Many recursion examples are on the lectures page
These were mostly authored by Prof. Walker White.
• comments in braces are assertions: conditions assumed to hold
  if that line is reached. Example:
  # {s is empty}
• We are not currently emphasizing the use of assert statements
to enforce preconditions, but they can be quite useful to catch
bugs involving accidental precondition violation. Example:
assert type(s) == str,
repr(s) + ' is not a string'
(repr gives unambiguous string representation)

Reminder: our running example

```python
def num_es(s):
    """Returns: number of 'e's in <s>. Precond: <s> a string""
    # Strategy: break off first character, recur on the rest.
    1 if s == ''; # base case (no recursion))<s> is empty string
    2 return 0
    # recursive case <s> has at least one char
    # note this Python trick s[1:] is '' if len(s) <= 1
    3 return ((1 if s[0] == 'e' else 0) + num_es(s[1:]))
```

Let's understand what happens at execution.

How to Think About Recursive Functions

1. Have a precise function specification.
   • Test cases generally handy here
2. Recursive case(s):
   • Verify recursive cases with the specification
3. Reduction:
   • Arguments of calls must somehow get “smaller”, so each
     recursive call gets closer to a base case
4. Base case(s): (place them before the recursive one)
   • When the recursive case doesn't apply
   • When the argument values are as "small" as possible
   • When the answer is determined with little calculation.
What if we didn’t recur on a "smaller" value?

Hypothetical function definition

```python
def bad(s):
    if s == ""
        return 0
    return (1 if s[0] == 'e' else 0) + bad(s)
```

Alternate implementation

```python
def num_es2(s):
    """Returns: number of 'e's in <s>. Precond: <s> a string""
    # Strategy: break into two smaller strings, recur on both.
    # base case: cannot break into two smaller strings
    if s == "":
        return 0
    if len(s) == 1:
        return 1
    if len(s) <= 2:
        return 1
    # recursive case: choose a random breakpoint
    i = random integer between 1 and len(s)-1, inclusive
    return num of e's from 0 to up to but not including i, plus num of e's from i to the end of the string
```

Example: Palindromes

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

```
AMANAPLANACANALPANAMA
```

- All strings with fewer than 2 characters are palindromes

Practical application: RNA secondary structure:
loops form because of "antipalindromes" (G/C and A/U)

Example: Reversing a String

**Precise Specification:**
- Returns: reverse of s

**Solving with recursion**
- Suppose we can reverse a smaller string
  (e.g., one fewer character)
- Can we use that solution to reverse whole string?

```
> e i l o H
```

```
> o l i e H
```

```
> e I l o
```

```
> o I i e
```