Recursion

Part 2

April 11, 2022

Administrative Notes

Project 1 is due by midnight tonight

Project 2 is now available. We'll talk about it in this lecture

Homework 6 is also available. Due April 18 before midnight

- On Recursive

Recursion

From last Wednesday's lecture:

- When a function calls itself, that's called "recursion" or "recursive programming." And it turns out to be a very useful way to solve certain problems. Like, those where certain things have to be done over and over, with only slight differences.
- Using loops is called "iteration" or "iterative programming."
- Every programming problem that can be solved with recursion can also be solved with iteration!!
- So why use recursion? Sometimes it just makes the problem easier to solve

For Recursive programming to work

- That is, to get to a solution to your problem:
- There must be at least one “base case”
  - A “base case” is something so easy you can solve it directly; e.g. 1! = 1
  - There can be multiple base cases; you have to have at least one
- There must be at least one “recursive case”
  - A “recursive case” is when the function calls itself with a simpler or smaller input
  - The recursive case must eventually get you to a base case in order to eventually solve the problem
- And the base case(s) and recursive case(s) must eventually get to a solution to the problem you want to solve

An example: determining if a string is a palindrome

- A palindrome is a string that is identical forwards and backwards. The characters in the string have be exactly the same in each direction
- Examples:
  - “battab” is a palindrome - it’s the same whether you read it forward or backward
  - “cat” is not a palindrome - backward it’s “tac”
- So let’s write a recursive function that determines whether a string is a palindrome or not
  - The function will have one parameter, the string.
  - It will return a Boolean - True if the string is a palindrome; and False if it is not

Palindromes

Calculate whether a string is a palindrome using recursion

What's the base case?

- A string that is zero characters long - an empty string - IS a palindrome
- A string that is one character long IS a palindrome
- A string where the first character is DIFFERENT from the last character is NOT a palindrome
    - "cat" is NOT a palindrome
-

Palindromes - recursive case

What about the recursive case?

- IF the first character is the SAME as the last character, the string IS a palindrome if what's left when you throw away the first and last characters is a palindrome

yay - remove first character; remove last character; look at what's left

- a - IS a palindrome

Pseudocode

x=yay IS a palindrome

First character equals last character

- Create the substring by throwing away the first and last character
  - x[1:-1] or (x[1:len(x)-1])
- A

y = tt

Throw away first and last character - we have nothing left - we have an empty string left

battab - atta - tt - empty string - IS a palindrome

Another example - sum a list of numbers

What are the base cases?

- The sum of a list of zero numbers is 0
- The sum of a list of one number is that number

What's the recursive case?

1, 2,3 4 5 = 1 + sum 2, 3, 4, 5 = 1+2+sum 3, 4, 5

1 + 2 + 3 + sum 4, 5 1 + 2 + 3 + 4 +5 + sum empty list = 0