More on Functions
& Intro to VPython
Chapter 6 and beyond

Functions, again

Functions capture reusable program fragments

Usually they define an abstraction:

- increment: def inc(x): return x+1
- scale: def scl(x, f): return x*f

Of course, functions usually capture much more interesting computations

Functions and Style

- Functions should be used carefully so that program meaning is made clearer
- Overuse of functions can obfuscate, or obscure, the program meaning
- Try to use a consistent naming scheme
  - in_rect, readPoints, ShowLines
- Use direct verb names: drawRectangle

Functions and Style

- Parameters should be meaningful
- Parameters should capture logical varying inputs
- Avoid unrelated/unexpected/surprising side-effects
Functions and Style

- Avoid undergeneralizing: try to make a function broadly reusable
- Avoid overgeneralizing: Swiss-Army Chainsaws are dangerous

Default arguments

- Python functions can have default arguments, evaluated in the defining scope

```
i = 5
def f(arg=i):
    print arg
i = 6
f()
```

Keyword arguments

- Functions can be called with arguments by keyword instead of position
- Any positional arguments must come before keyword arguments

```
def f(a, L=[]):
    L.append(a)
    return L
print f(1)
print f(2)
print f(3)
```
### Function arguments

- Functions can take functions as arguments
- "Higher-Order" functions

```python
def f(g, x):
    return g(x)

def inc(x):
    return x + 1

f(inc, 2)
```

### VPython

- A Python graphics module for modeling and simulation
- VPython = Python + IDLE + visual
- from visual import *

### VPython Visual Objects

- VObjects exist for program duration
- VObjects are displayed on the display window
- VObjects have attributes: pos, color, length/height/width/radius, etc
- Changing attributes changes display
Other VPython Objects

These are not displayed

A vector object supports the usual:
- mag, mag2 (mag squared), norm (normalized), cross, dot, rotate, etc
from visual import *
ball = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
wallR = box(pos=(6,0,0), size=(0.2,12,12), color=color.green)
wallL = box(pos=(-6,0,0), size=(0.2,12,12), color=color.green)
dt = 0.05
ball.velocity = vector(2,1.5,1)
ball.trail = curve(color=ball.color)
while 1:
    rate(100)
    ball.pos = ball.pos + ball.velocity*dt
    if ball.x > wallR.x:
        ball.velocity.x = -ball.velocity.x
    if ball.x < wallL.x:
        ball.velocity.x = -ball.velocity.x
    bv.pos = ball.pos
    bv.axis = ball.velocity
    ball.trail.append(pos=ball.pos)

from visual.graph import *
Graph example

VPython has a powerful graph plotting subsystem

from visual.graph import *

VPython Graphs

Lines (gcurve), marks (gdots), bars (gvbars, ghbars), bins (ghistogram)