Topics for Today

- Project 1: questions?
- Clicker scores clarification
- Introduction to recursion
- Recursive definitions
- Writing recursive functions

Reading assignment: Zelle, Chapter 13.2
What is recursion?

- A description of something that refers to itself is called a \textit{recursive} definition.
- A function calling itself is called a \textit{recursive} function
- Why use recursive definitions and functions?
  - Recursion allows simple definitions
  - Powerful programming technique
- Recursive programming is directly related to mathematical induction, a proof technique used often to prove facts about discrete functions
Recursive definition: Factorial

- Factorial

\[ n! = \begin{cases} 
1 & \text{if } n = 0 \\
n(n-1)! & \text{otherwise} 
\end{cases} \]

\[ n! = n(n-1)(n-2)\ldots(1) \]

- \( \text{fac}(n) = n \times \text{fac}(n-1) \)

- Recursive definitions need a termination condition
  \( \text{fac}(0) = 1 \)
def **non-rec-fact**\(\)(n)  
    fact = 1  
    for factor in range(n, 1, -1):  
        fact = fact * factor  
    return(fact)  

def **fact**\(\)(n):  
    if n==1:  
        return 1  
    else:  
        return n*fact(n-1)
Recursive fact(n)

def fact(n):
    if n==1:
        return 1
    else:
        return n*fact(n-1)

fact(5)
    fact(4)
        fact(3)
            fact(2)
                fact(1)
                    return 1
                return 2*1 = 2
            return 3*2 = 6
        return 4*6 = 24
    return 5*24 = 120

Not a particularly useful program as the iterative version is more efficient and n! grows too quickly to use in computations
Recursive definitions: Fibonacci

- **Fibonacci Numbers**
  - \( F(n) = F(n-1) + F(n-2) \), \( F(0)=0 \) and \( F(1) = 1 \)
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711 …
  - named after Leonardo of Pisa, but first described by an Indian mathematician (about 300BC)
  - Closed form solution exists
    \[
    F(n) = \frac{\varphi^n - (1 - \varphi)^n}{\sqrt{5}} = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}},
    \]
    where \( \varphi \) is the golden ratio; i.e., a root of \( x^2 = x + 1 \),
Clicker Question 1 (part. only)

What is the solution to the following recursive definition?

\[ F(n) = F\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right); F(1) = 0 \]

A. It is approximately \(n/2\)
B. It is approx. \(\log n\) (base 2) - answer
C. None of the listed options
D. It is approximately \(\sqrt{n/2}\)
Recursive functions

- Euclid’s algorithm
- String reversal
- Searching in a sorted list/array
- Fast exponentiation
- Recursive graphics
- Paths searching in a grid
- SUDOKU
Example: Euclid’s GCD Algorithm

def nonrec_gcd(m, n):
    # assume m<=n
    while m != 0:
        n, m = m, n % m
    return n

def gcd(m, n):
    # assume m<=n
    if m == 0:
        return n
    else:
        return gcd(n % m, m)
Recursive gcd(m,n)

def gcd(m,n):
    if m == 0:
        return n
    else:
        return gcd(n%m, m)

euclid.py
Example: String Reversal

- Python lists have a built-in method that can be used to reverse a list: L.reverse()
- Reverse a string as follows:
  - convert the string into a list of characters
  - reverse the resulting list
  - convert the new list back into a string
Review basic string operations

- **Concatenation**
  - “CS” + “190C” gives “CS190C”

- **Repetition**
  - “CS”*3 gives “CSCSCS”

- **Indexing and slicing**
  - s= “computation” : s[1] is “o”
  - s[0:3] is “comp”, s[3:] is “putation”

- **Length**
  - len(“computation”) = 11

- **Iterating** through the characters of a string
  - for ch in “computation” :
Idea of recursive reversal (1)

- Goal is to use recursion to reverse a string without the intermediate list step:
  - Divide the string up into a first character and “all the rest”
  - Reverse the “rest” and append the first character to the end of it
- Recursion needs to terminate
  - choose empty string as the termination condition
Idea of recursive reversal (2)

```python
def reverse(s):
    return reverse(s[1:]) + s[0]
```

- The slice `s[1:]` returns all but the first character of the string.
- We reverse this slice and then concatenate the first character (`s[0]`) onto the end.
Recursive String Reversal

def reverse(s):
    if s == "":
        return s
    else:
        return reverse(s[1:])+s[0]

>>> reverse("Hello")
'olleH'
Recursion at work

Reverse(spring)

reverse(pring)+s

(reverse(ring)+p)+s

(((reverse(ing)+r)+p)+s

(((reverse(ng)+i) +r)+p)+s

((((reverse(g)+n) +i) +r)+p)+s

((((reverse()+g)+n) +i) +r)+ p)+s

(((({ +g) )+n) +i) +r)+ p)+s

((((g +n) +i) +r)+ p)+s

((((gn + i) +r ) + p) + s

((gni + r ) + p) + s

((gnir + p) + s

(gnirp + s

gnirps

2/4/2008
Clicker Question 2

```python
>>> myL = "rotavator"
>>> print reverse(reverse(myL)[2:])
```

A. tavator  
B. rotavat - answer  
C. rot  
D. tor
Searching

*Searching*: look for a particular value in a collection.
- It is a basic operation you already used
- Python provides a number of built-in search-related methods
- A search returns
  - -1 if the element is not found
  - the position of the element in the collection if it is found
Hi,

I’ve got a list with more than 500,000 ints. Before inserting new ints, I have to check that it doesn’t exist already in the list.

Currently, I am doing the standard:
if new_int not in long_list:
    long_list.append(new_int)

but it is extremely slow… is there a faster way of doing this in python?