Python for Teachers

Pycon 2009, Chicago

Urner / Holden

Python for Teachers

vs.

Python for Trainers

(pedagogy vs. andragogy)

Town-Gown relations:

how to bridge Pointy Haired Bosses

vs.

Dilberts

True FOSS stories:

- FreeGeek and SQL Clinic
- Hillsboro Police and SA:
- OS Bridge (looking ahead)
- [any story you please]

Bug: Not Enough Stories (Lore)

Why?

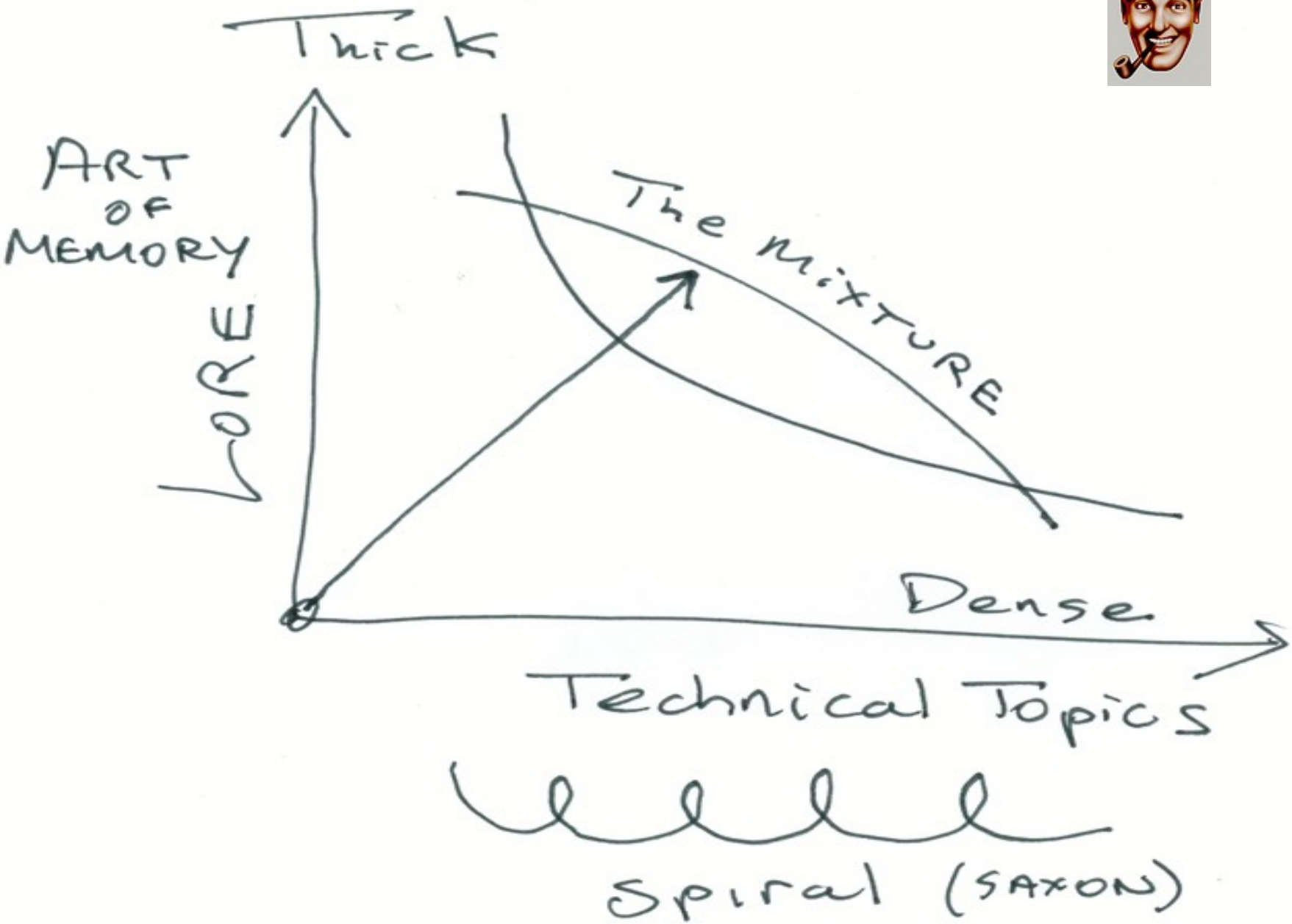
"Story problems should be Meaningless, the more meaningless the better."

Not!

Fix: Use Cases Matter,
we really do care about
the Real World and/or
about the Mnemonic
value of our stories.

Rule of Thumb: ok to tell fairy tales (science fiction OK), just don't waste our time (respect your audience)

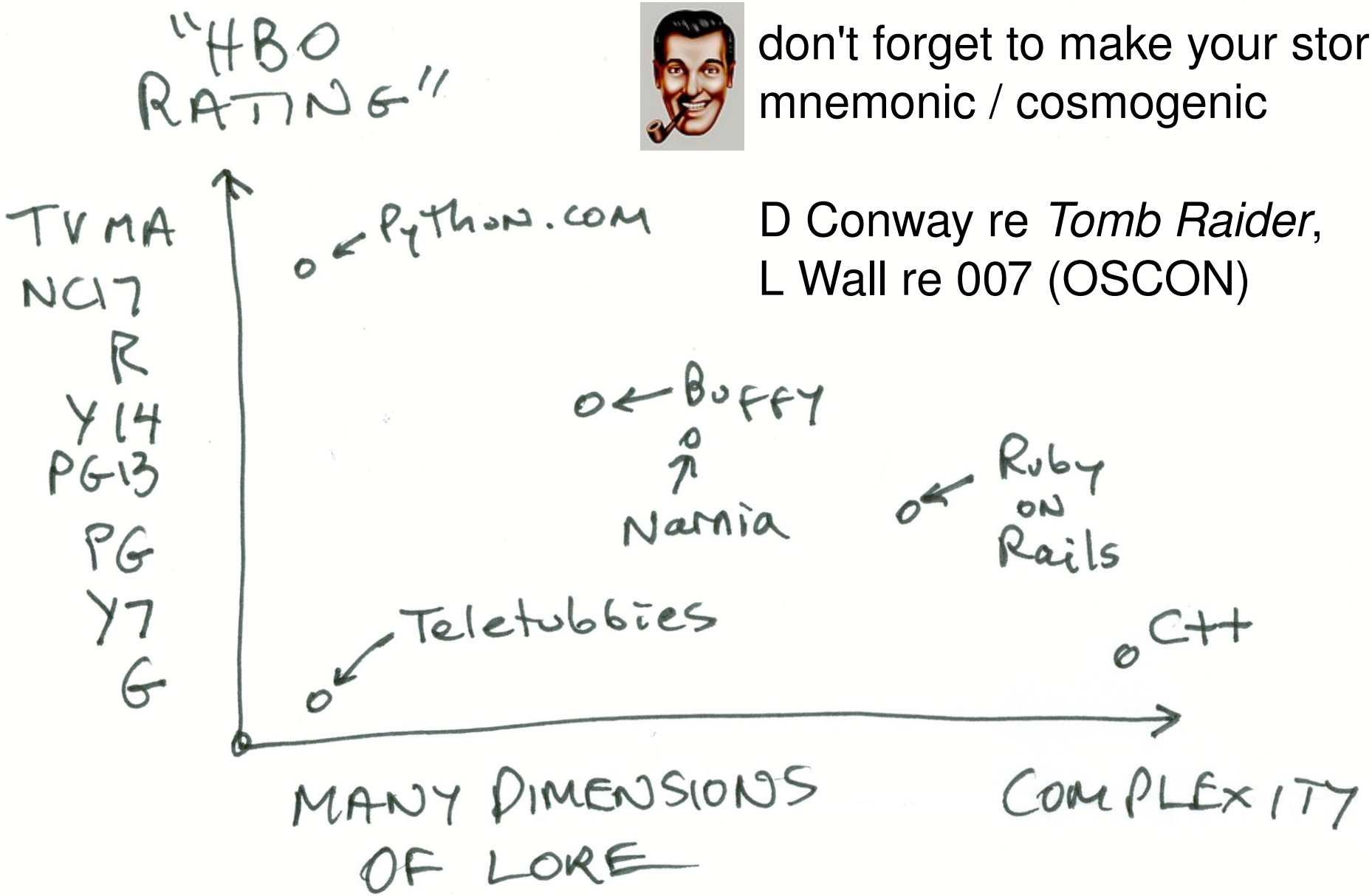
Remember: Neolithic Math (web frameworks, ORM divinities)



The phase space of lore is yours to explore (dreamweaving OK)

don't forget to make your stories mnemonic / cosmogenic

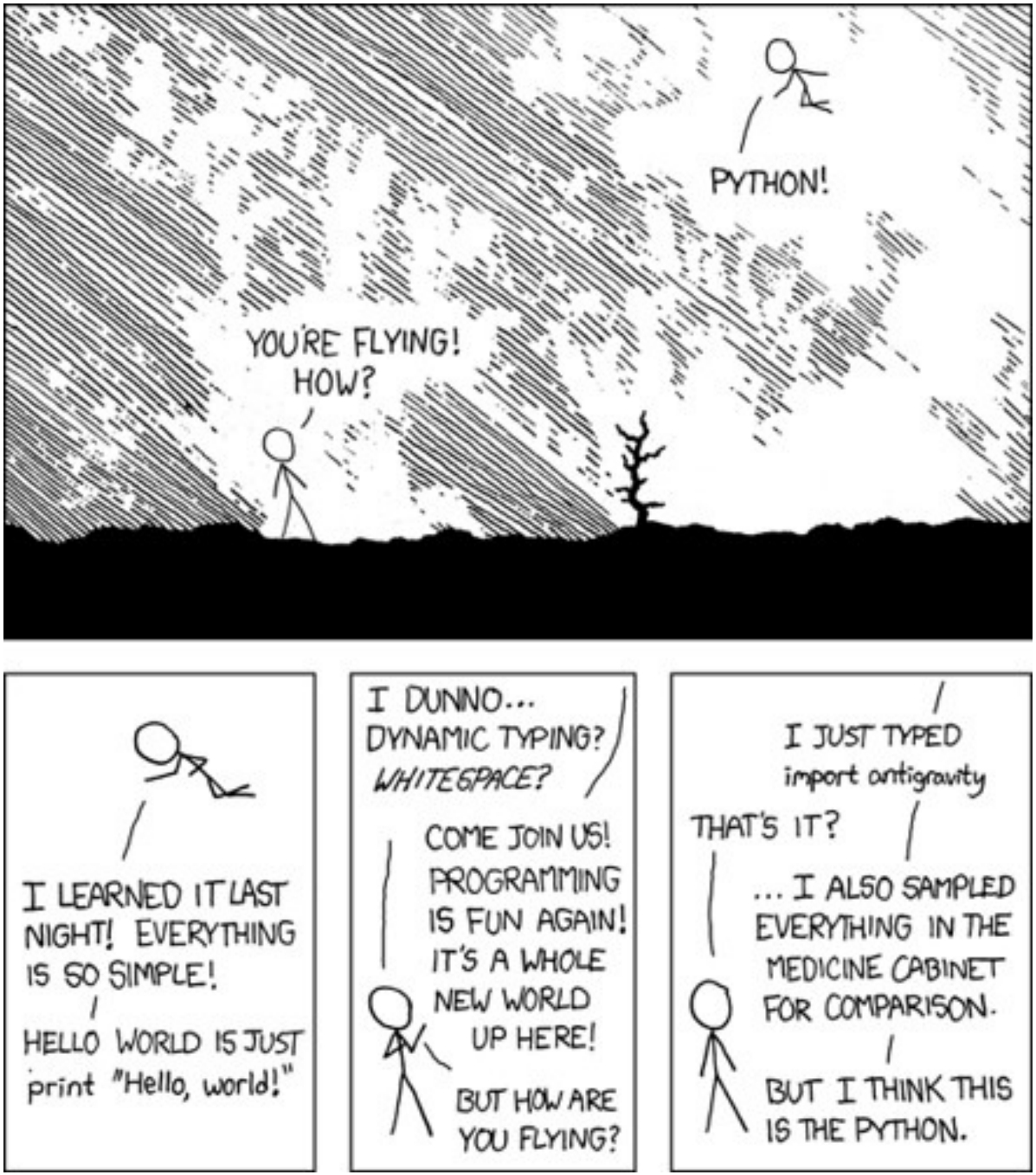
D Conway re *Tomb Raider*,
L Wall re 007 (OSCON)



Lore: Why am I here?

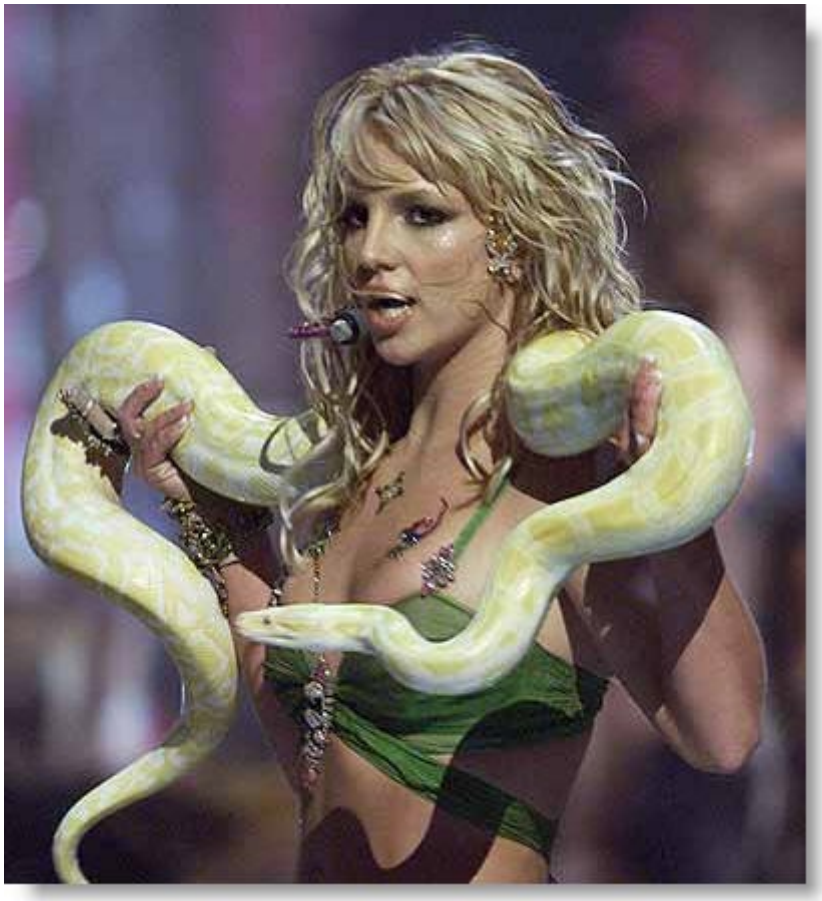
Skills: What am I doing?

xkcd:
adding spice

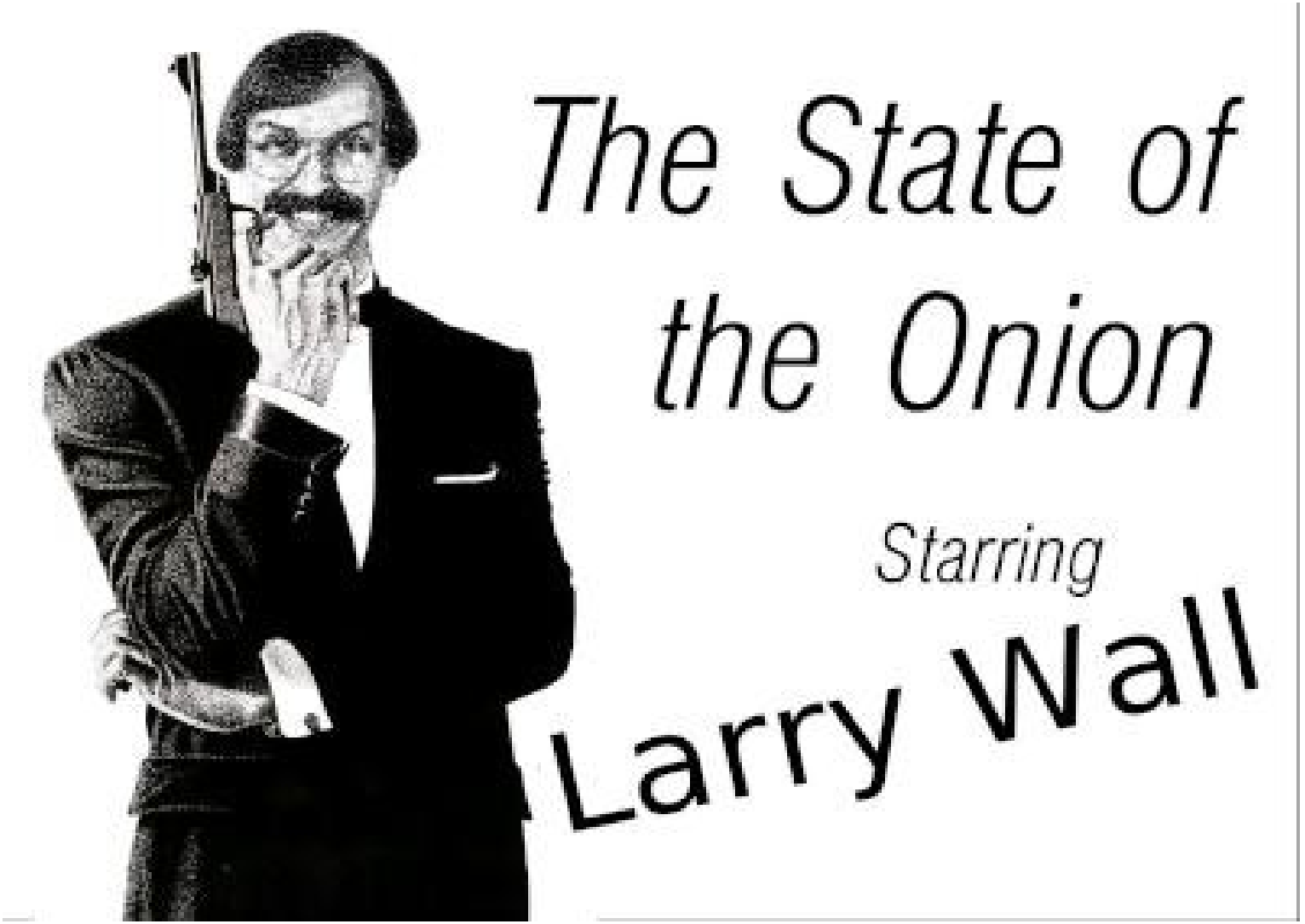


Subcultures R Us

Webus Vancouverous



Pythonista



http://www.perl.com/lpt/a/956

Where is Python Nation?

Guido: BDFL

... right next to
the Republic of Perl

Note Netherlands, quite near

The Rise of Structured Programming from the Spaghetti Code West

A more advanced civilization triumphs: the way of the Shaolin Kung Fu with David Carradine
http://www.imdb.com/title/tt0068093/

The Rise of SQL (keeping tabs)

A dark tale
http://www.waragainsttheweak.com/

The Rise of Unicode

Call me *Cosmopolitan*

File Edit View History Bookmarks Tools Help

http://www.blogger.com/post-edit.g?blogID=8204942&p

Getting Started | Latest BBC Headlines

Control Room | Blogger: Grain of Sand - ...

Grain of Sand

kirby.urner@gmail.com | Prietaisu skydelis | Pagalba | Atsijungti

Pranešimų rašymas | Nustatymai | Šablonas | (Peržiūrėti dienoraštį)

Sukurti | **Redaguoti pranešimus** | Redaguoti komentarus

Pavadinimas: Vilnius

Redaguoti Html | Sukurti

Peržiūrėti

```
<embed src="http://web.splashcast.net/go/so/2/p/KYNU4479UM" wmode="transparent" width="400"
height="300" allowFullScreen="true" type="application/x-shockwave-flash" ></embed>
Yikes, my Blogger control panel is in Lithuanian!  Today is a holiday and hardly anyone was
strolling in the rain early this morning.  I took a first few pix with my waterproof camera (my
wind breaker proved less than waterproof).

The main street next to the hotel is torn up for replumbing.  The hotel itself, Centrum Ratonda,
is quite satisfactory, with wireless in the room and an ample breakfast buffet (included).

Laura flies in late tonight.  I volunteered to help stuff bags and do other conference stuff.  My
hosts were very kind to fly me here.

All my puzzle pieces rejoined on this end minus the white 3-ring binder which I kindly (OK,
unintentionally) left next to 31A for Lufthansa management, or the dumpster as the case may be.

The Vilnius lost and found women kindly sent a query in teletype language to Frankfurt, using
```

Pranešimų nuostatos

žymės šiam pranešimui:
pvz. **motoroleriai, atostogos, ruduo**

Šaukiniai: paspauskite ***Ctrl*** su: *B* = **Paryškintas**, *I* = **Kursyvas**, *P* = **Publikuoti**, *D* = **Juodraštis** *daugiau*

ĮDĖTI PRANEŠIMĄ | IŠSAUGOTI JUODRAŠTĮ

Done

testuni.py - /home/kirby/testuni.p

File Edit Format Run Options Windows Help

```
class 艮:

    def __init__(self, 同人):
        self.同人 = 同人

    def __repr__(self):
        return "艮 @ %s" % str(id(self))
```

Ln: 5 Col: 0

```
IDLE 3.0a2
>>> from testuni import 艮
>>> themountain = 艮("Fuji")
>>> themountain
艮 @ 138188876
>>> themountain.同人
'Fuji'
>>>
```

Ln: 19 Col: 4

Public crypto: being private with strangers

One day, Bob had a secret for Alice, one that Eve should not know ...

GNU heart Linux
(a romantic epic)

Heroic
Epic
Genre

SCO versus IBM
(FOSS wins)

Back to the Future

or...

from __future__
import...

The Lost Generation

(thank you ABC.com)

Primordial Soup

- Atoms (primitive types)
- Molecules (data structures)
- Cells (functions, generators)
- Organisms ("am a")
- Organs ("have a")
- Ecosystems (name spaces)
- OK to keep zooming out

What about Hardware?

- Packets, tcp / ip
- *Warriors of the Net*
- Data centers, networks (GIS)
- Zoom in to CPU / RAM / ROM
- Note: XO has no hard disk
- OLPC vs. MDPA (control room)

Your job: train *House M.D.* in Python (a busy grumpy guy)

File Edit Format Run Options Windows Help

```python
"""module:  simplelife.py"""

class Biotum:

    def __init__(self, name):
        self.name = name
        self.stomach = []

    def __call__(self,food):
        self.stomach.append(food)

    def __repr__(self):
        return 'Biotum named ' + self.name
```

Ln: 1 Col: 0

File Edit Shell Debug Options Windows Help

```
>>> import simplelife
>>> imp.reload(simplelife)
<module 'simplelife' from '/home/ki
rby/simplelife.py'>
>>> from simplelife import Biotum
>>> cell1 = Biotum('xV')
>>> cell2 = Biotum('xY')
>>> cell1.stomach
[]
>>> cell1('**')
>>> cell1.stomach
['**']
>>> cell1('&&&')
>>> cell1.stomach
['**', '&&&']
>>> cell1(cell2)
>>> cell1.stomach
['**', '&&&', Biotum named xY]
>>>
>>> |
```

Ln: 55 Col: 4

Using mnemonics:

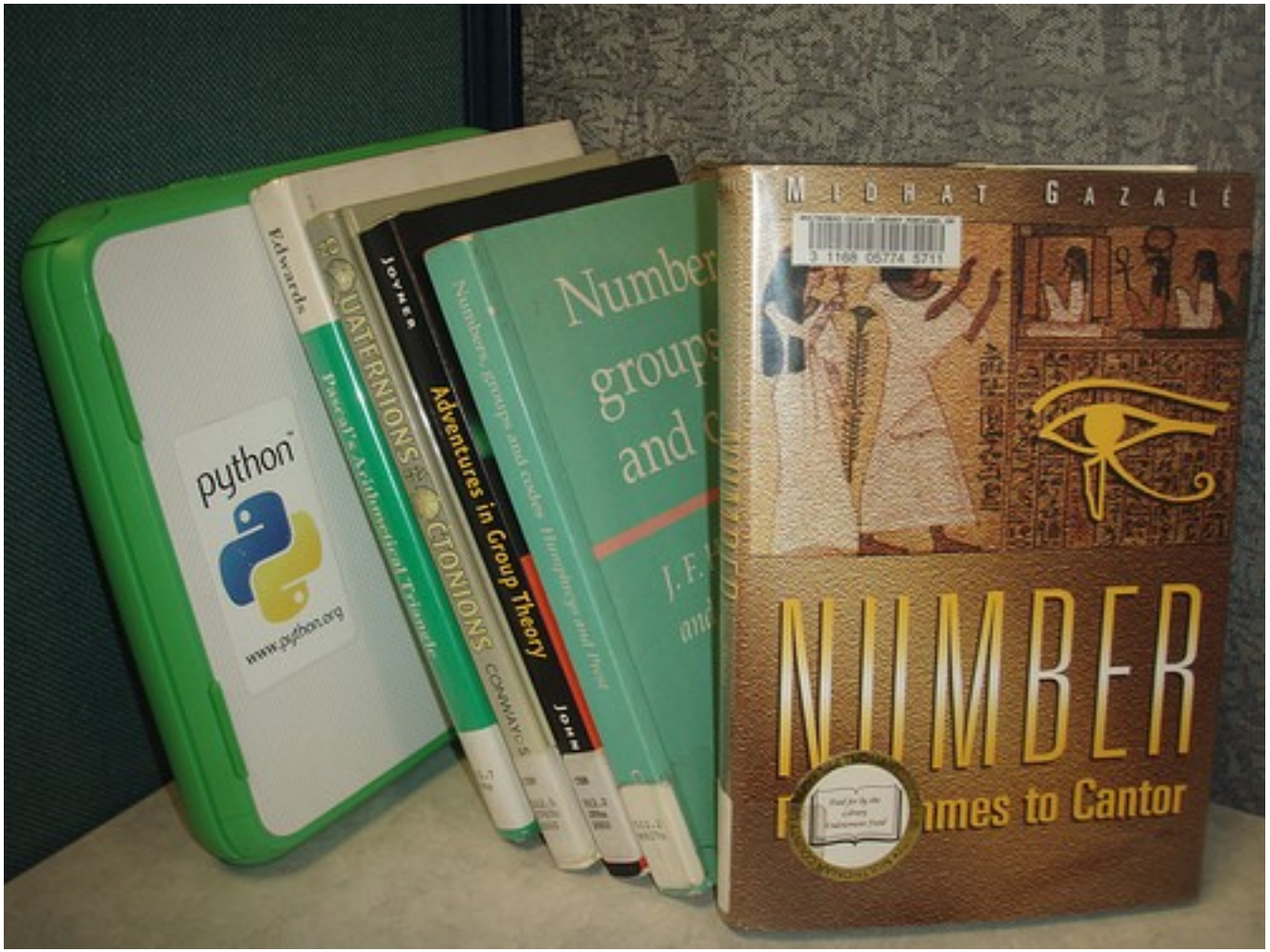

```
<pythonista>

class Patriotic_Snake:

    def __init__(self):
        self.proud = True
        self.vigilant = True
        self.brave = True

    def _watchout(self):
        return "I'm semi-private."

    def __backoff(self):
        return "Don't tread on me."

</pythonista>
```

(vary to suit, remember your freedoms)

Track A (lexical)

- Permutations (ascii / unicode)
- prime / composite
- gcd, lcd
- "modulo number"
- totative / totients
- finite groups, fields (algebra!)
- euler's theorem, RSA

Follow your curiosity



Study your heritage

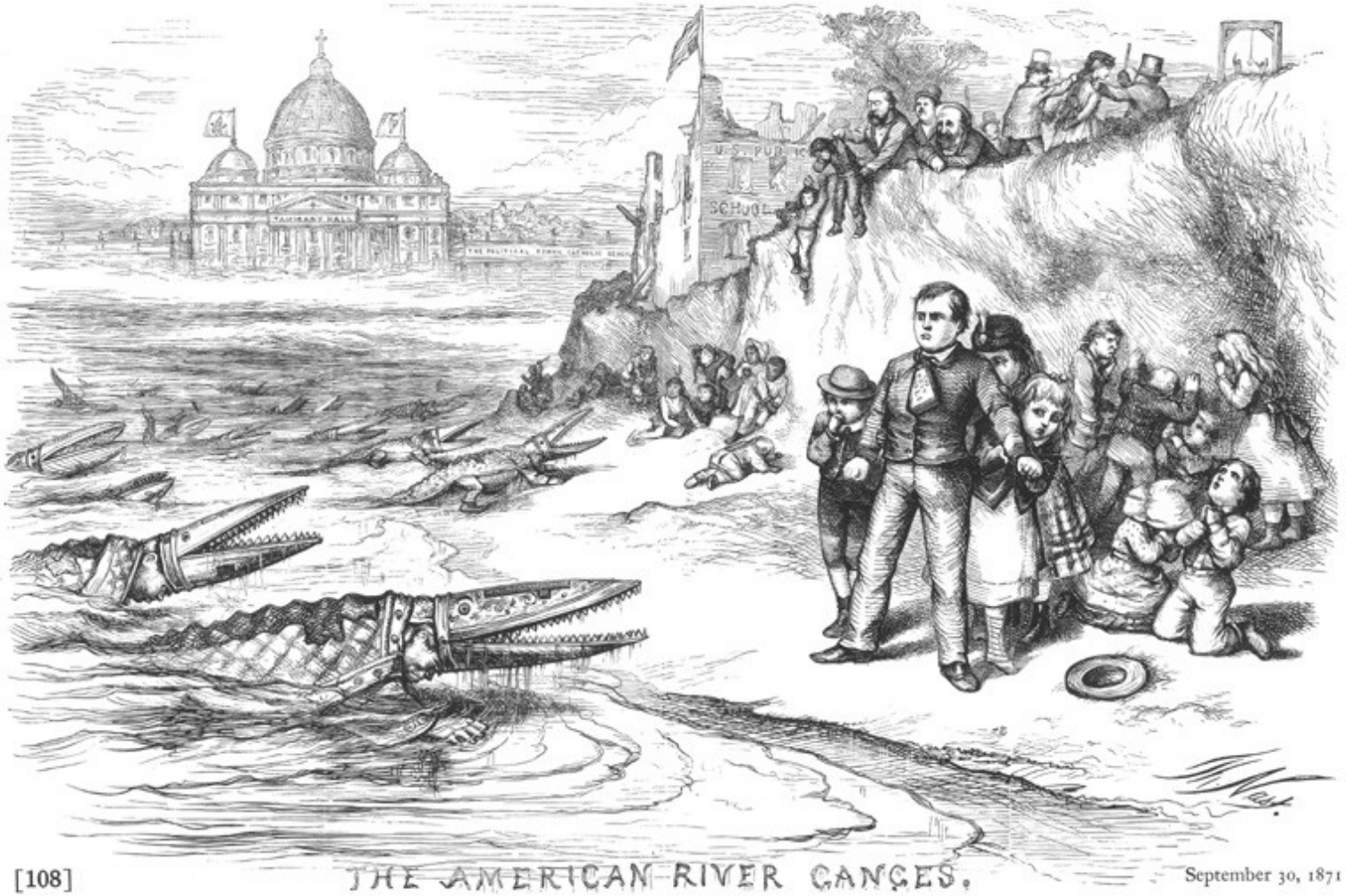
```
def gcd(a,b):
    """
    Guido's for the gcd:
    """
    while b:
        a, b = b, a % b
    return a
"""
Then two generics we've all seen many times,
generators for Pascal's Triangle and
Fibonacci Sequence respectively:
"""


def pascal():
    """
    Pascal's Triangle **
    """
    row = [1]
    while True:
        yield row
        row = [ a + b for a, b in zip( [0] + row, row + [0] ) ]

def fibonacci(a=0, b=1):
    while True:
        yield a
        a, b = a + b, a
```

Track B (graphical)

- animals, shapes, polymorphism
- figurate / polytopal numbers
- sphere packing
- vectors, trig, XYZ, lat/long etc.
- polyhedra as objects

A business story
(unfolding):
Portland as Toontown
(where some serious
cartooning goes on)

“biships”



The Priests and the Children.

Homer Davenport, from Springfield, Oregon was a student of Thomas Nast, inventor of political Elephant and Donkey icons

Full disclosure: I was a Catholic school teacher in the early 1980s and thought we did a great job

Oregon: home of the *real* Springfield

Animal zOO

- Biotum class, instances = biota
- Monkey, Dog, Snake classes
- Eating and pooping (deque?)
- Animal class (common ancestor)
- Polymorphism: "passing the buck"
- The ___rib___ cage
- "everything's a python in Python"

Your job: train *House M.D.* in Python (a busy grumpy guy)

File Edit Format Run Options Windows Help

```python
"""module:  simplelife.py"""

class Biotum:

    def __init__(self, name):
        self.name = name
        self.stomach = []

    def __call__(self,food):
        self.stomach.append(food)

    def __repr__(self):
        return 'Biotum named ' + self.name
```

Ln: 1 Col: 0

File Edit Shell Debug Options Windows Help

```
>>> import simplelife
>>> imp.reload(simplelife)
<module 'simplelife' from '/home/ki
rby/simplelife.py'>
>>> from simplelife import Biotum
>>> cell1 = Biotum('xV')
>>> cell2 = Biotum('xY')
>>> cell1.stomach
[]
>>> cell1('**')
>>> cell1.stomach
['**']
>>> cell1('&&&')
>>> cell1.stomach
['**', '&&&']
>>> cell1(cell2)
>>> cell1.stomach
['**', '&&&', Biotum named xY]
>>>
>>> |
```

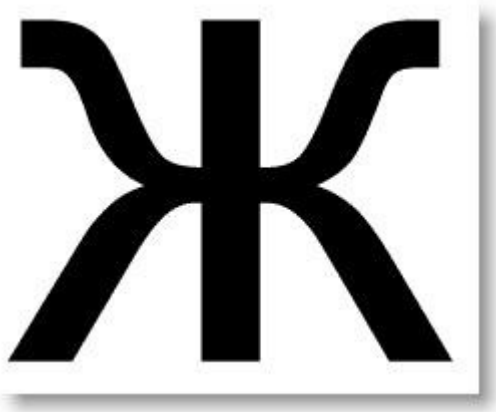
Ln: 55 Col: 4

Track C (synergetic)

- Rich data structures
- SQL (supermarket math)
- GIS / GPS (outdoor activities)
- mvps, time dimension
- user requests and events
- web framework (Django?)
- client / server design patterns

Lots of interconnections between tracks.

Lots of branches off both



Gnu Math:

What the Bleep!?

Excerpt from edu-sig

Figurate Numbers

Pascal's Triangle (triangular and tetrahedral numbers)

Fibonacci Numbers (converge to phi, pentagon math)

Vectors (VPython -- xyz, spherical coordinates etc.)

Prime Numbers (sieves)

Prime Numbers (trials by division)

Polyhedra (as Python objects: scale, rotate, translate)

Polyhedral Numbers (icosahedral, geodesic spheres)

Modulo Numbers (override __mul__, __add__)

Finite Groups (Python module)

Euclid's Algorithm (Guido's gcd)

Euclid's Extended Algorithm (needed for inverses)

Totient and Totative (gcd based)

Fermat's Little Theorem (assert...)

Euler's Theorem for Totients (assert...)

Mandelbrot Set (chaotic sequences)

Miller-Rabin (or Jython probablePrime)

RSA.encrypt(m, N)

RSA.decrypt(c, N, d=secretkey)

OLPC.XO.pippy includes Fibonacci's and Pascal's

The Rise of Unicode

Call me *Cosmopolitan*

"PASCAL'S TRIANGLE"

TRI

Tetra

1 = 2^0

1 1 = 2^1

1 2 1 = 2^2

1 3 3 1 = 2^3

1 4 6 4 1 = 2^4

1 5 10 10 5 1 = 2^5

$$2^N = \sum_{i=0}^{N-1} row[i]$$

BINOMIAL THEOREM

$(a+b)^N = \ldots$

"Bell Curve"

"Gaussian"

HOW MANY WAYS TO REACH THE BOTTOM ROW...

MAJOR BRIDGE

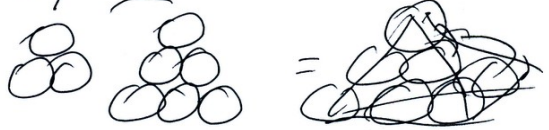
TRIANGULAR NUMBERS (TRI)

TETRAHEDRAL NUMBERS (TETRA)

4D SOLUTIONS

"a pioneer in open source"

1 3 6 ...



10 ...

Stack 'em

"Sphere packing" thread

geodesic spheres (domes)

biology

crystalography (gemology)

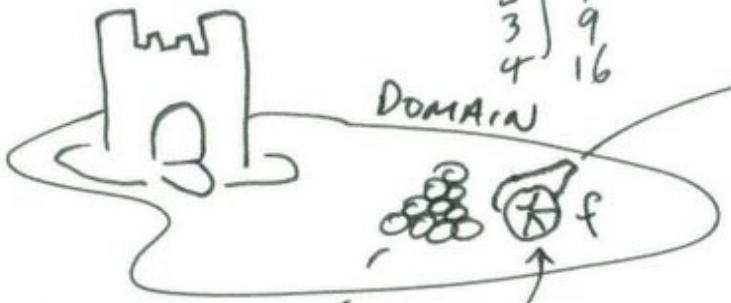
RENT RENT

Hi Tabit

x	y = f(x)
1	1
2	4
3	9
4	16

Sequences

Memory Palace (mnemonics)



WAR

hit (range)

Figurate & polyhedral Numbers

← easter egg in spokes

Tri numbs

Tetra hedron

CCP Pascal's

fruit

videogame (the war scenario is framed)

cash register

$

scanner

Kwh

HOME OFFICE (= STUDIO)

INVENTORY

$

Kwh

Checkout lanes

SUPERMARKET

Places show up here

MicroECON

geodesic dome (lots of seques!)

Relational Databases

client/server

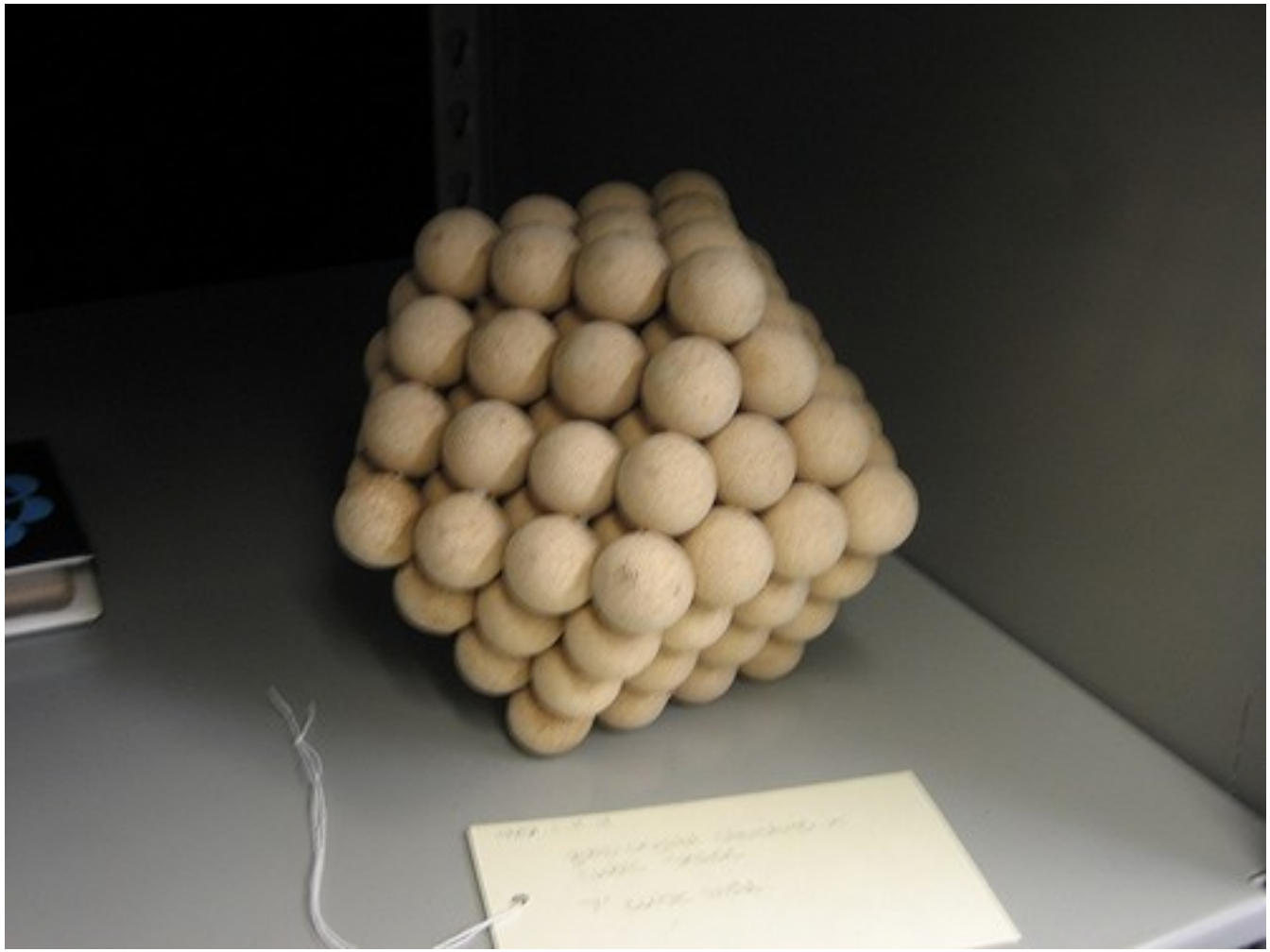
"energy cost accounting"

MEMORY

4D

Pythoneering in the Silicon Forest

1 + 12 + 42 + 92 spheres, CCP, Linus and Ava Helen Pauling special collection, Oregon Statue University, photo by K. Urner



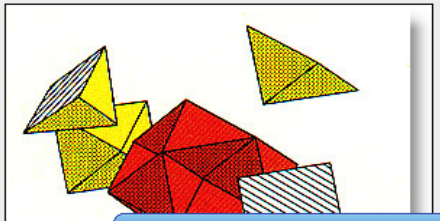
http://www.flickr.com/photos/17157315@N00/3049399298/in/set-72157612330643992/

Python Shell

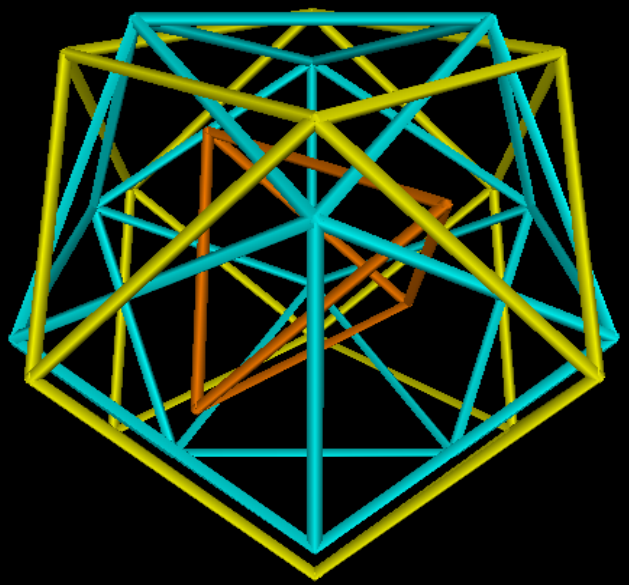
File Edit Shell Debug Options Windows Help

```
IDLE 1.2.1
>>> from rbf import Icosa, Tetra, Cubocta
>>> i = Icosa(); t = Tetra(); c = Cubocta()  # instantiate three polyhedra
>>> i.draw(); t.draw(); c.draw()  # draw them to a VPython canvas
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
```

cubocta20.png - Windows Picture an...



VPython



rbf.py - C:\Python25\Lib\site-packages\rbf.py

File Edit Format Run Options Windows Help

```
"""
VPYTHON VERSION

Originally for POV-Ray and VRML work. Re
directly with VPython.

By Kirby Urner, Oregon Curriculum Network

Ver 1.4:  March 15, 2005
Last modified:  March 14, 2005
Last modified:  April 15, 2001

Ver 0.2:  May 29, 2000

"""
```

Applications Places System Sun Mar 16, 1:45 PM

Python Shell

File Edit Shell Debug Options Windows Help

```
', 'ellipsoid', 'empty', 'equal', 'exp', 'fabs', 'faces', 'false', 'floor', 'flo
or_divide', 'fmod', 'frame', 'frexp', 'fromfunction', 'fromstring', 'greater', '
greater_equal', 'helix', 'hypot', 'i', 'identity', 'indices', 'init', 'innerprod
uct', 'invert', 'j', 'label', 'ldexp', 'left_shift', 'less', 'less_equal', 'load
', 'loads', 'log', 'log10', 'logical_and', 'logical_not', 'logical_or', 'logical
_xor', 'mag', 'mag2', 'math', 'matrixmultiply', 'maximum', 'minimum', 'modf', 'm
ul', 'multiarray', 'multiply', 'negative', 'nonzero', 'norm', 'not_equal', 'ones
', 'os', 'outerproduct', 'p1', 'p2', 'p3'                                  pickle_a
rray', 'pow', 'power', 'primitives', 'pro                                  , 'pyram
id', 'rad2deg', 'radians', 'rank', 'rate'                                  'remaind
er', 'repeat', 'reshape', 'resize', 'righ                                  ate', 'r
otx', 'roty', 'rotz', 's1', 's2', 's3', '                                  'scene',
 'searchsorted', 'shape', 'sign', 'sin',                                   rt', 'sp
here', 'sqrt', 'string', 'sub', 'subtract                                  ake', 't
an', 'tanh', 'time', 'trace', 'transpose'                                  odes', '
types', 'ui', 'util', 'vars', 'vdot', 've                                  ', 'wher
e', 'zeros']
>>> ico = Icosa()
>>> ico.draw()
>>>
```

Ln: 23 Col: 4

VPython

CPforKids_final_review.pdf

pyglet-1.1alpha1-py2.5.egg

Track C: SQL + Vpython (Polyhedra on tap!)

Applications Places System Tue Sep 9, 3:49 PM

Wing IDE: sqlpolys.py

File Edit Source Debug Tools Window Help

New Open... Save Save All Goto Definition Search Run Break Debug Stop Step Into Step Over Step Out

rbf.py sqlpolys.py

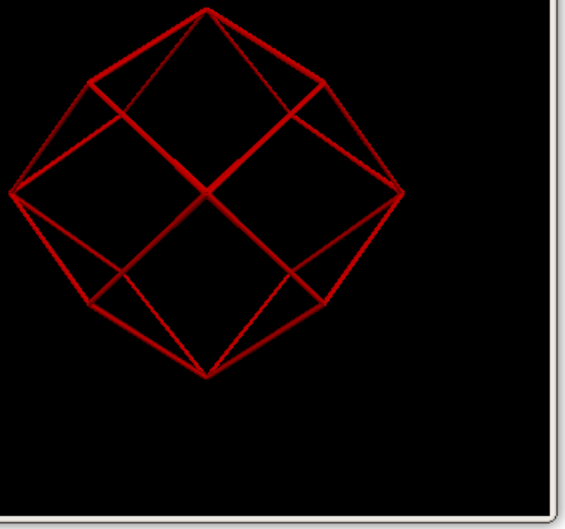
```python
from pysqlite2 import dbapi2 as sqlite
from visual import cylinder, color, vector

conn = sqlite.connect("newgeom")
curr = conn.cursor()


def drawpoly(shape):

    edges = getedges(shape)
    for a, b in edges:
        SELECT = """
        select x,y,z from vectors where label = ?
        """
        x,y,z = curr.execute(SELECT, (a,)).fetchone()
        v0 = vector(x,y,z)

        x,y,z = curr.execute(SELECT, (b,)).fetchone()
        v1 = vector(x,y,z)

        cyl=cylinder(pos=v0, axis=v1-v0,
                     radius= 0.01, color = color.red)

def getedges(shape):

    edgeset = set()  # need to weed out dupes

    SELECT = """
    select faces from polyhedra
```

Call Stack Exceptions Python Shell

Commands execute without debug. Use arrow keys for history. Options

```
Python 2.5.2 (r252:60911, Jul 31 2008, 17:28:52)
[GCC 4.2.3 (Ubuntu 4.2.3-2ubuntu7)]
Type "help", "copyright", "credits" or "license" for more informat
>>> import sqlpolys
>>> sqlpolys.drawpoly("cell")
>>>
```

VPython



Line 40 Col 44 -

Python Shell

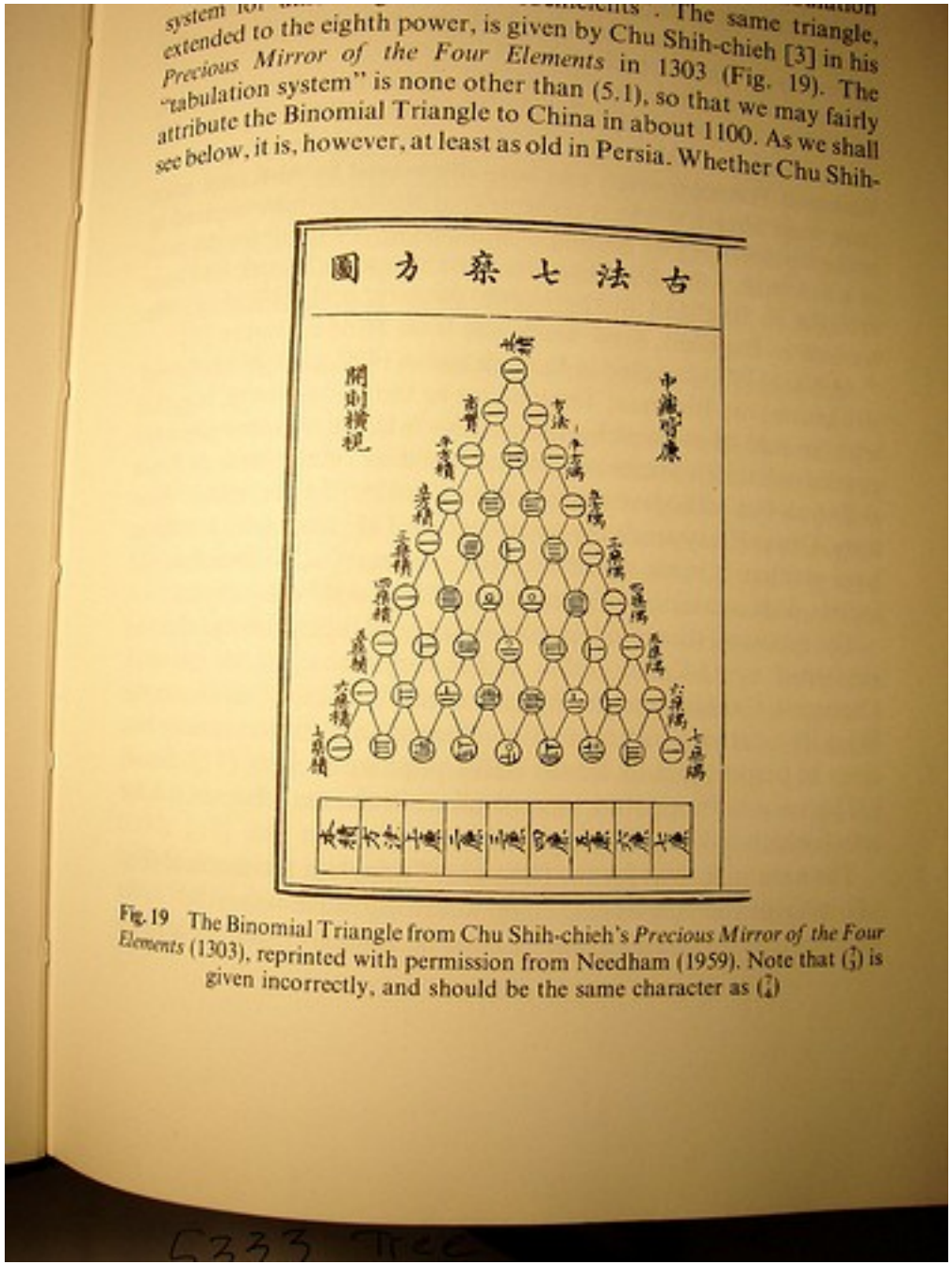
File Edit Shell Debug Options Windows Help

```
>>> def pascal():
      row = [1]
      while True:
        yield row
        row0 = [0] + row
        row1 = row + [0]
        row =  [i + j for i,j in zip(row0, row1)]


>>> pgen = pascal()
>>> pgen.next()
[1]
>>> pgen.next()
[1, 1]
>>> pgen.next()
[1, 2, 1]
>>> pgen.next()
[1, 3, 3, 1]
>>> pgen.next()
[1, 4, 6, 4, 1]
>>> pgen.next()
[1, 5, 10, 10, 5, 1]
```

Ln: 252 Col: 25

Python 2.5: generator for Pascal's Triangle

Remember me?



system for ... coefficients. The same triangle, extended to the eighth power, is given by Chu Shih-chieh [3] in his *Precious Mirror of the Four Elements* in 1303 (Fig. 19). The "tabulation system" is none other than (5.1), so that we may fairly attribute the Binomial Triangle to China in about 1100. As we shall see below, it is, however, at least as old in Persia. Whether Chu Shih-

古法七乘方圖

Fig. 19 The Binomial Triangle from Chu Shih-chieh's *Precious Mirror of the Four Elements* (1303), reprinted with permission from Needham (1959). Note that $\binom{7}{3}$ is given incorrectly, and should be the same character as $\binom{7}{4}$

Cite: http://mybizmo.blogspot.com/2009/01/wanderers-2009201.html

Python Shell

File Edit Shell Debug Options Windows Help

```
>>> def fibonaccis(a=0, b=1):
        while True:
                yield a
                a,b = a+b,a


>>> genf = fibonaccis()
>>> from __future__ import division
>>> genf.next()/genf.next()
0.0
>>> genf.next()/genf.next()
0.5
>>> genf.next()/genf.next()
0.5999999999999998
>>> genf.next()/genf.next()
0.6153846153846154 2
>>> genf.next()/genf.next()
0.6176470588235294 4
>>> genf.next()/genf.next()
0.6179775280898876
>>> genf.next()/genf.next()
0.6180257510729614 3
```

Ln: 252 Col: 25

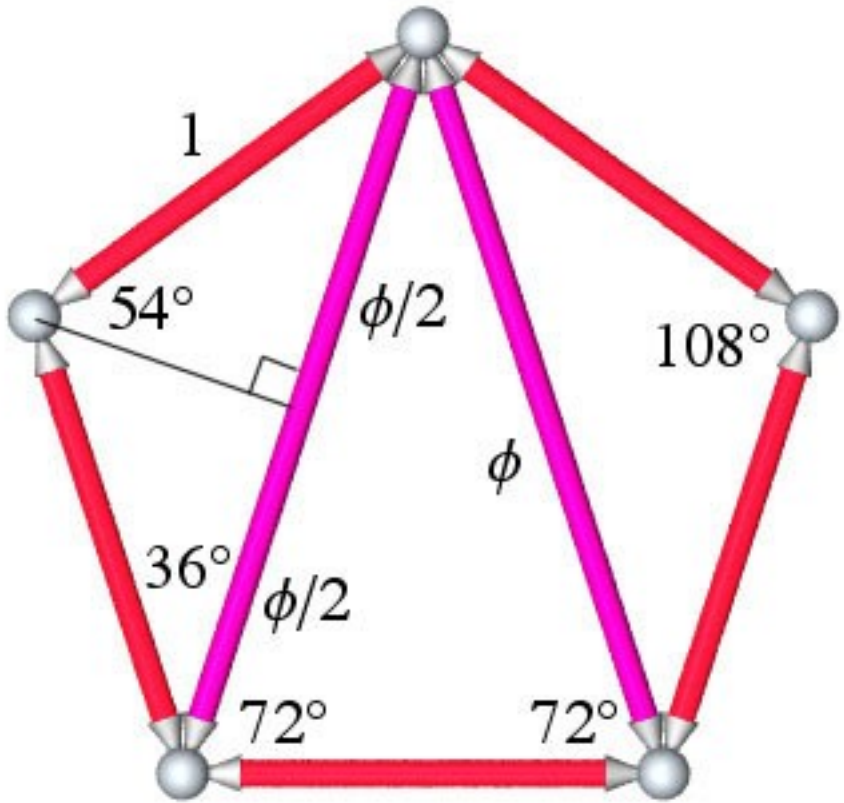
Python 2.5: generator for Golden Ratio

Pentagon Math:

a:b :: b:(a+b)
a = 1
b = phi

"nclb.polynomial"

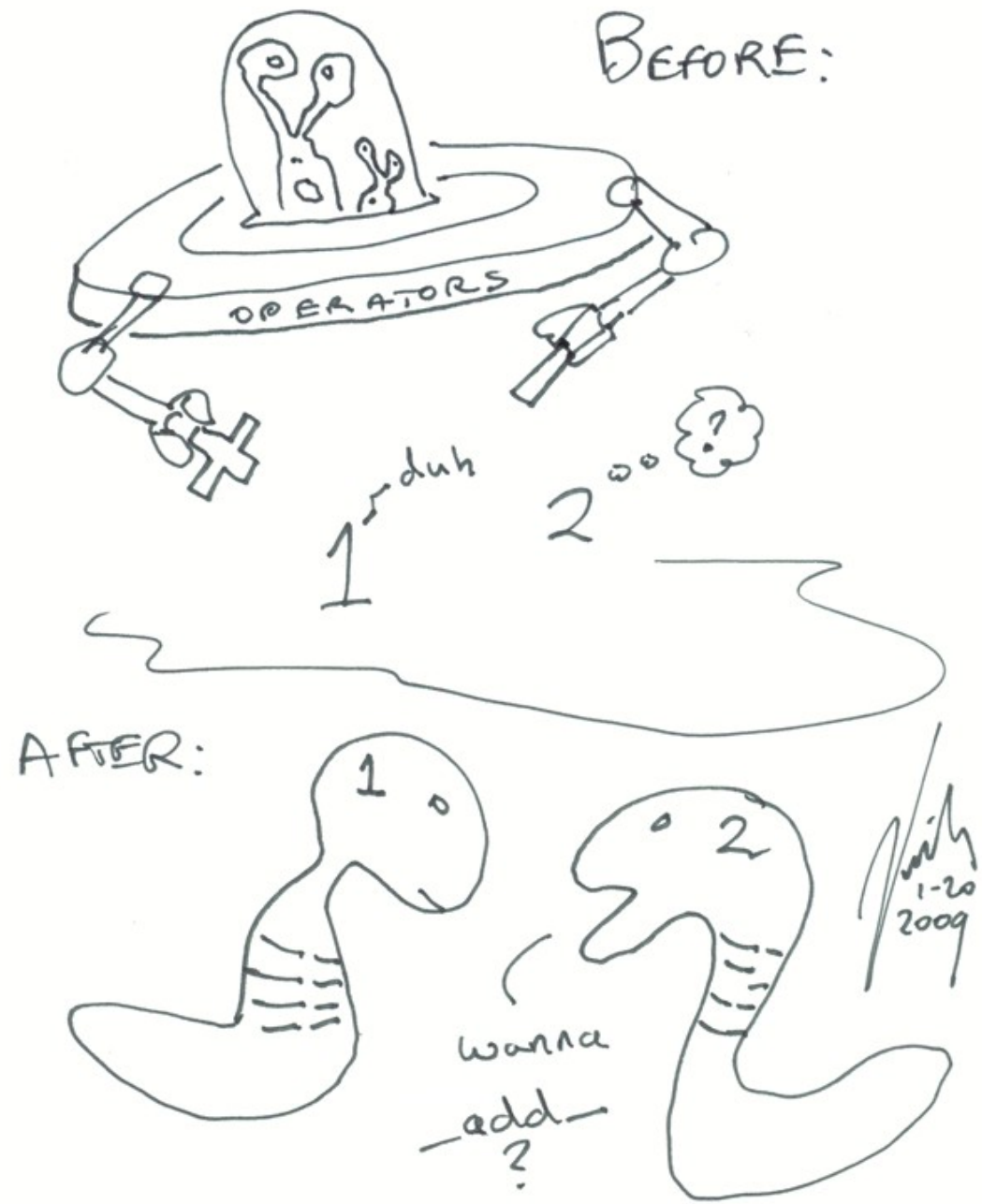
Paid work:
Python + POV-Ray
Design Science Toys;
StrangeAttractors project



$2\sin(54°) = 2\cos(36°) = \phi$

Note allusion to *Lost* here

Global Studies: Compiz on Ubuntu, GIS / GPS, Python, 4D Solutions



Preparing for "Modulo Numbers"

File Edit Format Run Options Windows Help

```python
from random import choice

def gcd(a,b):
    while b:
        a, b = b, a % b
    return a

def totatives(n):
    return [ x for x in range(1, n) if gcd(x, n) == 1]

class Modulo:
    m = 12
    def __init__(self, n):
        self.n = n % Modulo.m
    def __mul__(self, other):
        return Modulo(self * other)
    def __eq__(self, other):
        return self.n == other.n
    def __repr__(self):
        return str(self.n)

def closure_checker(pool):
    while True:
        a, b = choice(pool), choice(pool)
        result = a * b
        assert result in pool
        yield "%s * %s == %s" % (a, b, result)
```

Ln: 1 Col: 0

File Edit Shell Debug Options Windows Help

```
<module 'algebra' from '/home/kirby/algebra.py'>
>>> from algebra import *
>>> p = totatives(12)
>>> thegroup = [Modulo(x) for x in p]
>>> thegroup
[1, 5, 7, 11]
>>> g = closure_checker(thegroup)
>>> next(g)
'5 * 7 == 11'
>>> next(g)
'5 * 7 == 11'
>>> next(g)
'11 * 11 == 1'
>>> next(g)
'5 * 5 == 1'
>>> next(g)
'7 * 7 == 1'
>>> next(g)
'11 * 7 == 5'
```

Ln: 763 Col: 4

http://worldgame.blogspot.com/2009/01/modulo-numbers.html

File Edit Format Run Options Windows Help

```
from random import choice

def gcd(a,b):
    while b:
        a, b = b, a % b
    return a

def totatives(n):
    return [ x for x in range(1, n) if gcd(x, n) == 1]

class Modulo:
    m = 12
    def __init__(self, n):
        self.n = n % Modulo.m
    def __mul__(self, other):
        return Modulo(self * other)
    def __pow__(self, n):
        return Modulo(pow(self.n, n, Modulo.m))
    def __eq__(self, other):
        return self.n == other.n
    def __repr__(self):
        return str(self.n)
```

New, Improved: added __pow__

Ln: 1 Col: 0

Checking Stuff...

File Edit Format Run Options Windows Help

```
def closure_checker(pool):
    while True:
        a, b = choice(pool), choice(pool)
        result = a * b
        assert result in pool
        yield "%s * %s == %s" % (a, b, result)


def euler_checker(N, base=2):
    """
    a check (not a proof) of Euler's Theorem
    """
    if gcd(base, N) == 1: # coprime?
        totient = len(totatives(N))
        try:
            assert pow(base, totient, N) == 1
            print("True!:  pow(%s, %s, %s) == 1" % (base, totient, N))
        except:
            print("False!:  pow(%s, %s, %s) != 1" % (base, totient, N))
    else:  # (base, N) not coprime
        print("Pick a different base please")
```

Ln: 44 Col: 45

File Edit Shell Debug Options Windows Help

```
====================
>>> from algebra import *
>>> euler_checker(217, base=5)
True!:  pow(5, 180, 217) == 1
>>> euler_checker(17)
True!:  pow(2, 16, 17) == 1
>>> euler_checker(13452199029039023920, base=2)
Pick a different base please
>>> euler_checker(13452199029039023921, base=2)
Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    euler_checker(13452199029039023921, base=2)
  File "/home/kirby/algebra.py", line 37, in euler_checker
    totient = len(totatives(N))
  File "/home/kirby/algebra.py", line 9, in totatives
    return [ x for x in range(1, n) if gcd(x, n) == 1]
  File "/home/kirby/algebra.py", line 9, in <listcomp>
    return [ x for x in range(1, n) if gcd(x, n) == 1]
  File "/home/kirby/algebra.py", line 5, in gcd
    a, b = b, a % b
KeyboardInterrupt
>>>
```

Ln: 155 Col: 0

Totatives Algorithm insufficiently powered to take us into “really big numbers” territory

A Geometry of Lumps
(Karl Menger: dimension theorist)

points are pointy

lines are lengthy

planes are flat

"everything is an object"

Yakking about Namespaces

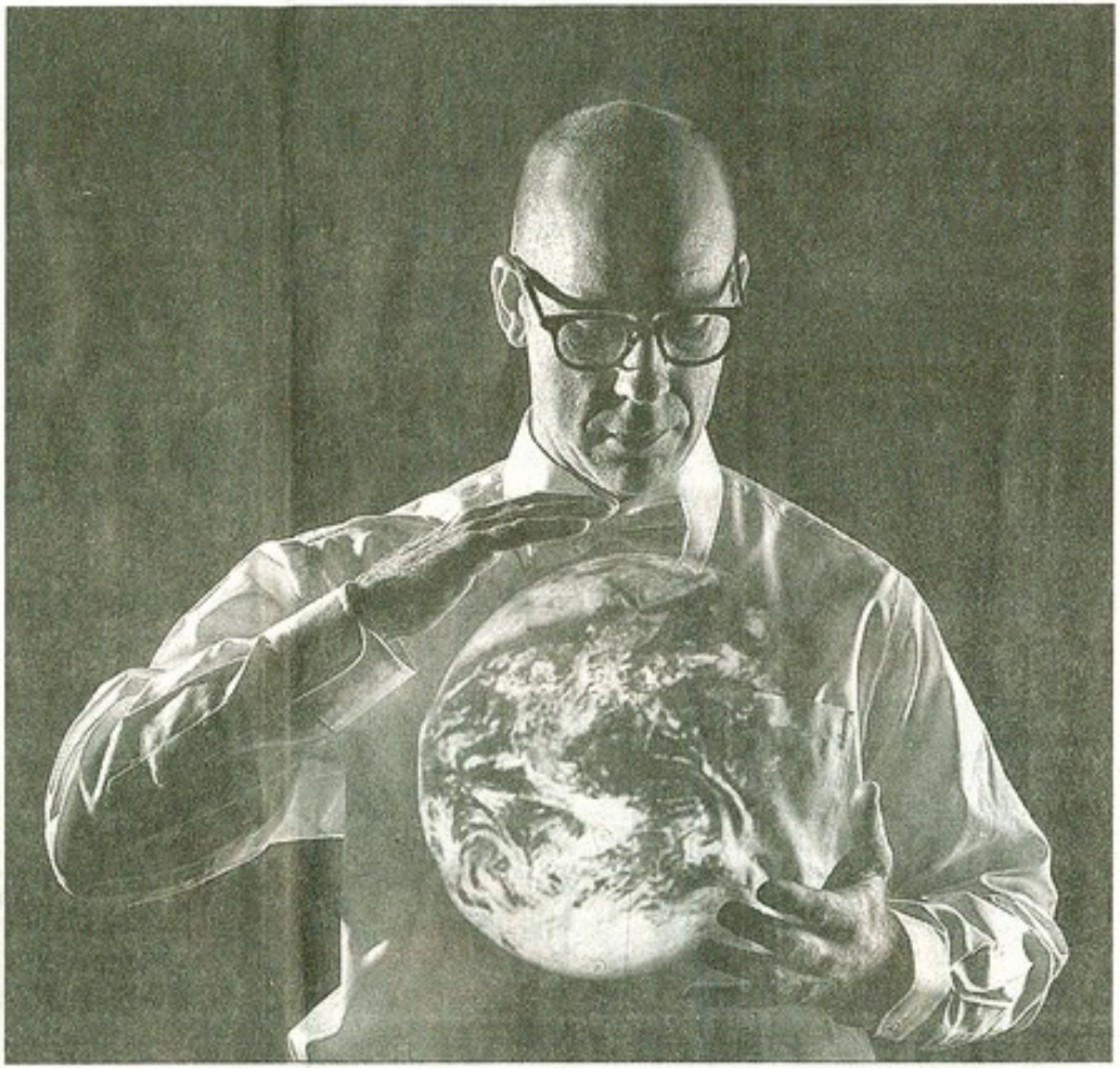
Einstein . 4D : time as the fourth dimension, imaginary time (Stephen Hawking)

Coxeter . 4D : distinct from einstein.4d per page 119 of *Regular Polytopes*

Fuller . 4D : everything is a lump, like a tetrahedron (*Synergetics* dedicated to Coxeter)

See: Linda Dalrymple Henderson, *The Fourth Dimension and Non-Euclidean Geometry in Modern Art*, Princeton University Press, 1983

From *The Oregonian – October 20, 2008*



OWEN CAREY

Doug Tompos is the title character in "R. Buckminster Fuller: The History (and Mystery) of the Universe," a one-man play at Portland Center Stage that captures the thinker's restless curiosity and playfulness.

Storytelling (lore) in PDX, 2008

Stickworks (recycle, reuse) :

- wrap Vpython's vector with some extra stuff
- keep Vector tails at the Origin
- define Edges as a pair of Vector tips
- define Faces as sequences of Vectors
- define Polyhedra as lists of Faces
- rbf volumes OK (encouraged even)

Stickworks is a Python module, also a namespace, also a “meme pool”