



South Texas Mathematics Consortium
Eighth Annual Meeting, February 12, 2000
Univ. of Texas at San Antonio, Downtown



PASCAL IS DEAD. LONG LIVE PYTHON!

Dr. Dmitry Gokhman
Division of Mathematics and Statistics
The University of Texas at San Antonio
San Antonio, Texas 78249-0664 U.S.A.
<http://www.math.utsa.edu/~gokhman>
gokhman@math.utsa.edu

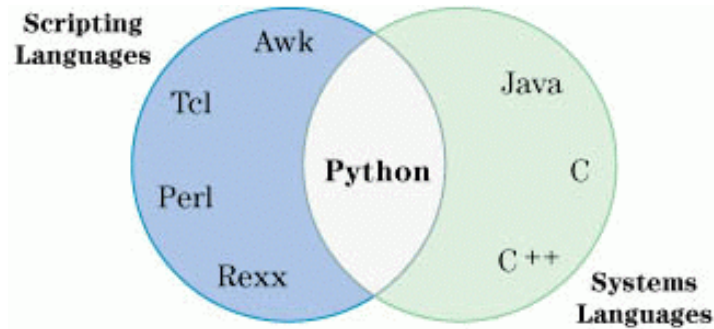
History



- **Precursor:** ABC v. 2 —
a scripting language by Guido van Rossum for teaching computer concepts (1980's)
Elegance, ease of learning.
- **Python:** Guido van Rossum (CWI, Amsterdam, winter break 1989/90)
Openness to external code, performance.
- **Numpy:** Konrad Hinsen et al.
Multiarrays, ufuncs, numerical linear algebra.
- **Users:**
 - Lawrence Livermore National Laboratory (numpy)
 - Johnson Space Center (Integrated Planning System)
 - American Association of University Presses (online catalog)
 - Swedish Meteorological and Hydrological Inst. (satellite data → TV)
 - Search engines (InfoSeek, Four11, Musi-Cal)
 - Bank United (online loan calculator)
 - Me!

Features

Python combines features from scripted (Perl, Tcl, Scheme) and systems (C, C++, Java, FORTRAN) languages.



- **Free:** Download for most platforms (Linux, Solaris, Win, OS/2, Mac, Amiga, VMS), docs (ref, tutorial) also free, fast and easy to install.
- **Scripted:** Easy to learn, flexible (e.g. eval, implicit typing, freedom from declaration and allocation chores and errors!), quick prototyping, easy debugging, slower execution.
- **Classes:** Pure object orientation, easier than C++ or Java.
- **Readable:** Conventional looking, scoping rules, exception handling, modules, ideal for teams!
- **Math:** Built-in large integers, eval.
- **Extensible:** Linking to precompiled code in systems languages (C, FORTRAN, Java) increases performance. Interfaces to applications (Gnuplot) and window system toolkits (Tk, Qt, MS Foundation Classes).
- **Portable:** The Python interpreter is the same for all platforms. Evolution extremely conservative.

Newton's method

```
from math import *
def Newton(f, df, xini=0., limit=100, tol=10.**(-14)):
    x=xini
    i=0
    while 1:
        print i, x
        i=i+1
        newx=x-f(x)/df(x)
        if abs(newx-x) <= tol:
            break
        x=newx
        if i > limit:
            print 'FATAL: loop count exceeds'+limit
            break
    return newx
if __name__ == '__main__':
    def f(x):
        return cos(x)-x
    def df(x):
        return -sin(x)-1
    x=Newton(f, df, xini=3.)
    print 'The answer is ', x
```

Here is the output

0 3.0

1 -0.496558178297

2 2.13100384448

3 0.689662720778

4 0.739652997531

5 0.739085204376

6 0.739085133215

The answer is 0.739085133215

Greatest Common Divisor

```
def gcd(a,b):
    if b >= a:
        a, b = b, a
    while (b != 0):
        rem = a % b
        a, b = b, rem
    return a
```

Vector norms

```
from math import *
from umath import *
from Numeric import *
def lpnorm(vector,p=2):
    if p == 'inf':
        return max(abs(vector))
    else:
        return add.reduce(vector**p)**(1./p)
v=array([-4,3])
print lpnorm(v)
print lpnorm(v,'inf')
```

Composing functions

```
from math import *
def comp(f, g):
    return lambda x, f=f, g=g: f(g(x))
sinsq = comp(lambda x: x**2, sin)
print sinsq(pi/6)
```

Linear Algebra

```
from Numeric import *
import LinearAlgebra; LA=LinearAlgebra
aug=array([[2.,1.,3.],[-1.,2.,-2]])
print 'aug=',aug
print 'shape(aug)=',shape(aug)
A=aug[:,0:2]
print 'A=',A
b=aug[:,2]
print 'b=',b
aug[1,:]=aug[1,:]-aug[0,:]*aug[1,0]/aug[0,0]
print aug
```

Output:

```
aug= [[ 2.  1.  3.]
      [-1.  2. -2.]]
shape(aug)= (2, 3)
A= [[ 2.  1.]
     [-1.  2.]]
b= [ 3. -2.]
[[ 2.  1.  3. ]
 [ 0.  2.5 -0.5]]
```

Linear Algebra continued ...

```
x=LA.solve_linear_equations(A,b)
print 'x=',x
Ainv=LA.inverse(A)
print 'Ainv=',Ainv
print 'Ainv b=',matrixmultiply(Ainv,b)
c=matrixmultiply(A,transpose(x))
print 'Check: A x=',c
print 'eigen(A)=',LA.eigenvalues(A)
```

Output:

```
x= [ 1.6 -0.2]
Ainv= [[ 0.4 -0.2]
       [ 0.2  0.4]]
Ainv b= [ 1.6 -0.2]
Check: A x= [ 3. -2.]
eigen(A)= (array([ 2.+1.j,  2.-1.j]),
array([[ 0.70710678+0.j      ,  0.      +0.70710678j],
       [ 0.70710678+0.j      ,  0.      -0.70710678j]]))
```

Files ...

```
import os
import tempfile
import string
from Numeric import *
def plotopen():
    file=tempfile.mktemp()
    pipe=open(file,'w')
    return [file,pipe]
def plot(data,buf):
    try:
        buf.write(string.replace(string.replace(
array2string(data),' ',''),'],'',''))
        buf.write('\n\n')
    finally:
        buf.close
def graphdisp(file,pipe):
    pipe.close()
    os.system('/usr/bin/env xgraph -bg white -M '+file+' &')
```

... and pipes

```
def plotopen(options=' '):
    return os.popen('xgraph -bg white '+options,'w')
```

Reading an array

```
from Numeric import *
import string
def ArrayFromFile(file):
    f=open(file)
    try:
        t=[]
        for line in f.readlines():
            t.append(string.atof(line))
    finally:
        f.close()
    return array(t)
```

Classes

Here is a part of my spline module showing the class definition, the initialization method, and the plotdata method.

```
import Xgraph
class Spline:
    def __init__(self,deg=0,data=[],boundary=[],domain=array([0.,1.]),
dt=0.,nodes=0):
        self.deg=deg
        self.data=data
...
    def plotdata(self,buf):
        data=concatenate((self.data[0],self.data[1]))
        data.shape=(2,len(self.data[0]))
        Xgraph.plot(buf=buf,data=transpose(data))
...
```

Here is how it is used in another module:

```
import Spline; S=Spline
import Xgraph
t=ArrayFromFile('grid.dat')
srhs=S.Spline(deg=3,data=[t])
file,xpipe=Xgraph.plotopen()
srhs.plotdata(buf=xpipe)
graphdisp()
```

Subclassing (inheritance)

```
class FancySpline(Spline):
...

```

Libraries

Python comes with nifty class libraries for all sorts of things.

String parsing

```
import string
words=string.split(line,',')
second_word=words[1]
```

FTP retrieval

```
from ftplib import FTP
ftp=FTP('ftp.python.org')
ftp.login()
ftp.cwd('pub/python/doc')
ftp.retrlines('LIST')
file=open('python_faq.html','w')
ftp.retrbinary('RETR FAQ.html', file.write, 1024)
ftp.quit()
```

HTTP Server

```
import SimpleHTTPServer
SimpleHTTPServer.test()
```

Resources

- <http://www.python.org/>
 - <http://starship.python.net/crew/hinsen/> (K. Hinsen's Python for Science)
-

References

- C. Laird, K. Soraiz, *Getting started with Python*
<http://sunsite.icm.edu.pl/sunworldonline/swol-02-1998/swol-02-python.html>
- Magnus Lie Hetland, *And now for something completely different... Instant Python*
<http://www.idt.unit.no/~mlh/python/instant.html>
- A. Kuchling, *Algorithms: A Bit of Math*
<http://www.pythonjournal.com/volume1/issue1/art-algorithms/>
- G. van Rossum, *Scripting the Web with Python*
<http://www.w3j.com/6/s3.vanrossum.html>

Python is not one of the research languages which seem to get promoted solely for pedagogical reasons. ... Python seems to encourage object oriented programming by clearing the paths, rather than erecting parapets.

— Jeff Bauer (Linux Journal)

Easy to learn, easy to use: Python is worth checking out.

— Cameron Laird and Kathryn Soraiz (Sun World)

From the Python Frequently Asked Questions list

1.16. Do I have to like "Monty Python's Flying Circus"?

No, but it helps. ...