http://www.cs.cornell.edu/courses/cs1110/2018sp

Lecture 16: More Recursion!

CS 1110

Introduction to Computing Using Python

Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Recursion

- **Recursive Function**:
  A function that calls itself (directly or indirectly)
- **Recursive Definition**:
  A definition that is defined in terms of itself

3

A Mathematical Example: Factorial

Non-recursive definition:

$n! = n \times n\text{-}1 \times \ldots \times 2 \times 1$

$\quad = n\ (n\text{-}1 \times \ldots \times 2 \times 1)$

Recursive definition:

$n! = n\ (n\text{-}1)!$	for $n > 0$	**Recursive case**
$0! = 1$		**Base case**

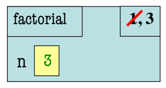
What happens if there is no base case?

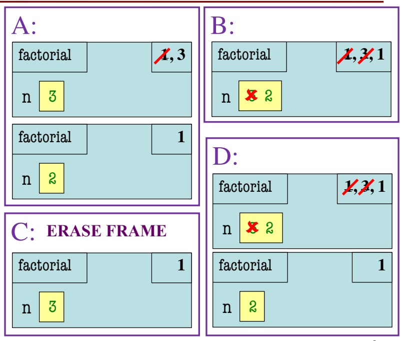
4

What happens next? (Q)

```
def factorial(n):
    """Returns: factorial of n.
    Pre: n ≥ 0 an int"""
1   if n == 0:
2       return 1

3   return n*factorial(n-1)
```

Call: factorial(3)



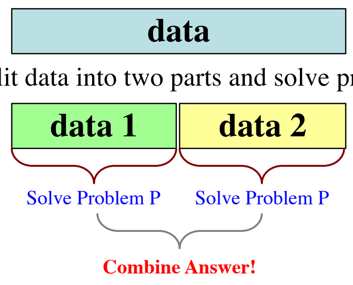


8

Recall: Divide and Conquer

Goal: Solve problem P on a piece of data

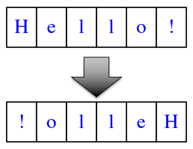
Idea: Split data into two parts and solve problem



24

Example: Reversing a String

```
def reverse(s):
    """Returns: reverse of s

    Precondition: s a string"""
    # 1. Handle base case



    # 2. Break into two parts



    # 3. Combine the result
```

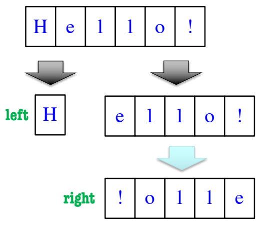


25

How to Combine? (Q)

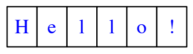
```
def reverse(s):
    """Returns: reverse of s

    Precondition: s a string"""
    # 1. Handle base case



    # 2. Break into two parts



    # 3. Combine the result
    return
```

A: left + right | B: right + left | C: left | D: right



What is the Base Case? (Q)

```
def reverse(s):
    """Returns: reverse of s

    Precondition: s a string"""
    # 1. Handle base case

    # 2. Break into two parts
    left  = reverse(s[0])
    right = reverse(s[1:])

    # 3. Combine the result
    return right+left
```



A: if s == "":
    return s

B: if len(s) <= 2:
    return s

C: if len(s) <= 1:
    return s

D: Either A or C would work

E: A, B, and C would all work

Alternate Implementation (Q)

```
def reverse(s):
    """Returns: reverse of s
    Precondition: s a string"""
    # 1. Handle base case
    if len(s) <= 1:
        return s

    # 2. Break into two parts
    half  = len(s)//2
    left  = reverse(s[:half])
    right = reverse(s[half:])

    # 3. Combine the result
    return right+left
```

Does this work?

A: YES

B: NO

Example: Palindromes

- **Example:**

  AMANAPLANACANALPANAMA

- Can we define recursively?

Example: Palindromes

- String with ≥ 2 characters is a palindrome if:
  - its first and last characters are equal, and
  - the rest of the characters form a palindrome
- **Example:**

  have to be the same

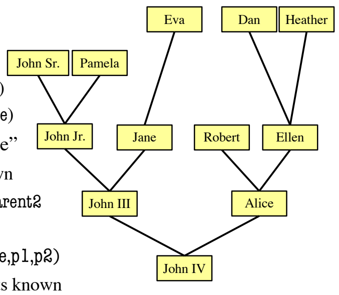
  AMANAPLANACANALPANAMA

  has to be a palindrome
- **Implement:** def ispalindrome(s):

  """Returns: True if s is a palindrome"""

Recursion and Objects

- Class Person
  - Objects have 3 attributes
  - name: String
  - parent1: Person (or None)
  - parent2: Person (or None)
- Represents the "family tree"
  - Goes as far back as known
  - Attributes parent1 and parent2 are None if not known
- **Constructor**: Person(name,p1,p2)
  - Or Person(n) if no parents known



Eva
Dan
Heather
John Sr.
Pamela
John Jr.
Jane
Robert
Ellen
John III
Alice
John IV