm done"
    else:
        return count_down(value-1)
```

The function calls itself!

Definition

- Recursion is a problem-solving process
  - Breaks a problem into identical but smaller **subproblems**.
- A function that calls itself is a **recursive function.**
  - The invocation is a **recursive call** or **recursive invocation**.

Design Guidelines

- Function must take parameters
- Function definition must contain logic involving the parameters, leading to different cases
- **Base Case:** One or more cases should provide solution that does not require recursion
  - Else infinite recursion
- **Recursive Case:** One or more cases must include a recursive invocation

Basic Recursion Steps

1. Put an IF/ELSE statement in your code
2. Handle the "Base Case"
3. Handle the Recursive Case
    1. Modify the input
    2. Call the recursive method
    3. Do something with the result

Recursion Template

```
def function_name(parameter):
    if ____:
        ...
    else:
        reduced = ... parameter ...
        result = function_name(... reduced ...)
        ... result
```

1. The "simplest case"

2. Need to reduce parameter

3. Need to call function with reduced data

4. Need to do something with result

Base Case

- What do when
  - The number is zero
  - The string is empty
  - The list is empty
  - The dictionary is empty
  - …
- Usually…
  - Return zero, an empty string, an empty list, etc.

Recursive Case

1. **Reduce:** Break off a piece
2. **Recurse:** Call the recursive method on the rest
3. **Result:** Combine the results

Why recursion?

- Anything you can do with a WHILE or FOR loop, you can do with recursion
  - And vice versa
- Some people find recursion more natural, some people prefer WHILE/FOR loops
- Some problems are very easy to express with recursion

Examples

Fibonacci

- The nth Fibonacci number is the sum of the two previous Fibonacci numbers
- Fib(1) is 1
- Fib(2) is 1
- Fib(n) = Fib(n-1) + Fib(n-2)

Sum a list

- A list can be seen as recursive
- A list is either:
  - EMPTY
  - A first element and the rest
    - First = List[0]
    - Rest = List[1:]
- Anything you can do with a for loop you can do with recursion

Process a tree

- Person is a dictionary with the following keys:
  - "Name": string
  - "Wealth": integer
  - "Children": list of Person
- How much money does a Person have if you include all their descendants?

Common Novice Mistakes

When using recursion

Forgot base case

- You must have an IF statement, and one of the cases should handle the end result
  - If the integer is zero, return 0/False/empty string/etc
  - If the string is empty, return 0/False/empty string/etc
  - …

Forgot to reduce recursive case

- In the recursive case, you must make something smaller!
  - Decrease the integer by 1
  - Remove the last character from the string
  - Remove the last element of a list
  - …

Forgot to recurse

- If you didn't call the function somewhere inside of its definition, then you aren't using recursion

Forgot to use result

- Did you remember to **return** your recursive result?

Tried to handle n+1+1 case instead of n+1

- Students often try to think two steps ahead, instead of focusing on just the current case