A201/A597 Introduction to Programming I

First Summer 2007

Lecture Notes for Monday June 4, 2007 (ED1204)

Today: functions for hierarchical design.

Think how you describe a human.

You will use words such as: head, hands, body, legs.

Only when you describe head will you mention: eyes, nose, brain, mouth, ears.

Same goes for: lungs, liver, heart etc.

Each of them is as complex as anything you're likely to see.

For description (knowledge organization) purposes we label the components and describe them separately. This is an attempt to overcome complexity.

Here's a program like your new homework assignment:

```
def process(letter):
    global already
    if letter in already:
        print "You have tried this before."
        return mistakes, mask # both read-only
    else:
        already += letter
        newMask = ""
        if letter in secret:
            for index in range(len(mask)):
                if secret[index] == letter:
                    newMask += letter
                else:
                    newMask += mask[index]
            return mistakes, newMask
        else:
            return mistakes + 1, mask
```

The function above is a bit contrived.

It's meant to exemplify and illustrate beyond good design.

So it's demonstrating the use of global variables.

Note that variables outside are read-only by default.

It also demonstrate how a function can return more than one value.

The function is used in this program:

```
secret = "nectarine"
mask = "-" * len(secret)
mistakes = 0
already = ""

while mistakes < 6 and mask != secret:
    print str(mistakes) + ": " + mask
    letter = raw_input("Guess: ")
    (mistakes, mask) = process(letter)

if mistakes == 6:
    print "Sorry, the secret word was:", secret
else:
    print "Well done, with only", mistakes, "mistakes"
```

Hopefully you will agree this program is now a lot clearer?

Much complexity has been delegated to the `process` function.

Minute paper: will be announced in class.

Homework Assignment Eight: Designing and Implementing Functions.

Define the missing functions (identified in red below) needed by the following (partial) implementations for the Game of Nim and Addition Quiz.

1. Game of Nim program:

```
height = int(raw_input("Enter height: "))

won = false

while not won:
    (height, valid, won) = userMove()
    if valid:
        if won:
            print "Congratulations, you won."
            continue
        else:
            report(height, "computer's turn")
            (height, won) = computerMove()
    else:
        print "User loses, due to error."
        won = True
        continue
    if won:
        print "Computer won."
        continue
    else:
        report(height, "user's turn")
```

2. Addition Quiz program:

```
import random

limit = 0

while limit > 0:
    n1 = generateNumber()
    n2 = generateNumber()
    answer = askQuestion(n1, n2)
    if answer == 'hard':
        continue
    elif answer == 'quit':
        break
    else:
        limit -= 1
    if answer == n1 + n2:
        (good, total) = (good+1, total+1)
        print "Great job."
    else:
        total += 1
        print "Nope, the answer was:", (n1 + n2)
report(limit) # sees good and total, can't change them
```