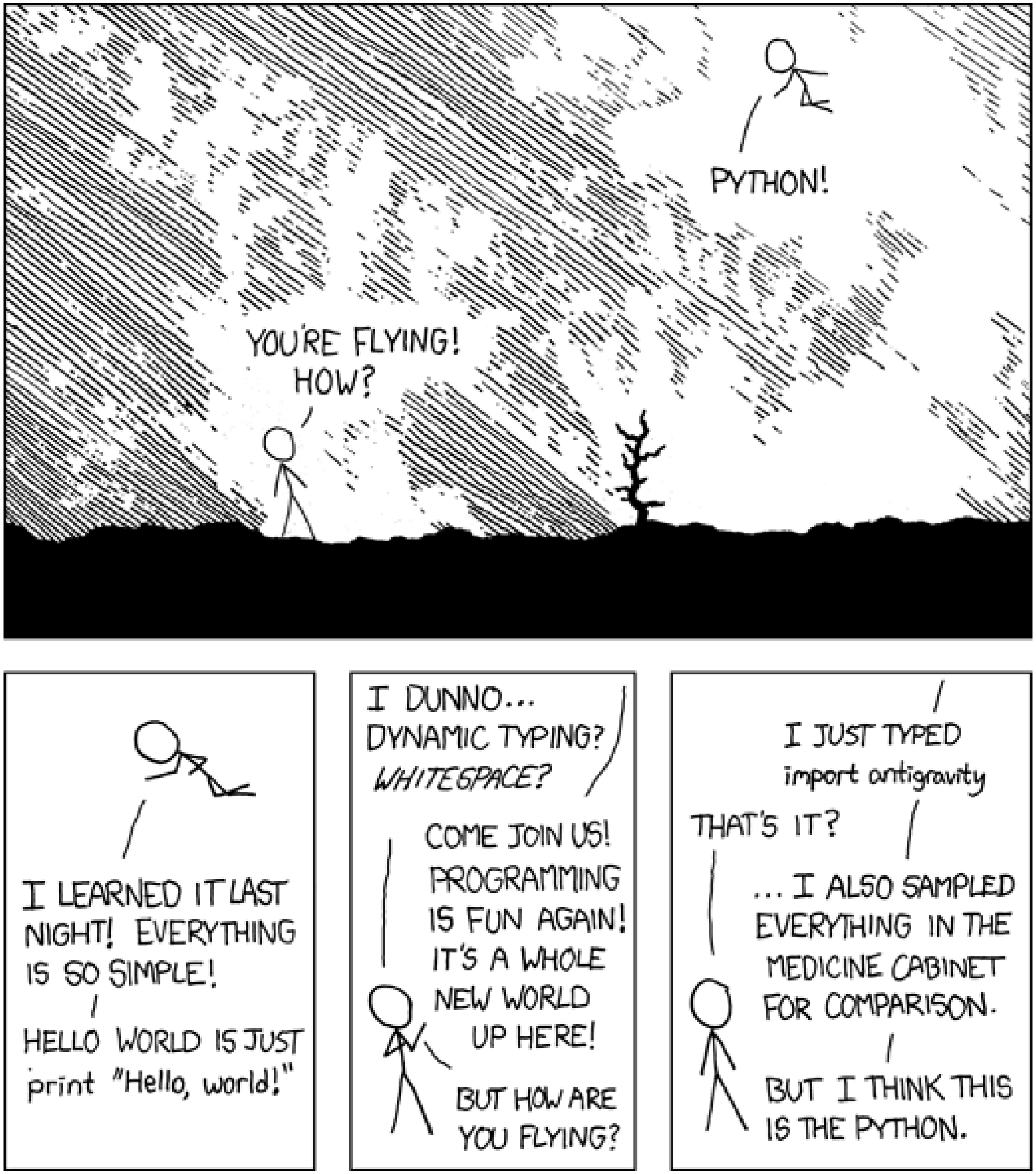
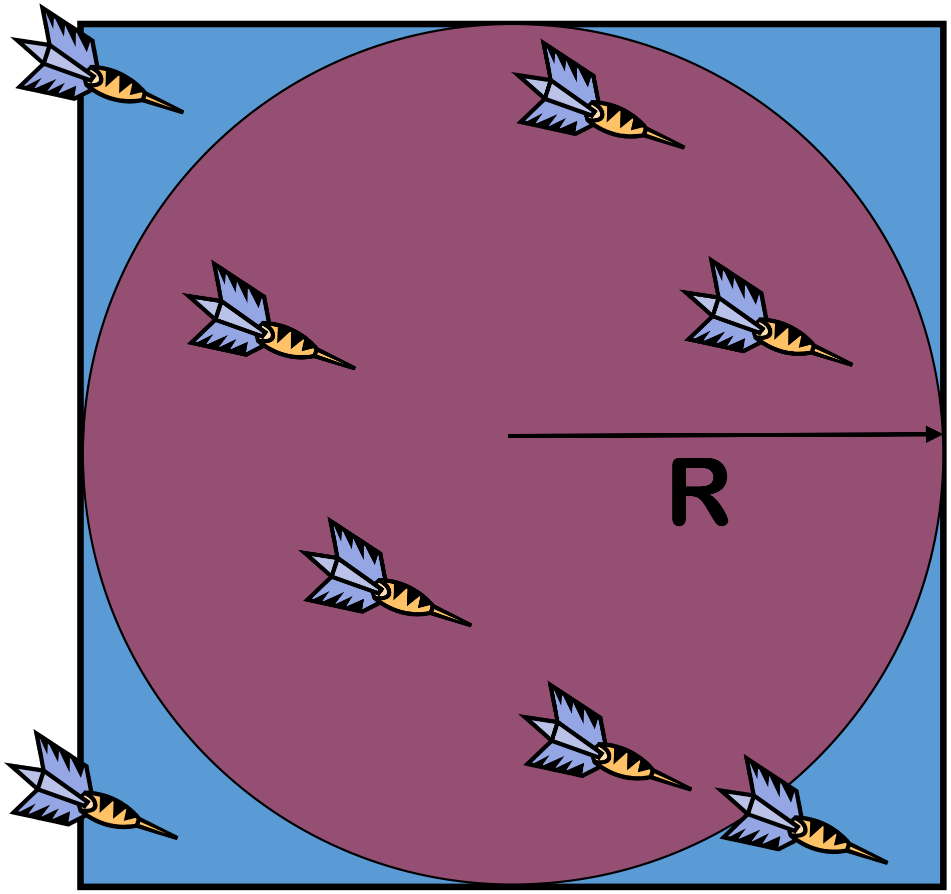
ICGE Module 2 Session 3—Python!


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

https://xkcd.com/353/

Monte Carlo simulations use random numbers to statistically sample different outcomes

A simple use of Monte Carlo simulation is to calculate the relative area of a region:



What's the ratio of the area of the circle to the area of the square?

$$\frac{\pi \mathbf{R}^2}{(2\mathbf{R})^2} = \frac{\pi}{4} = \frac{\text{Darts in circle}}{\text{Darts in square}}$$

$$\pi = 4 \times \frac{\text{Darts in circle}}{\text{Darts in square}}$$

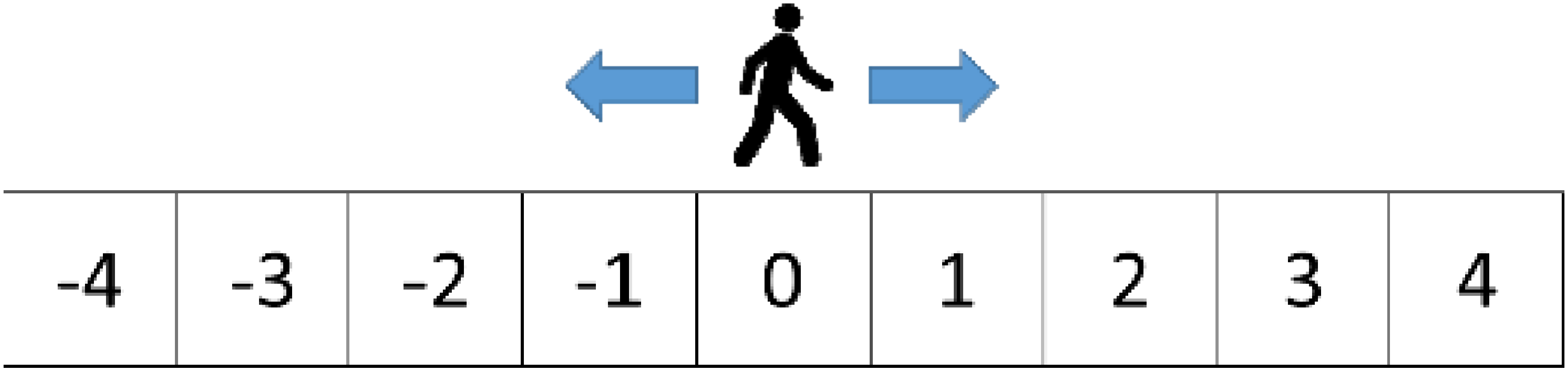
Here's a simple Python program to simulate this process using virtual darts

```
#!/usr/bin/python
import random
import math

inside=0
trials=1000
for i in range(trials):
    x=random.random()
    y=random.random()
    if (x*x+y*y)<1.0:
        inside+=1

pi=4.*float(inside)/float(trials)
print "N=%d Error=%8.5f "%(trials,pi-math.pi)
```

Indentation matters!

Enter into an idle editing window and then save as "**pi.py**"

Many problems in physics, chemistry and biology essentially boil down to "random walks"



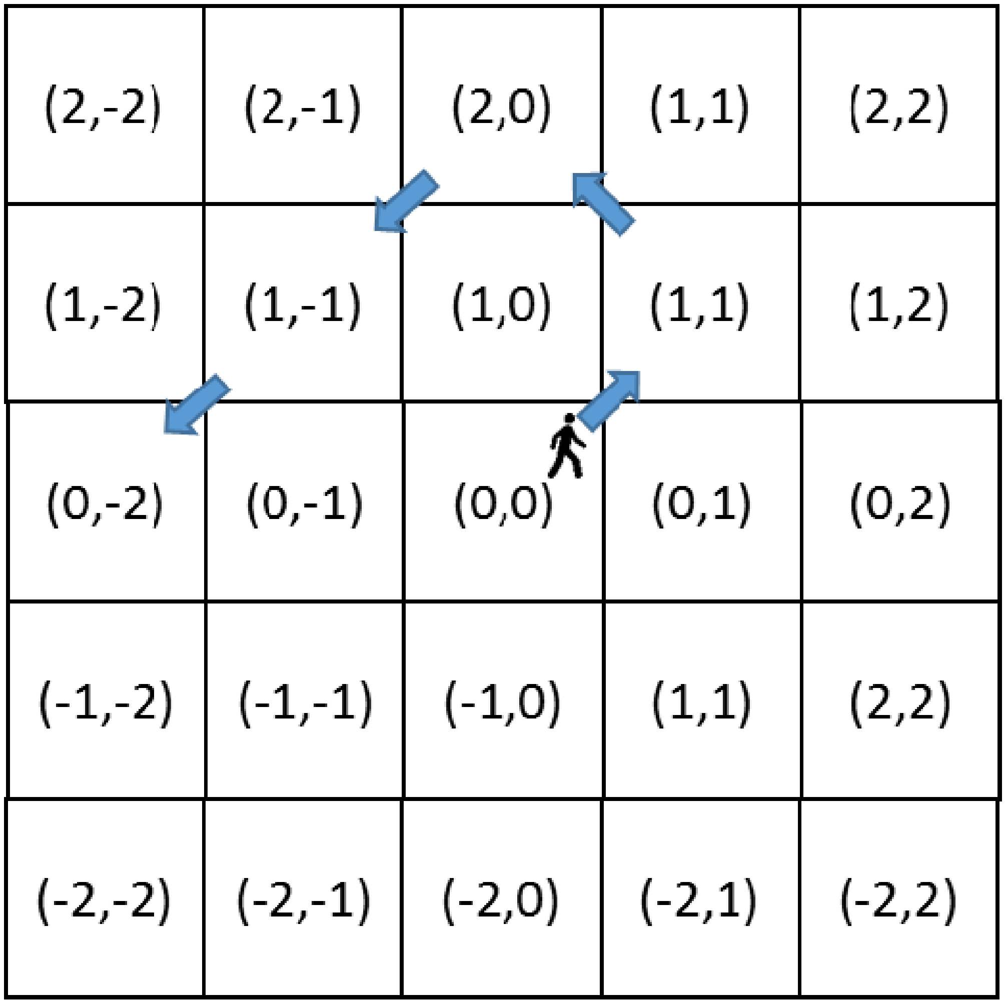
Simplest question to ask—does the walker ever get back home?

Python program for 1-D random walk

```
from __future__ import division
from random import choice
trials=1000
steps=1000
gothome=0
for i in range(trials):
    point=0
    for step in range(steps):
        point+=choice((-1,1))
        if point==0:
            gothome+=1
            break
print "Fraction that got home=%f" % (gothome/trials)
```

Save this as “**rwalk1d.py**” and run for different numbers of steps

The problem gets more interesting if the walker moves in 2 or more dimensions



(2,-2)	(2,-1)	(2,0)	(1,1)	(2,2)
(1,-2)	(1,-1)	(1,0)	(1,1)	(1,2)
(0,-2)	(0,-1)	(0,0)	(0,1)	(0,2)
(-1,-2)	(-1,-1)	(-1,0)	(1,1)	(2,2)
(-2,-2)	(-2,-1)	(-2,0)	(-2,1)	(-2,2)

Note that if we randomly change both x & y coordinates by -1 or +1, the walker moves diagonally like a checkers piece.

Python program for arbitrary-D random walk making diagonal moves at every step

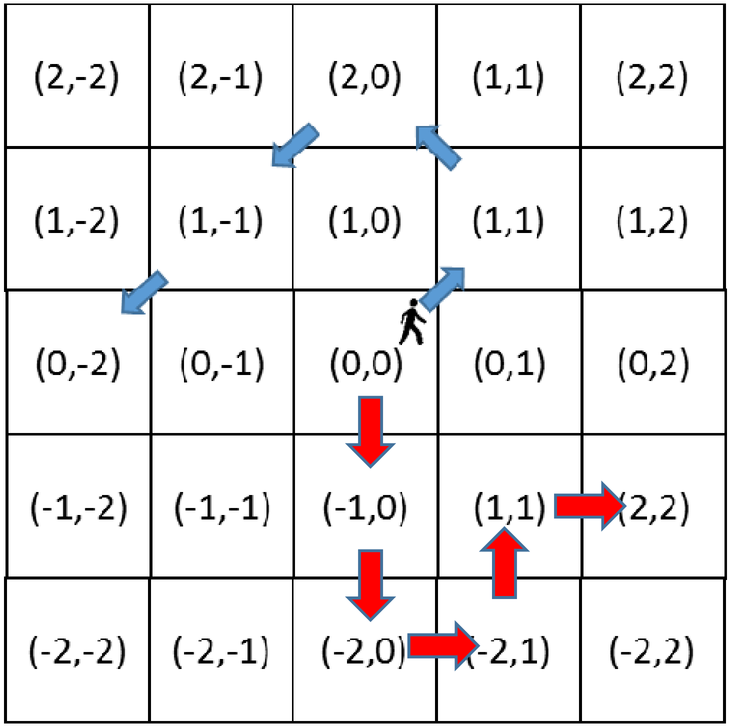
```
from __future__ import division
from random import choice
dim=3
trials=1000
steps=1000
gothome=0
for i in range(trials):
    point=[0]*dim
    for step in range(steps):
        for j in range(dim):
            point[j]+=choice((-1,1))
        if point.count(0)==dim:
            gothome+=1
            break
print "Fract that got home=%f in %d dims" % (gothome/trials,dim)
```

Save this as “rwalknd.py“ and run for different dimensions

For what #'s of dimensions does the walker make it home?

Exact results for an infinite number of steps moving in only one dimension at a time

Dimensions	Prob(get home)
1	1.000
2	1.000
3	0.341
4	0.193
5	0.135
6	0.105
7	0.086
8	0.073



(2,-2)	(2,-1)	(2,0)	(1,1)	(2,2)
(1,-2)	(1,-1)	(1,0)	(1,1)	(1,2)
(0,-2)	(0,-1)	(0,0)	(0,1)	(0,2)
(-1,-2)	(-1,-1)	(-1,0)	(1,1)	(2,2)
(-2,-2)	(-2,-1)	(-2,0)	(-2,1)	(-2,2)

Research questions:

1. How well does **rwalknd.py** agree with the exact results?
2. Do your results match better if you modify the program to step in only one dimension at a time?

Exact results from http://mathworld.wolfram.com/PolyasRandomWalkConstants.html