Lecture 16:
More Recursion!

CS 1110
Introduction to Computing Using Python

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

What happens if there is no base case?

Recall: Divide and Conquer

Goal: Solve problem P on a piece of data

Idea: Split data into two parts and solve problem

Data

Data 1
Data 2

Solve Problem P
Solve Problem P
Combine Answer!

Recursion

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

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

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
How to Combine? (Q)

```python
def reverse(s):
    """Returns: reverse of s
    Precondition: s a string"
    # 1. Handle base case
    left = reverse(s[0])
    right = reverse(s[1:])
    # 2. Break into two parts
    half = len(s) // 2
    left = reverse(s[:half])
    right = reverse(s[half:]):
    # 3. Combine the result
    return left + right
```

What is the Base Case? (Q)

```python
def reverse(s):
    """Returns: reverse of s
    Precondition: s a string"
    # 1. Handle base case
    if s == "":
        return s
    if len(s) <= 2:
        return s
    if len(s) <= 1:
        return s
    # 2. Break into two parts
    left = reverse(s[0])
    right = reverse(s[1]):
    # 3. Combine the result
    return right + left
```

Alternate Implementation (Q)

```python
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
```

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: AMANAPLANACANALPANAMA
- Implement: def ispalindrome(s):

```
def ispalindrome(s):
    """Returns: True if s is a palindrome"
    has to be the same
    # has to be 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

Example: Palindromes

- Example:
  - AMANAPLANACANALPANAMA
- Can we define recursively?