Lecture 16: More Recursion!

CS 1110
Introduction to Computing Using Python

Recursion

Recursive Function:
A function that calls itself (directly or indirectly)

Recursive Definition:
A definition that is defined in terms of itself

From previous lecture: Factorial

Non-recursive definition:
\[ n! = n \times (n-1) \times \ldots \times 2 \times 1 \]
\[ = n \times (n-1) \times \ldots \times 2 \times 1 \]

Recursive definition:
\[ n! = n \times (n-1)! \quad \text{for } n > 0 \quad \text{Recursive case} \]
\[ 0! = 1 \quad \text{Base case} \]

Divide and Conquer

Goal: Solve problem P on a piece of data

Idea: Split data into two parts and solve problem

Combine Answer!
Example: Reversing a String

```python
def reverse(s):
    """Returns: reverse of s
    Precondition: s a string"
    # 1. Handle base case
    # 2. Break into two parts
    # 3. Combine the result
```

Alternate Implementation (Q)

```python
def reverse(s):
    """Returns: reverse of s
    Precondition: s a string"
    # 1. Handle base case
    # 2. Break into two parts
    # 3. Combine the result
```

Does this work?

A: YES  
B: NO

Following the Recursion

```python
def deblank(s):
    """Returns: s without spaces"
    if s == "":
        return s
    elif len(s) == 1:
        return ""
    if s[0] == " "
        else s
    left = deblank(s[0])
    right = deblank(s[1:])
    return right+left
```

Execute the function call

```python
reverse("Hello")
Result: 'olleH'
```
Example: Palindromes

- Example:
  
  AMANAPLANACANALPANAMA
  
  MOM

  - Dictionary definition: “a word that reads (spells) the same backward as forward”

  - 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

```
def is_palindrome(s):
    # Base case
    if len(s) <= 1:
        return True

    # Recursive case
    middle_pal = is_palindrome(s[1:-1])

    return s[0] == s[-1] and middle_pal
```

Implement: `def is_palindrome(s):`

```python
# example: Palindromes
```

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)`

```
def num_ancestors(p):
    # Base case
    if (p.parent1 is None) and (p.parent2 is None):
        return 0

    # Recursive case
    parent1s = num_ancestors(p.parent1) + 1
    parent2s = num_ancestors(p.parent2) + 1

    return parent1s + parent2s
```

```python
# example: Recursion and Objects
```
def num_ancestors(p):
    """Returns: num of known ancestors
    Pre: p is a Person"
    # 1. Handle base case.
    if p.parent1 == None and p.parent2 == None:
        return 0
    # 2. Break into two parts
    parent1s = 0
    if p.parent1 != None:
        parent1s = 1+num_ancestors(p.parent1)
    parent2s = 0
    if p.parent2 != None:
        parent2s = 1+num_ancestors(p.parent2)
    # 3. Combine the result
    return parent1s+parent2s

We don’t actually need this. It is handled by the conditionals in #2.

def all_ancestors(p):
    """Returns: list of all ancestors of p"
    # 1. Handle base case.
    # 2. Break into parts.
    # 3. Combine answer.

Optional practice question. Try it after you complete this week’s lab exercise.