Foundations of Programming

Recursion Practice

Announcements

- CSE 8A vs CSE 11 -- What's the story and how to choose?

Learning outcomes/key ideas

- Practice with recursion
- More list/string manipulation functions
- Lab04 walk through

Let recursion do the work for you.

Exploit self-similarity

Produce short, elegant code

} **Less work !**

```
def fac(N):
    if N <= 1:
        return 1
    else:
        rest = fac(N-1)
        return rest * N
```

You handle the base case – the easiest case!

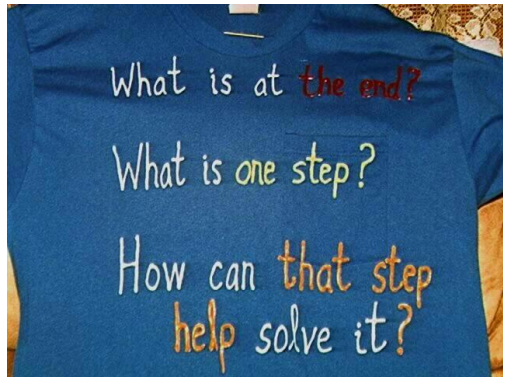
Recursion does almost all of the rest of the problem!

You specify one step at the end

But you ***do*** need to do one step yourself…

```
def fac(N):

    if N <= 1:
        return 1
    else:
        return fac(N)
```



Breaking Up…

is ***easy*** to do with Python.

```
s = "this has 2 t's"
```

How do we get at the *initial character of s?*

How do we get at ALL THE REST *of s?*

Recursion Examples

```
def mylen(s):
    """ input: any string, s
        output: the number of characters in s
    """
```

Recursion Examples

```
def mylen(s):
    """ input: any string, s
        output: the number of characters in s
    """
    if s == '':
        return
    else:
        return
```

Recursion Examples

```
def mylen(s):
    """ input: any string, s
        output: the number of characters in s
    """
    if s == '':
        return 0
    else:
        return 1 + mylen(s[1:])
```

What will this do?

```
def mylen(s):
    """ input: any string, s
        output: the number of characters in s
    """
    if s == '':
        print 0
    else:
        print 1 + mylen(s[1:])
```

A. Work correctly, just like before
B. Print the length of the string correctly, but return nothing
C. Print several values for the length of the string, and return the correct answer
D. Cause an error
E. Something else

What will this do?

```
def mylen(s):
    """ input: any string, s
        output: the number of characters in s
    """
    if s == '':
        return 0
    return 1 + mylen(s[1:])
```

A. Work correctly, just like before
B. Return an incorrect answer
C. Cause an error
D. Something else

Recursion Examples

```
def sajak(s):
    """ input: any string, s
        output: the number vowels in s
    """
```

```
def isVowel(letter):
    """ input: a string of length 1, letter
        output: True if the letter is a vowel, otherwise False
    """
    if letter in 'aeiou':
        return True
    else:
        return False
```

Can you make isVowel shorter?
Hint: `letter in 'aeiou'` is a boolean expression

Recursion Examples

```
def sajak(s):
    """ input: any string, s
        output: the number vowels in s
    """
```

```
def isVowel(letter):
    """ input: a string of length 1, letter
        output: True if the letter is a vowel, otherwise False
    """
    return letter in 'aeiou'
```

Recursion Examples

```
def censor( word, forbidden ):
    """ input: any string, word; a string containing
               forbidden characters
        output: the word with its forbidden characters
                replaced with *'s.
    """
```

```
>>> censor( "sip", "ip" )
's**'
>>> censor( "seven", "e" )
's*v*n'
```

Recursion Examples

```
def recFind(theList, value):
    """ input: a list, a value to find in the list
        output: the first index in the list where the value was found,
                or -1 if the value is not in the list
    """
```

```
>>> myList = [2, 3, 4, 3, 5, 3]
>>> recFind(myList, 3)
1
>>> recFind(myList, 10)
-1
>>> recFind(myList, 2)
0
```

Can you return the LAST index?

In preparation for lab04

```
def drawSquareRec(theTurtle, sideLen):
    """ input: a turtle, the length of the side
        output: returns nothing. Uses theTurtle to draw a square
                with the given side length
    """
```

Recursion Examples

```
def recFind(theList, value, maxIndex):
    """ input: a list, a value to find in the list, the maximum Index
               to look at
        output: the LAST index in the list where the value was found,
                or -1 if the value is not in the list
    """
```

```
>>> myList = [2, 3, 4, 3, 5, 3]
>>> recFind(myList, 3, 5)
5
>>> recFind(myList, 3, 4)
3
>>> recFind(myList, 10)
-1
```

Recursion Examples

```
def bRecFind(theList, value, low, high):
    """ input: a list, a value to find in the list, the range in the list
               where you are allowed to look (between low and high)

        output: the index in the list where the value was found,

                or -1 if the value is not in the list

    """
```