Topics for Today

- Exam 1
  - Wed, Feb 13, 7:00-8:00 PM
  - MJIS 1001 (Biomedical Engineering Bld.)
  - Closed book; no calculators
- Office hour updates pre exam
  - Aaron Lint: Wed, 11:30-12:30, in class
  - Susanne Hambrusch: Mon, 2:30-3:30 (and class time Wed)
  - Tim Korb: Tue, 2:30-3:30, in lab
  - John Valko: Tue, 1:30-2:30
  - Tony Hosking: Tue, 2:00-4:00 (also has 352 traffic)
- Exam review suggestions and Python examples

Reviewing for the Exam (1)

- Slides from class (at class website)
- Zelle, Appendix A, Python Quick Reference
- Python language features to know…
  - assignment statements: simultaneous assignments
  - if statement: matching else clauses
  - for loops: iterables
  - expressions: operator precedence, parentheses, function calls
  - range function: 3-parameter version
  - string operations: concatenate, replicate, index, slice, len(), iterate (with for)
Reviewing for the Exam (2)

- More Python language features to know…
  - type conversions: float, int, str
  - functions: defining, calling, parameter passing, returning
  - relational operators: \( a < b < c \)
  - logical operators: and, or, not
  - Boolean values: True, False
  - Special value: None
  - lists: subscripting, slicing, concatenation, etc., but not “dot methods”
  - arrays: create, zeros, ones, append

Reviewing for the Exam (3)

- Binary arithmetic: word length example from slides
- Simple programs: defs, loops, tests
- Lists of lists: ala percolation problem
- Tracing recursive function execution

- Not on the exam:
  - I/O operations
  - Vpython
  - Matplotlib
  - No syntax details
Review of programs

Recursion examples
- Fast exponentiation
- Summing lists element (Lab 5, in-lab problem)

Problem Set 2, problem 3:
- String to list to array: rotating and shuffling

Fast Exponentiation

- Compute \( a^n \) using recursion
- We know that \( 2^8 = 2^4 (2^4) \).
  - If we know \( 2^4 \), we can calculate \( 2^8 \) using one multiplication.
  - How is \( 2^4 \) computed? Using \( 2^2 \) and one multiplication
  - How is \( 2^2 \) computed? Using 2 and one multiplication
  - We can calculate \( 2^8 \) using only three multiplications!

\[
a^n = \begin{cases} 
  a^{n/2} (a^{n/2}) & \text{if } n \text{ is even} \\
  a^{n/2} (a^{n/2})(a) & \text{if } n \text{ is odd}
\end{cases}
\]
Fast Exponentiation

def recPower(a, n):
    # raises a to the n-th power
    if n == 0:
        return 1
    else:
        factor = recPower(a, n/2)
        if n%2 == 0:  # n is even
            return factor*factor
        else:  # n is odd
            return factor*factor*a

Use variable factor so that we don’t calculate \( a^{n/2} \) more than once

Lab 5: Recursive sum of a list

- Given either a nested list of numbers, compute the sum of all the elements in the list
- Function has one argument and returns an integer.
- Base case of recursion: parameter is an integer

```python
>>> recsum([1,2,3])
6
>>> recsum([1,2,[3,4,5]])
15
>>> recsum([1,[4,5]],-4,[6,[7],[9,10]],0])
38
```
def recsum(x):
    if type(x) != list:
        # check if recursion should terminate
        return x
    else:
        total = 0
        for y in x:
            # make a recursive call on each list element
            total = total + recsum(y)
        return total

Problem Set 2, problem 3

• Read two strings of equal length and store them in arrays, A and B
• Read an integer k
• Rotate elements in array A k places to the right (with wrap-around)
• Rotate elements in array B k places to the left (with wrap-around)
• Combine the two arrays into one of twice the size by “shuffling” the elements
From posted solution

\[ A, B = \text{array(list(a)), array(list(b))} \]

# Rotate the arrays
\[ A = \text{append(A[len(A)-k:],A[:len(A)-k])} \quad \# \text{k positions to right} \]
\[ B = \text{append(B[k:],B[:k])} \quad \# \text{k positions to left} \]

# Combine arrays A and B through shuffling
\[ C = \text{array[]} \]
\[ \text{for } i \text{ in range(0, len(A))}: \]
\[ \quad C = \text{append(C, append(A[i],B[i]))} \]

AA = array(list(a))
BB = array(list(b))
CC = array([" "]*(2*n))

# Rotate using helper arrays
HA, HB = array([" "]*n), array([" "]*n)
\[ \text{for } i \text{ in range(n):} \]
\[ \quad HA[i] = AA[i-k] \]
\[ \quad HB[i] = BB[i-n+k] \]
AA = HA
BB = HB

# Combine
\[ \text{for } i \text{ in range(n):} \]
\[ \quad CC[2*i] = AA[i] \]
\[ \quad CC[2*i+1] = BB[i] \]
A few array operations

Creating an array
- A = zeros(5); B = ones(10)
- C = arange(12); D = array([1,2,3,4])

Combining arrays
- C = append (A,B)

Operations on arrays
- C = A+B; A**2; etc
- A = B; A != B

Good tutorial:
http://www.scipy.org/Tentative_NumPy_Tutorial