Announcements

**Prelim preparation**
Study suggestion: be able to re-do labs 2-5 and A1, A2, A3 on paper and without much hesitation.

For help on A2: try using the Online Python Tutor.

Fall 2012 prelim 1 and review material is/will be posted on the exams page.

**Organization suggestion**
Get a three-ring binder and a 3-hole punch. Use these oldie-but-goodie technologies to store your CS1110 handouts 'n stuff.
Many recursion examples on the Lectures page
These were authored by Prof. Walker White last semester.

- comments in braces are *assertions*: conditions assumed to hold if that line is reached. Example:
  
  ```python
  # {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:
  
  ```python
  assert type(s) == str, `s` + ' is not a string'
  ```
  (backquotes give unambiguous string representation)
def num_es(s):
    """Returns: number of ‘e’s in <s>. Precond: <s> a string"""
    # Strategy: break off first character, recur on the rest.
    if s == "":  # base case (no recursion): <s> is empty string
        return 0
    # recursive case: process 1st char and rest of string
    # note: s[1:] is " if len(s) <= 1
    return ((1 if s[0] == 'e' else 0) + num_es(s[1:]))

Let's understand what happens at execution.
Execution in "typical" recursion case

inside module lec 12

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

code with function call

```python
import lec12
print lec12.num_es('ae')
```

import creates the function objects that are defined in lec12, like `num_es`, so we can call them.
What if we didn't recur on a "smaller" value?

Hypothetical function definition

def bad(s):
1 if s == ''
2 return 0
3 return ((1 if s[0] == 'e' else 0) + bad(s))

code with function call

print bad('ae')

[lots of copies of the same message]
RuntimeError: maximum recursion depth exceeded
What if we didn't have a base case (version I)?

hypothetical function definition

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

code with function call

```python
print oops('ae')
```

error (index error) when splitting not applicable
What if we didn't have a base case (version II)?

hypothetical function definition

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

code with function call

```python
print oops('ae')
```

[ lots of copies of the same message ]

RuntimeError: maximum recursion depth exceeded
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 == '':
        ...
    elif len(s) == 1:
        ...
    else:
        ...

    # 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
   Implement the recursive case (leave base case for later)
```
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 len(s) <= 1:
        return 1 if s == 'e' else 0
    # recursive case: choose a random breakpoint
    i = random.randrange(1:len(s) - 1)
    return num_es2(s[:i]) + num_es2(s[i:])
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):
   - When the recursive case doesn't apply
   - When the argument values are as "small" as possible
   - When the answer is determined with little calculation.
Example: Palindromes

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

- All strings with fewer than 2 characters are palindromes

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

(see posted code for solutions)

def ispalindrome(s):
    """Returns: True if string s is a palindrome, False otherwise"""

    # can check 1st against last character, then interior
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?
Example: Reversing a String

```python
def reverse(s):
    """Returns: reverse of s
    Precondition: s a string"
    if s == 
        return
    return reverse(s[1:]) + s[0]

def reverse2(s):
    """Returns: reverse of s
    Precondition: s a string"
    if s == 
        return
    return reverse2(s[1:]) + s[0]
```

Can you fill in the missing lines?
(see posted code for solutions)