CSCI 241
Scott Wehrwein
Recursion: Understanding Recursive Methods
Goals

Be able to **understand** and **develop** recursive methods *without* thinking about the details of how they are executed.
How do we understand recursive methods?

1. Make sure it has a **precise specification**.
2. Make sure it works in the **base case**.
3. Ensure that each recursive call makes **progress** towards the base case.
4. Replace each **recursive call** with the **spec** and verify overall behavior is correct.
/** returns # of ‘e’ in string s */

def count_e(s):
    if len(s) == 0:
        return 0
    first = 0
    if s[0] == ‘e’:
        first = 1
    return first + count_e(s[1:end])

How do we understand recursive methods?

1. spec
2. base case
3. progress
4. recursive call → spec
How do we understand recursive methods?

```python
/** returns # of 'e' in string s */
def count_e(s):
    if len(s) == 0:
        return 0
    first = 0
    if s[0] == 'e':
        first = 1
    return first + /*# of 'e' in s[1..end]*/  
```

1. spec

2. base case

3. progress

4. recursive call → spec
How do we develop recursive methods?

1. Write a precise specification.

2. Write a base case without using recursion.

3. Define all other cases in terms of subproblems of the same kind.

4. Implement these definitions using the recursive call to compute solutions to the subproblems.
Example: Reverse a String

1. Write a **precise specification**.

2. Write a **base case** without using recursion.

3. Define all other cases in terms of **subproblems**.
   *The reverse of a string is: (last character) + (interior characters in reverse) + (first character)*

4. Implement the subproblems using **recursive calls**.

```java
/** Return the reverse of s. Pre: s is not null. */
reverse(String s):
    len = s.length();
    if len < 2:
        return s;
    return s[len-1] + reverse(s[1:len]) + s[0]
```