

Python »

← (Alt+X) |

Python

(HYRY Studio)

-
-
-
-
-
-
-
-

reStructuredText Sphinx hyry.dip.jp IE7.0

FireFox Google Chome

- <http://hyry.dip.jp/pydoc> ()
- <http://pyscin.appspot.com/html/index.html> ()

Html CHM PDF

Html CHM PDF

Google Zoom Quiet.org() PDF

HYRY Studio

- HYRY Studio <http://hyry.dip.jp>
- <http://hyry.dip.jp/blogt.py>

Python

NumPy, SciPy, Matplotlib, Enthought librarys
Python2D3D
MatlabPythonMatlabMatlabtoolboxPython

Python2D3DC

PythonPython

Python <http://wiki.woodpecker.org.cn/moin/PyBooks>

Windows XPPython(x,y)PythonPython(x,y)

- -
 -
- NumPy-
 - ndarray
 - ufunc
 -
 -
- SciPy-
 -
 -
 -
 - B-Spline
 -
 -
 -
 - WeaveC
- SymPy-
 -
 -
- matplotlib-
 -
 -
 -
 - Artist
- Traits-Python
 -
 - Traits
 - Trait
 - Property

- Trait
- TraitsUI-
 -
 -
 -
- Chaco-
 -
 -
- TVTK-
 - TVTK
 - TVTK
- Mayavi-
 - mlab
 - Mayavi
 - Mayavi
- Visual-3D
 -
 -
 -
- OpenCV-
 -

- Traits
 - traits
 - traits.ui
- Visual
 -

- - Wave
 - pyAudio
 - pyMediaMp3
- - FIRIIR
 - FIR
 - IIR
 -
 -
- FFT
 - FFT
 -
 - FFT
- -
 -
 - Hilbert
- CtypesNumPy
 - ctypes
 - ctypesDLL
 - numpyctypes
- NLMS
 -
 - NLMS
 - NumPy
 - DLL
 - ctypespython
- -

-
- - Mandelbrot
 - (IFS)
 - L-System



-
-
-
- ReST
-
-

Python »

← (Alt+X) |

Python »

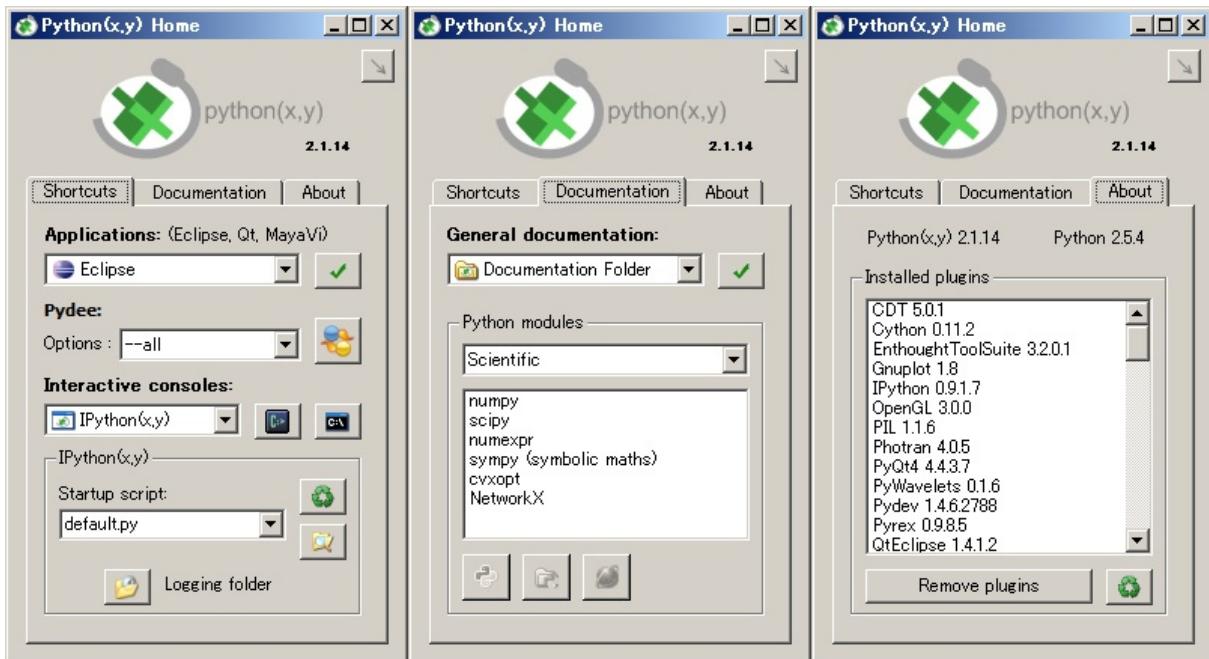
← (Alt+X) | |

MatlabPython

Python

Python(x,y)

<http://www.pythonxy.com> Python(x,y)400M

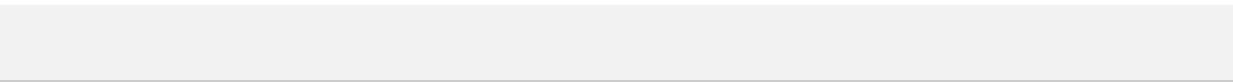


Python(x,y)

- Shortcuts
- Documentation
- About

Enthought Python Distribution (EPD)

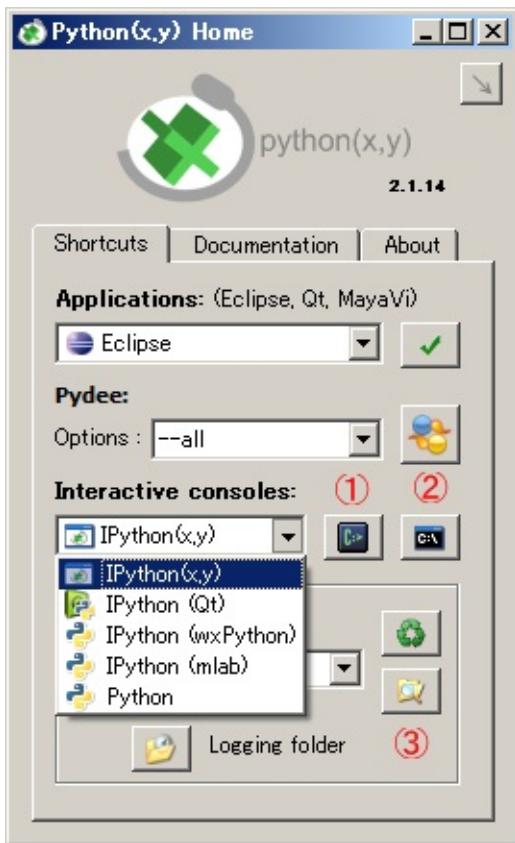
<http://www.enthought.com/products/getepd.php> EPD Python
250M



iPython

ipython python shell python shell bash
shell

Python(x,y) Python(x,y) iPython



Python(x,y) Home iPython

iPython
IPython(mlab) iPython Python Python Shell

① ② iPython iPython(x,y)

IPython(x,y)	-pylab -p xy
IPython(Qt)	-q4thread
IPython(wxPython)	-wthread
IPython(mlab)	-wthread

①ConsoleShellShell

python(x,y)IPython(x,y)iPythoniPythondefault.py:

```
import numpy
import scipy
from numpy import *
```

numpy, scipy

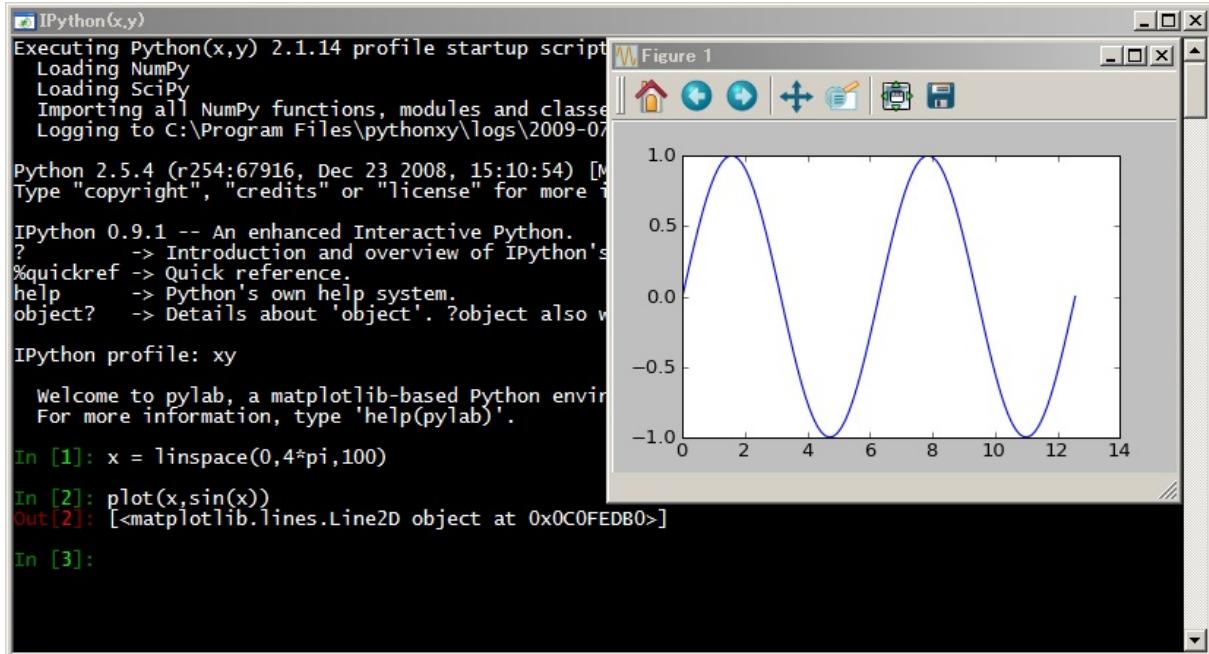
③numpy.py:

```
import numpy as np
import scipy as sp
import pylab as pl
```

IPython(x,y)numpy.py

pylabTraitsUIshell-wthread(-pylab)

pylabsin



IPython

iPython

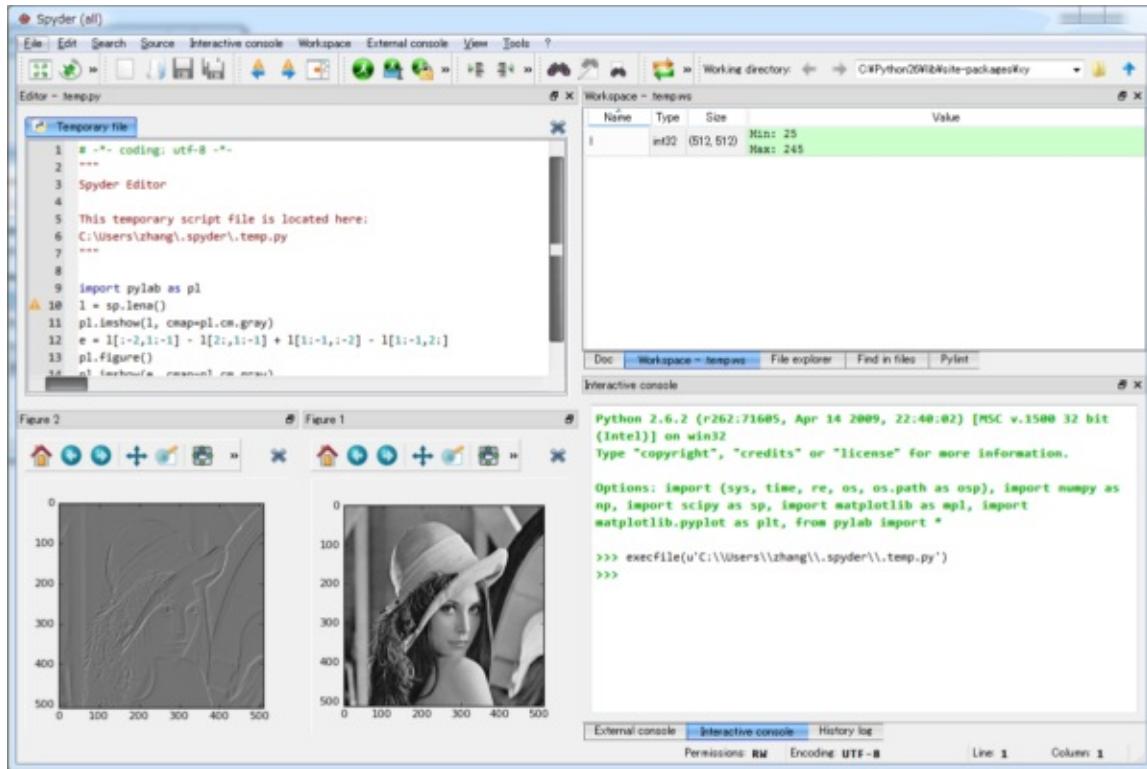
- tabiPython
- ??????PythonPython
- cmdls-cd-
- Pythonrun *.pyIPythonpy-ilPythonIPython
- IPythonpastepaste foofoofoolPythonSList
- !!test.pytest.pyruntest.py

spyder

spyderPython(x,y)PythonPython IDEMATLABworkspace

spyder <http://code.google.com/p/spyderlib>

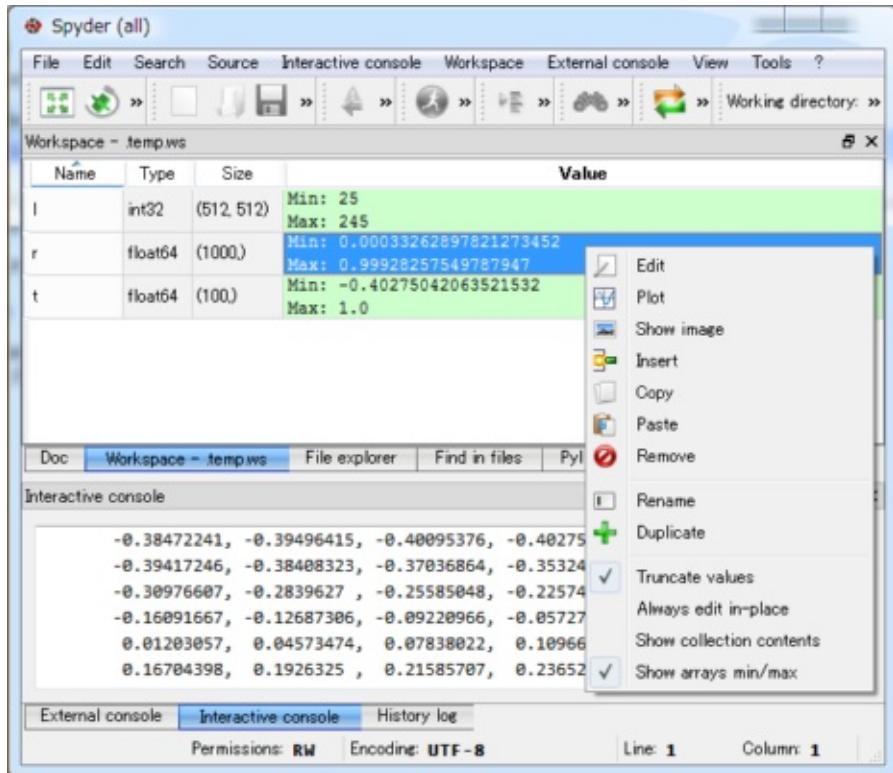
spyder



Spyder

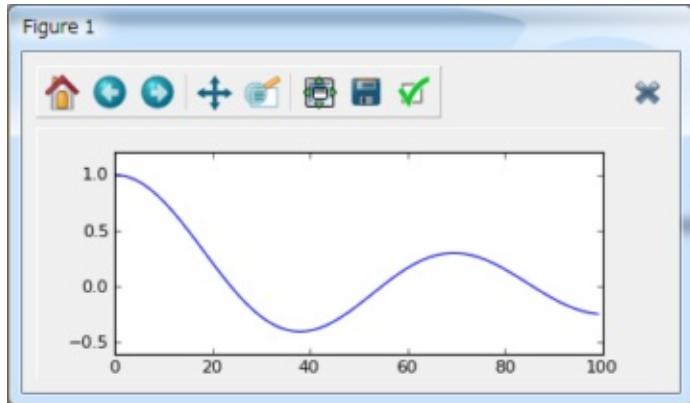
Workspace

Warning: Workspace



Workspace

Plot



Workspace

Edit

Array editor - a

	0	1	2	3	4
0	0.000	1.000	2.000	3.000	4.000
1	10.000	11.000	12.000	13.000	14.000
2	20.000	21.000	22.000	23.000	24.000
3	30.000	31.000	32.000	33.000	34.000
4	40.000	41.000	42.000	43.000	44.000
5	50.000	51.000	52.000	53.000	54.000
6	60.000	61.000	62.000	63.000	64.000
7	70.000	71.000	72.000	73.000	74.000
8	80.000	81.000	82.000	83.000	84.000
9	90.000	91.000	92.000	93.000	94.000

Format Resize Background color

OK Cancel



Python

NumPy Python **SciPy** NumPy Python Matlab

NumPy SciPy <http://www.scipy.org>

NumPy Python Python C

SciPy Fortran

- LAPACK
- FFTPACK
- ODEPACK
- MINPACK

SymPy Python 1.0

SymPy <http://code.google.com/p/sympy>

Traits

Traits <http://code.enthought.com/projects/traits>

Traits Traits Traits UI Traits Python traits

Traits UI Traits MVC Traits Traits UI wxPython pyQt

Chaco **matplotlib2D** **Chaco Traits** **matplotlib3D**

Chaco <http://code.enthought.com/projects/chaco>

matplotlib <http://matplotlib.sourceforge.net>

Tvtk **VTK** **KVTK** **Traits** **Python** **VTK** **VTK** **Mayavi2** **Tvtk** **3D**

Mayavi2 <http://code.enthought.com/projects/mayavi>

VTK(Visualization Toolkit)

VTK Visualization Toolkit
VTK1 Petabyte
Los Alamos 10242005 ZSU23-42.5 --

Visual3D

Visual <http://vpython.org>

OpenCV **BSD** **OpenCV** **OpenCV** **Python API**

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

NumPy-

Python(list)[1,2,3]3CPU

PythonarrayarrayC

NumPyNumPyndarrayN-dimensional array object
ufuncuniversal function objectndarray()ufunc

ndarray

NumPy:

```
import numpy as np
```

arrayPython(c):

```
>>> a = np.array([1, 2, 3, 4])
>>> b = np.array((5, 6, 7, 8))
>>> c = np.array([[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 10]])
>>> b
array([5, 6, 7, 8])
>>> c
array([[1, 2, 3, 4],
       [4, 5, 6, 7],
       [7, 8, 9, 10]])
>>> c.dtype
dtype('int32')
```

shape

```
>>> a.shape
(4, )
>>> c.shape
(3, 4)
```

ashapecshape0314shapecshape(4,3)
(3,4)(4,3)

```
>>> c.shape = 4, 3
>>> c
array([[ 1,  2,  3],
       [ 4,  4,  5],
       [ 6,  7,  7],
       [ 8,  9, 10]])
```

-1cshape(2,6)

```
>>> c.shape = 2, -1
>>> c
array([[ 1,  2,  3,  4,  4,  5],
       [ 6,  7,  7,  8,  9, 10]])
```

reshapeshape

```
>>> d = a.reshape((2,2))
>>> d
array([[1, 2],
       [3, 4]])
>>> a
array([1, 2, 3, 4])
```

ad

```
>>> a[1] = 100 # a100
>>> d # d2
array([[ 1, 100],
       [ 3,   4]])
```

dtype32bitdtype:

```
>>> np.array([[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 10]], dtype=np
array([[ 1.,  2.,  3.,  4.],
       [ 4.,  5.,  6.,  7.],
       [ 7.,  8.,  9., 10.]])
>>> np.array([[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 10]], dtype=np
array([[ 1.+0.j,  2.+0.j,  3.+0.j,  4.+0.j],
       [ 4.+0.j,  5.+0.j,  6.+0.j,  7.+0.j],
       [ 7.+0.j,  8.+0.j,  9.+0.j, 10.+0.j]])
```

PythonarrayNumPy

- arange python range:

```
>>> np.arange(0,1,0.1)
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,
```

- linspace endpoint:

```
>>> np.linspace(0, 1, 12)
array([ 0.          ,  0.09090909,  0.18181818,  0.27272727,  0
       0.45454545,  0.54545455,  0.63636364,  0.72727273,  0
       0.90909091,  1.         ])
```

- logspace linspace 1(10^0)100(10^2)20:

```
>>> np.logspace(0, 2, 20)
array([ 1.          ,  1.27427499,  1.62377674,  2.0691
       2.6366509 ,  3.35981829,  4.2813324 ,  5.4555
       6.95192796,  8.8586679 ,  11.28837892,  14.3844
      18.32980711,  23.35721469,  29.76351442,  37.9269
      48.32930239,  61.58482111,  78.47599704,  100.])
```

frombuffer, fromstring, fromfile fromstring:

```
>>> s = "abcdefg"
```

Python s8bit ASCII:

```
>>> np.fromstring(s, dtype=np.int8)
array([ 97,  98,  99, 100, 101, 102, 103, 104], dtype=int8)
```

s16bit 98971698*256+97 = 25185 little endian()

```
>>> np.fromstring(s, dtype=np.int16)
array([25185, 25699, 26213, 26727], dtype=int16)
>>> 98*256+97
25185
```

64:

```
>>> np.fromstring(s, dtype=np.float)
array([ 8.54088322e+194])
```

Cdoublefromstringfloat64

Python

```
>>> def func(i):
...     return i%4+1
...
>>> np.fromfunction(func, (10,))
array([ 1.,  2.,  3.,  4.,  1.,  2.,  3.,  4.,  1.,  2.])
```

fromfunction(shape)(10,)10

aa[i, j]func2(i, j)

```
>>> def func2(i, j):
...     return (i+1) * (j+1)
...
>>> a = np.fromfunction(func2, (9,9))
>>> a
array([[ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.],
       [ 2.,  4.,  6.,  8., 10., 12., 14., 16., 18.],
       [ 3.,  6.,  9., 12., 15., 18., 21., 24., 27.],
       [ 4.,  8., 12., 16., 20., 24., 28., 32., 36.],
       [ 5., 10., 15., 20., 25., 30., 35., 40., 45.],
       [ 6., 12., 18., 24., 30., 36., 42., 48., 54.],
       [ 7., 14., 21., 28., 35., 42., 49., 56., 63.],
       [ 8., 16., 24., 32., 40., 48., 56., 64., 72.],
       [ 9., 18., 27., 36., 45., 54., 63., 72., 81.]])
```

Python

```
>>> a = np.arange(10)
>>> a[5]      #
5
>>> a[3:5]   # a[3]a[5]
array([3, 4])
```

```

>>> a[:5]    # a[0]
array([0, 1, 2, 3, 4])
>>> a[:-1]  #
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
>>> a[2:4] = 100,101    #
>>> a
array([ 0,  1, 100, 101,  4,  5,  6,  7,  8,  9])
>>> a[1:-1:2]  # 2
array([ 1, 101,  5,  7])
>>> a[::-1] # -1
array([ 9,  8,  7,  6,  5,  4, 101, 100,  1,  0])
>>> a[5:1:-2] #
array([ 5, 101])

```

Python

```

>>> b = a[3:7] # bba
>>> b
array([101, 4, 5, 6])
>>> b[2] = -10 # b2-10
>>> b
array([101, 4, -10, 6])
>>> a # a510
array([ 0,  1, 100, 101,  4, -10,  6,  7,  8,  9])

```

NumPy

```

>>> x = np.arange(10,1,-1)
>>> x
array([10, 9, 8, 7, 6, 5, 4, 3, 2])
>>> x[[3, 3, 1, 8]] # x3, 3, 1, 84
array([7, 7, 9, 2])
>>> b = x[np.array([3,3,-3,8])] #
>>> b[2] = 100
>>> b
array([7, 7, 100, 2])
>>> x # bxx
array([10, 9, 8, 7, 6, 5, 4, 3, 2])
>>> x[[3,5,1]] = -1, -2, -3 #
>>> x

```

```
array([10, -3,  8, -1,  6, -2,  4,  3,  2])
```

bxxbTrue

```
>>> x = np.arange(5,0,-1)
>>> x
array([5, 4, 3, 2, 1])
>>> x[np.array([True, False, True, False, False])]
>>> # 02Truem0,2
array([5, 3])
>>> x[[True, False, True, False, False]]
>>> # True1, False0x
array([4, 5, 4, 5, 5])
>>> x[np.array([True, False, True, True])]
>>> # False
array([5, 3, 2])
>>> x[np.array([True, False, True, True])] = -1, -2, -3
>>> #
>>> x
array([-1,  4, -2, -3,  1])
```

ufuncufunc ufunc

```
>>> x = np.random.rand(10) # 100-1
>>> x
array([ 0.72223939,  0.921226 ,  0.7770805 ,  0.2055047 ,  0.1750
       0.95799412,  0.12015178,  0.7627083 ,  0.43260184,  0.9130
>>> x>0.5
>>> # x>0.5Truem0.5
array([ True,  True,  True, False, False,  True, False,  True, Fa
>>> x[x>0.5]
>>> # x>0.5xx0.5
array([ 0.72223939,  0.921226 ,  0.7770805 ,  0.95799412,  0.762
       0.91379859])
```

NumPy(tuple)

a6x6

Python x,y=y,x

```
>>> a[0,3:5]
array([3,4])
>>> a[4:,4:]
array([[44,45],[54,55]])
>>> a[:,2]
array([2,12,22,32,42,52])
>>> a[2::2,:2]
array([[20,22,24],
       [40,42,44]])
```

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55



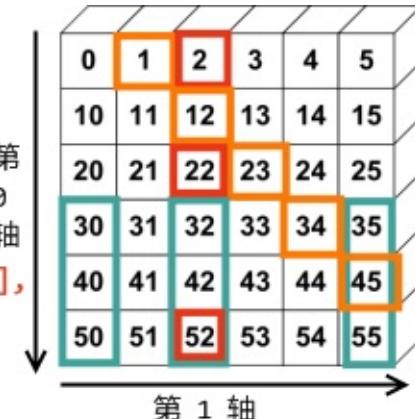
aa0, 10, 20, 30, 40, 500, 1, 2, 3, 4, 5a

```
>>> np.arange(0, 60, 10).reshape(-1, 1) + np.arange(0, 6)
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15],
       [20, 21, 22, 23, 24, 25],
       [30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
```

```

>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([1,12,23,34,45])
>>> a[3:,[0,2,5]]
array([[30,32,35],
       [40,42,45],
       [50,52,55]])
>>> mask=np.array([1,0,1,0,0,1],
                  dtype=np.bool)
>>> a[mask,2]
array([2,22,52])

```



- $a[(0,1,2,3,4),(1,2,3,4,5)]$: 01 $a[0,1]$, $a[1,2]$, ..., $a[4,5]$
- $a[3:,[0, 2, 5]]$: 0310, 2, 5
- $a[mask, 2]$: 02512

CstructCNumPyNumPyCNumPyCNumPy

name, age weight NumPy

```

import numpy as np
persontype = np.dtype({
    'names': ['name', 'age', 'weight'],
    'formats': ['S32', 'i', 'f']})
a = np.array([('Zhang', 32, 75.5), ('Wang', 24, 65.2)],
            dtype=persontype)

```

dtype person type names formats names formats

- **S32** : 32
- **i** : 32bit np.int32
- **f** : 32bit np.float32

array dtype=person type el Python a

```
>>> a.dtype
dtype([('name', '|S32'), ('age', '<i4'), ('weight', '<f4')])
```

(,) '|', '<'

- | :
- < :
- > :

```
>>> a[0]
('Zhang', 32, 75.5)
>>> a[0].dtype
dtype([('name', '|S32'), ('age', '<i4'), ('weight', '<f4')])
```

a[0]a

```
>>> c = a[1]
>>> c["name"] = "Li"
>>> a[1]["name"]
"Li"
```

```
>>> a[0]["name"]
'Zhang'
```

b[0]a[0]["age"]

```
>>> b=a[:, "age"] # a["age"]
>>> b
array([32, 24])
>>> b[0] = 40
>>> a[0]["age"]
40
```

a.tostringa.tofilea

```
>>> a.tofile("test.bin")
```

Ctest.bin

```
Cname[32]name[30]nameagenumpyC  
dtypealign=TruenumpyC
```

```
#include <stdio.h>

struct person
{
    char name[32];
    int age;
    float weight;
};

struct person p[2];

void main ()
{
    FILE *fp;
    int i;
    fp=fopen("test.bin","rb");
    fread(p, sizeof(struct person), 2, fp);
    fclose(fp);
    for(i=0;i<2;i++)
        printf("%s %d %f\n", p[i].name, p[i].age, p[i].weight);
    getchar();
}
```

f1f1f216bit

```
>>> np.dtype([('f1', [('f2', np.int16)])])
dtype([('f1', [('f2', '<i2')])])
```

f1shape(2,3):

```
>>> np.dtype([('f0', 'i4'), ('f1', 'f8', (2, 3))])
```

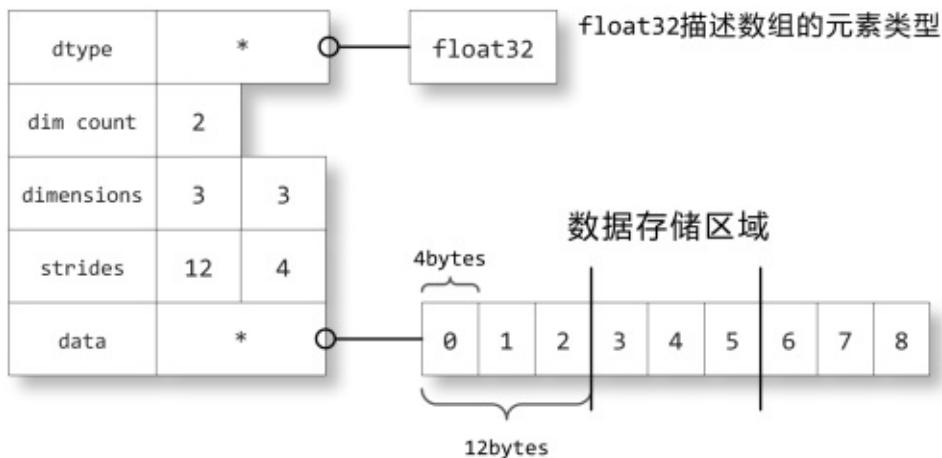
```
dtype([('f0', '<i4'), ('f1', '<f8', (2, 3))])
```

age25

```
>>> np.dtype({'surname':('S25',0), 'age':(np.uint8,25)})  
dtype([('surname', '|S25'), ('age', '|u1')])
```

ndarray dtype

ndarray数据结构



ndarray

dtypendarray

```
>>> a = np.array([[0,1,2],[3,4,5],[6,7,8]], dtype=np.float32)
```

strides1strides12,40112a[1,0]a[0,0]123114

strides

```
>>> b = a[::-2,::2]  
>>> b  
array([[ 0.,  2.],  
       [ 6.,  8.]], dtype=float32)
```

```
>>> b.strides  
(24, 8)
```

bab01abstrides24,8a 0280624

CFortanC001Fortan001NumPyC
Fortanorder="F"

```
>>> c = np.array([[0,1,2],[3,4,5],[6,7,8]], dtype=np.float32, order='F')  
>>> c.strides  
(4, 12)
```

ufunc

ufunc universal function NumPy ufunc C

```
>>> x = np.linspace(0, 2*np.pi, 10)
# x
>>> y = np.sin(x)
>>> y
array([ 0.00000000e+00,   6.42787610e-01,   9.84807753e-01,
        8.66025404e-01,   3.42020143e-01,  -3.42020143e-01,
       -8.66025404e-01,  -9.84807753e-01,  -6.42787610e-01,
      -2.44921271e-16])
```

linspace 0 2 * PI 10 sin np.sin ufunc xy xsinx
ufunc:

```
>>> t = np.sin(x, x)
>>> x
array([ 0.00000000e+00,   6.42787610e-01,   9.84807753e-01,
        8.66025404e-01,   3.42020143e-01,  -3.42020143e-01,
       -8.66025404e-01,  -9.84807753e-01,  -6.42787610e-01,
      -2.44921271e-16])
>>> id(t) == id(x)
True
```

sinxxxxid(tx)

numpy.math Python math.sin:

```
import time
import math
import numpy as np

x = [i * 0.001 for i in xrange(1000000)]
start = time.clock()
for i, t in enumerate(x):
    x[i] = math.sin(t)
print "math.sin:", time.clock() - start

x = [i * 0.001 for i in xrange(1000000)]
x = np.array(x)
```

```
start = time.clock()
np.sin(x,x)
print "numpy.sin:", time.clock() - start

#
# math.sin: 1.15426932753
# numpy.sin: 0.0882399858083
```

100numpy.sinmath.sin10numpy.sinCnumpy.sin
numpy.sin(0.5)math.sinnumpy.sin

```
x = [i * 0.001 for i in xrange(1000000)]
start = time.clock()
for i, t in enumerate(x):
    x[i] = np.sin(t)
print "numpy.sin loop:", time.clock() - start

#
# numpy.sin loop: 5.72166965355
```

numpy.sinmath.sin1/5numpy.sinCmath.sinPython
numpy.sinmath.sinmath.sinPythonfloatnumpy.sin
numpy.float64

```
>>> type(math.sin(0.5))
<type 'float'>
>>> type(np.sin(0.5))
<type 'numpy.float64'>
```

mathnumpy*import numpy as np

NumPyufuncsinadd:

```
>>> a = np.arange(0,4)
>>> a
array([0, 1, 2, 3])
>>> b = np.arange(1,5)
>>> b
array([1, 2, 3, 4])
>>> np.add(a,b)
array([1, 3, 5, 7])
>>> np.add(a,b,a)
```

```
array([1, 3, 5, 7])
>>> a
array([1, 3, 5, 7])
```

add3add

Python a + b np.add(a, b, a) a += bufunc "/
__future__.division

y = x1 +
x2: add(x1, x2 [, y])

y = x1 -
x2: subtract(x1, x2 [, y])

y = x1 *
x2: multiply (x1, x2 [, y])

y = x1 /
x2: divide (x1, x2 [, y]),

y = x1 /
x2: true divide (x1, x2 [, y]),

y = x1 //

x2: floor divide (x1, x2 [, y]),

y = -x:
negative(x [,y])

y =
x1**x2: power(x1, x2 [, y])

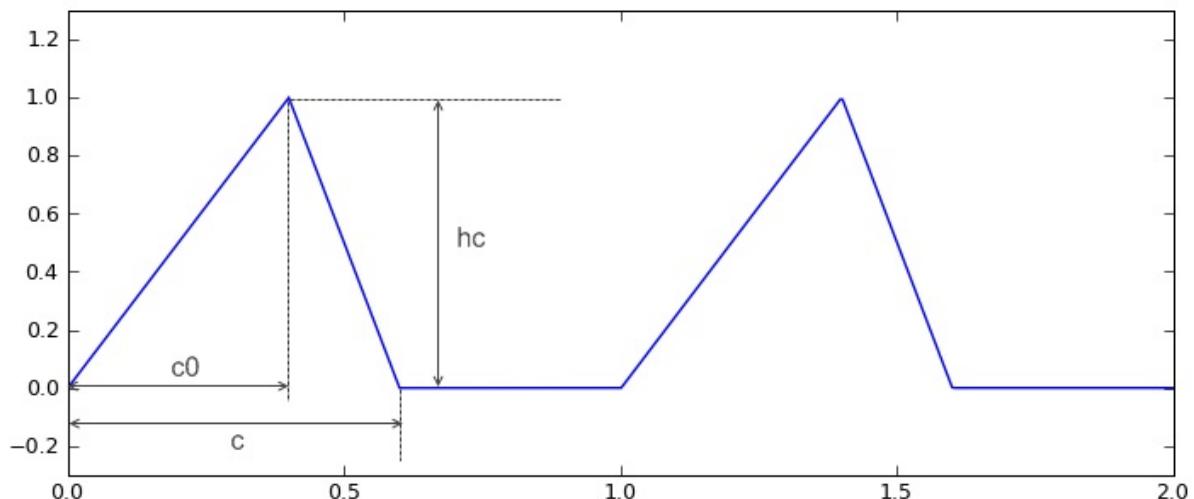
y = x1 %
x2: remainder(x1, x2 [, y]), mod(x1, x2, [, y])

a b c x = a * b + c:

```
t = a * b
x = t + c
del t
```

txx = a * b; x += c

ufunc Python from pyfunc ufunc ufunc



y:

```
def triangle_wave(x, c, c0, hc):
    x = x - int(x) # 1x
    if x >= c: r = 0.0
    elif x < c0: r = x / c0 * hc
    else: r = (c-x) / (c-c0) * hc
    return r
```

triangle_wave(List comprehension)listarray

```
x = np.linspace(0, 2, 1000)
y = np.array([triangle_wave(t, 0.6, 0.4, 1.0) for t in x])
```

frompyfunc

```
triangle_ufunc = np.frompyfunc( lambda x: triangle_wave(x, 0.6, 0
y2 = triangle_ufunc(x)
```

frompyfuncfrompyfunc(func, nin, nout)funcninnout
triangle_wave4c, c0, hcufunclambdatriangle_wave
frompyfunc

```
def triangle_func(c, c0, hc):
    def trifunc(x):
```

```

x = x - int(x) # 1x
if x >= c: r = 0.0
elif x < c0: r = x / c0 * hc
else: r = (c-x) / (c-c0) * hc
return r

# trifuncufunc,
# Object
return np.frompyfunc(trifunc, 1, 1)

y2 = triangle_func(0.6, 0.4, 1.0)(x)

```

triangle_func trifunc trifunc triangle_func triangle_func
frompyfunc

frompyfunc object frompyfunc Python y2.astype(np.float64)

ufuncufunc(shape) shape(broadcasting)

1. shape shape1
2. shape shape
3. 1
4. 1

4

a shape(6,1)

```

>>> a = np.arange(0, 60, 10).reshape(-1, 1)
>>> a
array([[ 0], [10], [20], [30], [40], [50]])
>>> a.shape
(6, 1)

```

b shape(5,)

```

>>> b = np.arange(0, 5)
>>> b

```

```
array([0, 1, 2, 3, 4])
>>> b.shape
(5, )
```

aba,bshape(6,5)

```
>>> c = a + b
>>> c
array([[ 0,  1,  2,  3,  4],
       [10, 11, 12, 13, 14],
       [20, 21, 22, 23, 24],
       [30, 31, 32, 33, 34],
       [40, 41, 42, 43, 44],
       [50, 51, 52, 53, 54]])
>>> c.shape
(6, 5)
```

abshape(ndim)1bshapeabshape1(1,5)

```
>>> b.shape=1,5
>>> b
array([[0, 1, 2, 3, 4]])
```

shape(6,1)(1,5)2shape(6,5)

b01a060b06

```
>>> b = b.repeat(6, axis=0)
>>> b
array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]])
```

a11b51a15

```
>>> a = a.repeat(5, axis=1)
>>> a
array([[ 0,  0,  0,  0,  0],
       [10, 10, 10, 10, 10],
```

```
[20, 20, 20, 20, 20],  
[30, 30, 30, 30, 30],  
[40, 40, 40, 40, 40],  
[50, 50, 50, 50, 50]])
```

ab

numpya+b1repeat

numpya,b ogrid

```
>>> x,y = np.ogrid[0:5,0:5]  
>>> x  
array([[0],  
       [1],  
       [2],  
       [3],  
       [4]])  
>>> y  
array([[0, 1, 2, 3, 4]])
```

ogrid

- ::np.arange(, ,)
- ::jnp.linspace(, ,)

```
>>> x, y = np.ogrid[0:1:4j, 0:1:3j]  
>>> x  
array([[ 0.        ],  
       [ 0.33333333],  
       [ 0.66666667],  
       [ 1.        ]])  
>>> y  
array([[ 0. ,  0.5,  1. ]])
```

ogrid

Pythonogridslice

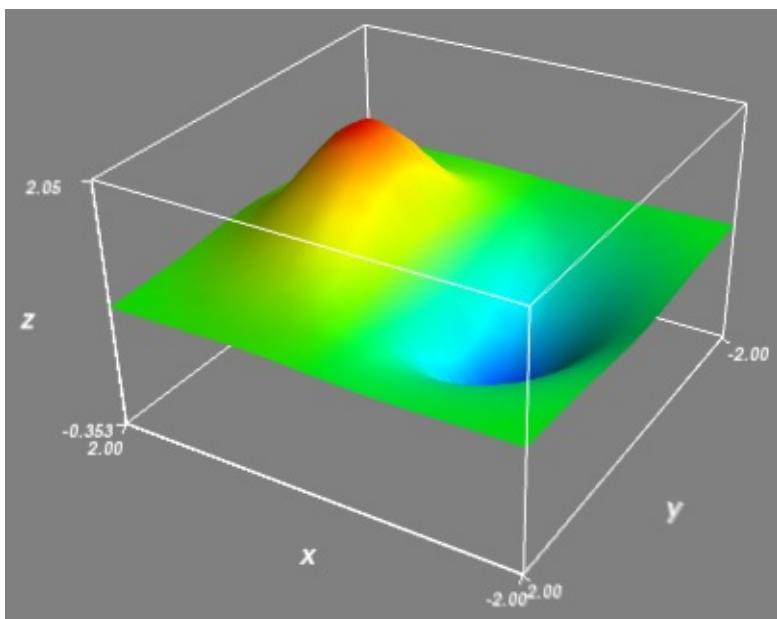
`ogridx, yx, y, z x * exp(x**2 - y**2)`

```
import numpy as np
from enthought.mayavi import mlab

x, y = np.ogrid[-2:2:20j, -2:2:20j]
z = x * np.exp( - x**2 - y**2)

pl = mlab.surf(x, y, z, warp_scale="auto")
mlab.axes(xlabel='x', ylabel='y', zlabel='z')
mlab.outline(pl)
```

mayavimlab3Dmlab



`ogrid`

ufunc

`ufunc``ufunc``ufunc``ValueError`

reduce Python reduce axis array<op>axis

```
<op>.reduce (array=, axis=0, dtype=None)
```

```
>>> np.add.reduce([1,2,3]) # 1 + 2 + 3
6
>>> np.add.reduce([[1,2,3],[4,5,6]], axis=1) # 1,4 + 2,5 + 3,6
array([ 6, 15])
```

accumulate reducereshape

```
>>> np.add.accumulate([1,2,3])
array([1, 3, 6])
>>> np.add.accumulate([[1,2,3],[4,5,6]], axis=1)
array([[ 1,  3,  6],
       [ 4,  9, 15]])
```

reduceat reduceindicesreduceat

```
>>> a = np.array([1,2,3,4])
>>> result = np.add.reduceat(a, indices=[0,1,0,2,0,3,0])
>>> result
array([ 1,  2,  3,  6,  4, 10])
```

indicesreduceindicesresult

```
if indices[i] < indices[i+1]:
    result[i] = np.reduce(a[indices[i]:indices[i+1]])
else:
    result[i] = a[indices[i]]
```

:

```
np.reduce(a[indices[-1]:])
```

```
1 : a[0] = 1
2 : a[1] = 2
3 : a[0] + a[1] = 1 + 2
3 : a[2] = 3
6 : a[0] + a[1] + a[2] = 1 + 2 + 3 = 6
4 : a[3] = 4
10: a[0] + a[1] + a[2] + a[4] = 1+2+3+4 = 10
```

```
result[::2] = result[1::2] * np.add.accumulate(a)
```

outer <op>.outer(a,b)

```
>>> a.shape += (1,) * b.ndim  
>>> <op>(a, b)  
>>> a = a.squeeze()
```

squeeze a1

```
>>> np.multiply.outer([1,2,3,4,5], [2,3,4])  
array([[ 2,   3,   4],  
       [ 4,   6,   8],  
       [ 6,   9,  12],  
       [ 8,  12,  16],  
       [10,  15,  20]])
```

outer

```
#      2,  3,  4  
# 1  
# 2  
# 3  
# 4  
# 5
```

NumPyMatlab

matrix

numpy matrix matlab NumPy ndarray matrix
Python “”matrix matrix

```
>>> a = np.matrix([[1,2,3],[5,5,6],[7,9,9]])
>>> a*a**-1
matrix([[ 1.00000000e+00,   1.66533454e-16,  -8.32667268e-17],
       [-2.77555756e-16,   1.00000000e+00,  -2.77555756e-17],
       [ 1.66533454e-16,   5.55111512e-17,   1.00000000e+00]])
```

a matrix a

dot reshape

```
>>> a = array([1, 2, 3])
>>> a.reshape((-1,1))
array([[1],
       [2],
       [3]])
>>> a.reshape((1,-1))
array([[1, 2, 3]])
```

dot NumPy inner outer

- **dot** : () ab

```
dot(a, b)[i,j,k,m] = sum(a[i,j,:,:] * b[:,k,m])
```

3 dot

```
>>> a = np.arange(12).reshape(2,3,2)
>>> b = np.arange(12,24).reshape(2,2,3)
>>> c = np.dot(a,b)
```

dotca,b

```
>>> np.alltrue( c[0,:,:,:] == np.dot(a[0],b[0]) )
True
>>> np.alltrue( c[1,:,:,:] == np.dot(a[1],b[0]) )
True
>>> np.alltrue( c[0,:,:,:] == np.dot(a[0],b[1]) )
True
>>> np.alltrue( c[1,:,:,:] == np.dot(a[1],b[1]) )
True
```

- **inner** : dotabab

```
inner(a, b)[i,j,k,m] = sum(a[i,j,:]*b[k,m,:])
```

inner

```
>>> a = np.arange(12).reshape(2,3,2)
>>> b = np.arange(12,24).reshape(2,3,2)
>>> c = np.inner(a,b)
>>> c.shape
(2, 3, 2, 3)
>>> c[0,0,0,0] == np.inner(a[0,0],b[0,0])
True
>>> c[0,1,1,0] == np.inner(a[0,1],b[1,0])
True
>>> c[1,2,1,2] == np.inner(a[1,2],b[1,2])
True
```

- **outer** : outer

```
>>> np.outer([1,2,3],[4,5,6,7])
array([[ 4,   5,   6,   7],
       [ 8,  10,  12,  14],
       [12,  15,  18,  21]])
```

NumPy linear algebra solvesolve

```
>>> a = np.random.rand(10,10)
>>> b = np.random.rand(10)
>>> x = np.linalg.solve(a,b)
>>> np.sum(np.abs(np.dot(a,x) - b))
3.1433189384699745e-15
```

solveabaN*NbNsolveNxaxbx

NumPyNumPy

tofiletofilenumpy.fromfile

```
>>> a = np.arange(0,12)
>>> a.shape = 3,4
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> a.tofile("a.bin")
>>> b = np.fromfile("a.bin", dtype=np.float) # float
>>> b #
array([ 2.12199579e-314,   6.36598737e-314,   1.06099790e-313,
       1.48539705e-313,   1.90979621e-313,   2.33419537e-313])
>>> a.dtype # dtype
dtype('int32')
>>> b = np.fromfile("a.bin", dtype=np.int32) # int32
>>> b #
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
>>> b.shape = 3, 4 # ashapebshape
>>> b #
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

dtypeshapetofileCFortranC

fromfiletofilesep

numpy.loadnumpy.savetxtNumPyshapenumpy.save

```
>>> np.save("a.npy", a)
>>> c = np.load( "a.npy" )
>>> c
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
numpy.savezsavezarr_0, arr_1, ...savez(npz)
savenpyloadnpz
```

```
>>> a = np.array([[1,2,3],[4,5,6]])
>>> b = np.arange(0, 1.0, 0.1)
>>> c = np.sin(b)
>>> np.savez("result.npz", a, b, sin_array = c)
>>> r = np.load("result.npz")
>>> r["arr_0"] # a
array([[1, 2, 3],
       [4, 5, 6]])
>>> r["arr_1"] # b
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
>>> r["sin_array"] # c
array([ 0.          ,  0.09983342,  0.19866933,  0.29552021,  0.389...
      0.47942554,  0.56464247,  0.64421769,  0.71735609,  0.783...
```

```
result.npzarr_0.npy arr_1.npy sin_array.npya, b, c
```

```
numpy.savetxtnumpy.loadtxt12
```

```
>>> a = np.arange(0,12,0.5).reshape(4, -1)
>>> np.savetxt("a.txt", a) # '%.18e'
>>> np.loadtxt("a.txt")
array([[ 0. ,  0.5,  1. ,  1.5,  2. ,  2.5],
       [ 3. ,  3.5,  4. ,  4.5,  5. ,  5.5],
       [ 6. ,  6.5,  7. ,  7.5,  8. ,  8.5],
       [ 9. ,  9.5, 10. , 10.5, 11. , 11.5]])
>>> np.savetxt("a.txt", a, fmt="%d", delimiter=",") #
>>> np.loadtxt("a.txt",delimiter=",") #
array([[ 0.,  0.,  1.,  1.,  2.,  2.],
       [ 3.,  3.,  4.,  4.,  5.,  5.],
       [ 6.,  6.,  7.,  7.,  8.,  8.],
       [ 9.,  9., 10., 10., 11., 11.]])
```

```
loadsavenpy
```

```
>>> a = np.arange(8)
>>> b = np.add.accumulate(a)
>>> c = a + b
```

```
>>> f = file("result.npy", "wb")
>>> np.save(f, a) # a,b,cf
>>> np.save(f, b)
>>> np.save(f, c)
>>> f.close()
>>> f = file("result.npy", "rb")
>>> np.load(f) # f
array([0, 1, 2, 3, 4, 5, 6, 7])
>>> np.load(f)
array([ 0,  1,  3,  6, 10, 15, 21, 28])
>>> np.load(f)
array([ 0,  2,  5,  9, 14, 20, 27, 35])
```

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

SciPy-

SciPyNumPySciPyC

$$(x[i], y[i]): y = f(x) \quad f(x) = k*x + b$$

p p S

$$S(\mathbf{p}) = \sum_{i=1}^m [y_i - f(x_i, \mathbf{p})]^2$$

(Least-square fitting)

scipy.optimize.leastsq leastsq

```
1 # -*- coding: utf-8 -*-
2 import numpy as np
3 from scipy.optimize import leastsq
4 import pylab as pl
5
6 def func(x, p):
7     """
8     : A*sin(2*pi*k*x + theta)
9     """
10    A, k, theta = p
11    return A*np.sin(2*np.pi*k*x+theta)
12
13 def residuals(p, y, x):
14     """
15     x, yp
16     """
17     return y - func(x, p)
18
19 x = np.linspace(0, -2*np.pi, 100)
20 A, k, theta = 10, 0.34, np.pi/6 #
21 y0 = func(x, [A, k, theta]) #
22 y1 = y0 + 2 * np.random.randn(len(x)) #
23 p0 = [7, 0.2, 0] #
24
25
26 # leastsq
27 # residuals
28 # p0
29 # args
```

```

30     plsq = leastsq(residuals, p0, args=(y1, x))
31
32     print u":", [A, k, theta]
33     print u":", plsq[0] #
34
35     pl.plot(x, y0, label=u":")
36     pl.plot(x, y1, label=u":")
37     pl.plot(x, func(x, plsq[0]), label=u":")
38     pl.legend()
39     pl.show()

```

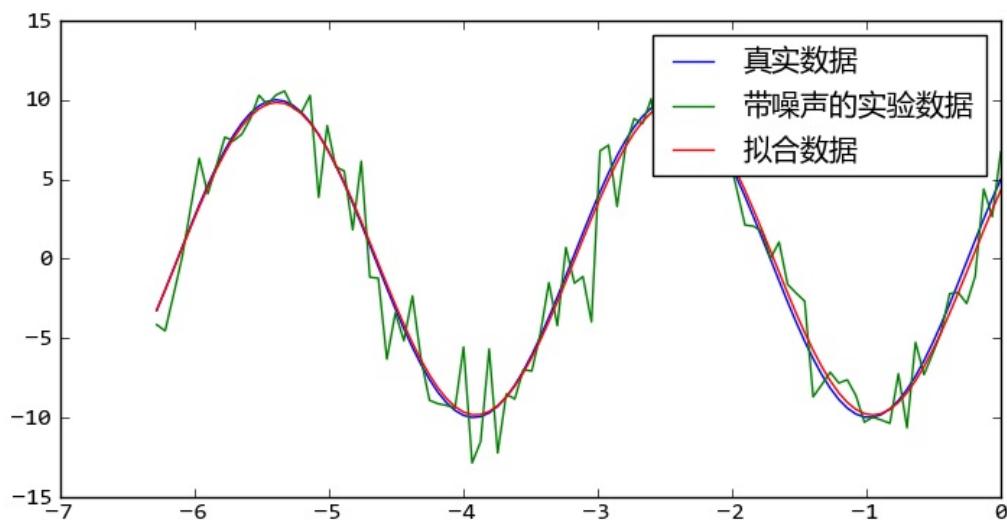
A, k, theta x, y1y1y0

leastsq x, y1xy0 A, k, theta

```

>>> : [10, 0.3400000000000002, 0.52359877559829882]
>>> [-9.84152775 0.33829767 -2.68899335]

```



leastsq

optimizefmin, fmin_powell, fmin_cg, fmin_bfgsfmin

h, xyxh

$$y = x * h$$

xyhhxy

fminfmin

```
1 # -*- coding: utf-8 -*-
2 # fmin
3
4 import scipy.optimize as opt
5 import numpy as np
6
7 def test_fmin_convolve(fminfunc, x, h, y, yn, x0):
8     """
9     x (*) h = y, (*)
10    yny
11    x0x
12    """
13    def convolve_func(h):
14        """
15            yn - x (*) h power
16            fminpower
17        """
18        return np.sum((yn - np.convolve(x, h))**2)
19
20    # fminx0
21    h0 = fminfunc(convolve_func, x0)
22
23    print fminfunc.__name__
24    print "-----"
25    # x (*) h0 y
26    print "error of y:", np.sum((np.convolve(x, h0)-y)**2),
27    # h0 h
28    print "error of h:", np.sum((h0-h)**2)/np.sum(h**2)
29    print
```

```
30
31 def test_n(m, n, nscale):
32     """
33         x, h, y, yn, x0fminb
34         mx, nh, nscale
35     """
36     x = np.random.rand(m)
37     h = np.random.rand(n)
38     y = np.convolve(x, h)
39     yn = y + np.random.rand(len(y)) * nscale
40     x0 = np.random.rand(n)
41
42     test_fmin_convolve(opt.fmin, x, h, y, yn, x0)
43     test_fmin_convolve(opt.fmin_powell, x, h, y, yn, x0)
44     test_fmin_convolve(opt.fmin_cg, x, h, y, yn, x0)
45     test_fmin_convolve(opt.fmin_bfgs, x, h, y, yn, x0)
46
47 if __name__ == "__main__":
48     test_n(200, 20, 0.1)
```

```
fmin

error of y: 0.00568756699607
error of h: 0.354083287918

fmin_powell

error of y: 0.000116114709857
error of h: 0.000258897894009

fmin_cg

error of y: 0.000111220299615
error of h: 0.000211404733439

fmin_bfgs

error of y: 0.000111220251551
error of h: 0.000211405138529
```

optimize fsolve

```
fsolve(func, x0)
```

func(x) $\neq 0$

- $f_1(u_1, u_2, u_3) = 0$
- $f_2(u_1, u_2, u_3) = 0$
- $f_3(u_1, u_2, u_3) = 0$

func

```
def func(x):
    u1, u2, u3 = x
    return [f1(u1, u2, u3), f2(u1, u2, u3), f3(u1, u2, u3)]
```

- $5x_1 + 3 = 0$
- $4x_0 \cdot x_0 - 2 \sin(x_1 \cdot x_2) = 0$
- $x_1 \cdot x_2 - 1.5 = 0$

```
1 from scipy.optimize import fsolve
2 from math import sin,cos
3
4 def f(x):
5     x0 = float(x[0])
6     x1 = float(x[1])
7     x2 = float(x[2])
8     return [
9         5*x1+3,
10        4*x0*x0 - 2*sin(x1*x2),
11        x1*x2 - 1.5
```

```

12     ]
13
14 result = fsolve(f, [1,1,1])
15
16 print result
17 print f(result)

```

```

[-0.70622057 -0.6       -2.5      ]
[0.0, -9.1260332624187868e-14, 5.3290705182007514e-15]

```

fsolveffloatPythonmath

fsolve5064

f1,f2,f3u1,u2,u3

$$\begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} & \frac{\partial f_3}{\partial u_3} \end{bmatrix}$$

fsolvejfprimefsolvejfxjx

```

1 # -*- coding: utf-8 -*-
2 from scipy.optimize import fsolve
3 from math import sin,cos
4 def f(x):
5     x0 = float(x[0])
6     x1 = float(x[1])
7     x2 = float(x[2])
8     return [
9         5*x1+3,

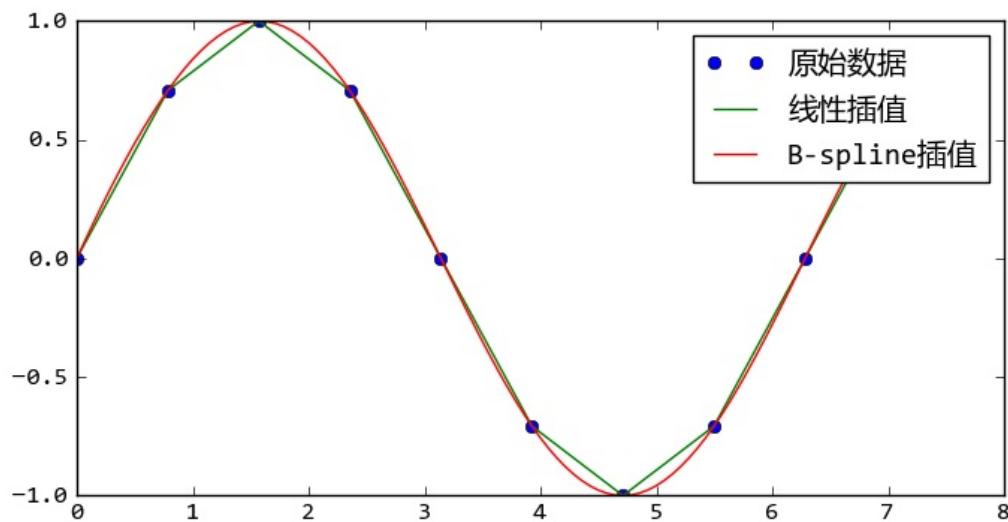
```

```
10         4*x0*x0 - 2*sin(x1*x2),
11         x1*x2 - 1.5
12     ]
13
14 def j(x):
15     x0 = float(x[0])
16     x1 = float(x[1])
17     x2 = float(x[2])
18     return [
19         [0, 5, 0],
20         [8*x0, -2*x2*cos(x1*x2), -2*x1*cos(x1*x2)],
21         [0, x2, x1]
22     ]
23
24 result = fsolve(f, [1,1,1], fprime=j)
25 print result
26 print f(result)
```

B-Spline

interpolateB-Spline

```
1 # -*- coding: utf-8 -*-
2 import numpy as np
3 import pylab as pl
4 from scipy import interpolate
5
6 x = np.linspace(0, 2*np.pi+np.pi/4, 10)
7 y = np.sin(x)
8
9 x_new = np.linspace(0, 2*np.pi+np.pi/4, 100)
10 f_linear = interpolate.interp1d(x, y)
11 tck = interpolate.splrep(x, y)
12 y_bspline = interpolate.splev(x_new, tck)
13
14 pl.plot(x, y, "o", label=u"")
15 pl.plot(x_new, f_linear(x_new), label=u"")
16 pl.plot(x_new, y_bspline, label=u"B-spline")
17 pl.legend()
18 pl.show()
```



interpolateB-Spline

interp1dB-Spline
splrepB-Spline
splev

1PI/2

```
def half_circle(x):
    return (1-x**2)**0.5
```

```
>>> N = 10000
>>> x = np.linspace(-1, 1, N)
>>> dx = 2.0/N
>>> y = half_circle(x)
>>> dx * np.sum(y[:-1] + y[1:]) #
3.1412751679988937
```

numpy.trapz

```
>>> import numpy as np
>>> np.trapz(y, x) * 2 #
3.1415893269316042
```

x,yXtrapz

scipy.integratequad

```
>>> from scipy import integrate
>>> pi_half, err = integrate.quad(half_circle, -1, 1)
>>> pi_half*2
3.1415926535897984
```

quadintegratedblquadtplquaddblquad

(x,y,z)

$$x^2 + y^2 + z^2 = 1$$

(x,y)z

```
def half_sphere(x, y):  
    return (1-x**2-y**2)**0.5
```

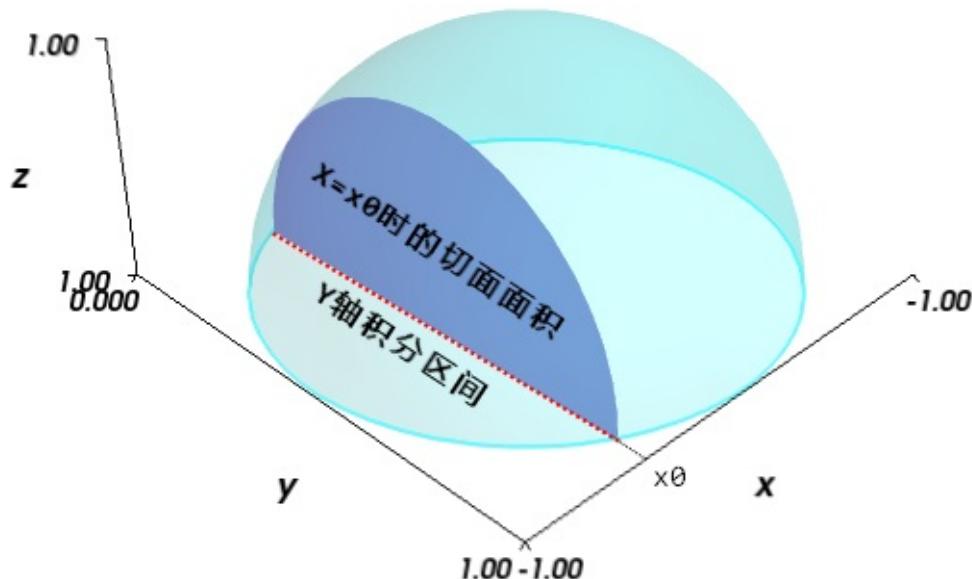
X-YX-11Y -half_circle(x) half_circle(x) dblquad

```
>>> integrate.dblquad(half_sphere, -1, 1,  
lambda x:-half_circle(x),  
lambda x:half_circle(x))  
>>> (2.0943951023931988, 2.3252456653390915e-14)  
>>> np.pi**4/3/2 #  
2.0943951023931953
```

dblquad

```
dblquad(func2d, a, b, gfun, hfun)
```

func2d(x,y)a,bxgfun(x)hfun(x)y



X-1.01.0X=x0YY

scipy.integrateodeintodeint

$$\frac{dx}{dt} = \sigma(y - x)$$

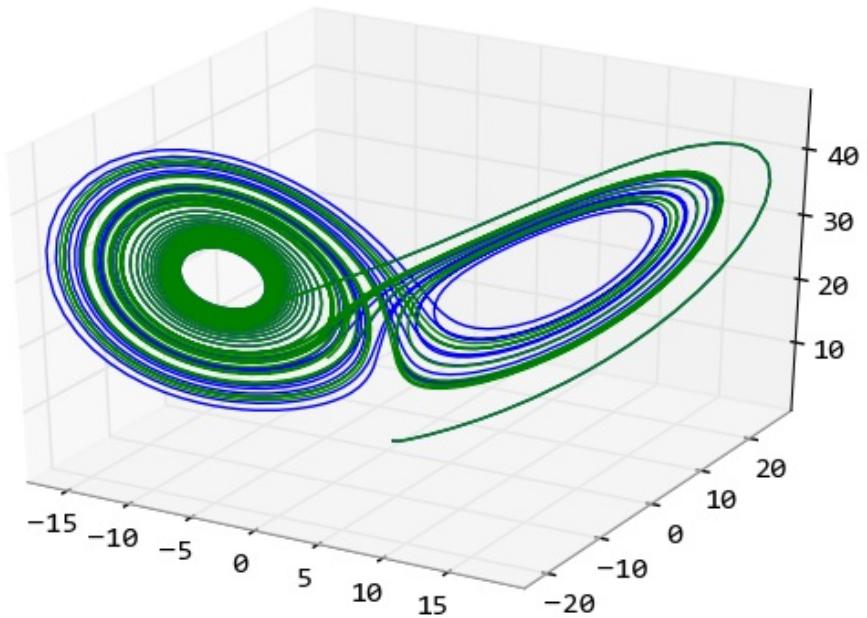
$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

: http://bzhang.lamost.org/website/archives/lorenz_attactor

$$\sigma, \rho, \beta \quad x(t), y(t), z(t)$$

```
1 # -*- coding: utf-8 -*-
2 from scipy.integrate import odeint
3 import numpy as np
4
5 def lorenz(w, t, p, r, b):
6     # wp, r, b
7     # dx/dt, dy/dt, dz/dt
8     x, y, z = w
9     # lorenz
10    return np.array([p*(y-x), x*(r-z)-y, x*y-b*z])
11
12 t = np.arange(0, 30, 0.01) #
13 # odelorenz,
14 track1 = odeint(lorenz, (0.0, 1.00, 0.0), t, args=(10.0, 28))
15 track2 = odeint(lorenz, (0.0, 1.01, 0.0), t, args=(10.0, 28))
16
17 #
18 from mpl_toolkits.mplot3d import Axes3D
19 import matplotlib.pyplot as plt
20
21 fig = plt.figure()
22 ax = Axes3D(fig)
23 ax.plot(track1[:,0], track1[:,1], track1[:,2])
24 ax.plot(track2[:,0], track2[:,1], track2[:,2])
25 plt.show()
```



 `odeint`

`0.01`

`lorenzodeintodeint`

1. `lorenz ()`
2. `(0.0, 1.0, 0.0)`
3. `t` `odeint`
4. `args lorenz`

scipy.signal.signal

signal:

```
>>> import scipy.signal as signal
```

IIR

```
>>> b, a = signal.iirdesign([0.2, 0.5], [0.1, 0.6], 2, 40)
```

0.2*f00.5*f00.1*f00.6*f0f01/28kHz800Hz2kHz2dB
40dB2dB40dB

iirdesginba IIRa[0]1

freqz

```
>>> w, h = signal.freqz(b, a)
```

freqzwhww/pi*f0hw

dBh0clip

```
>>> power = 20*np.log10(np.clip(np.abs(h), 1e-8, 1e100))
```

8kHz

```
>>> pl.plot(w/np.pi*4000, power)
```

chirp28kHz

```
>>> t = np.arange(0, 2, 1/8000.0)
```

chirp2

```
>>> sweep = signal.chirp(t, f0=0, t1 = 2, f1=4000.0)
```

f00Hzf14kHz4kHz2t

lfiltersweep

```
>>> out = signal.lfilter(b, a, sweep)
```

lfilterIIR

xxxy

$$y[n] = b[0]x[n] + b[1]x[n - 1] + \dots + b[P]x[n - P] \\ - a[1]y[n - 1] - a[2]y[n - 2] - \dots - a[Q]y[n - Q]$$

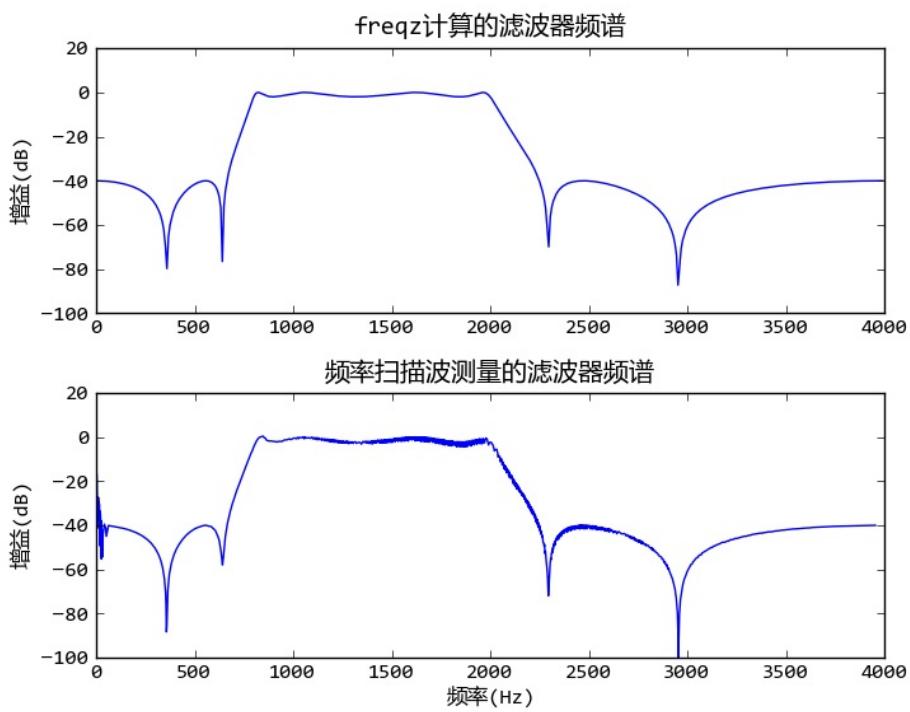
```
>>> out = 20*np.log10(np.abs(out))
```

()

```
>>> index = np.where(np.logical_and(out[1:-1] > out[:-2], out[1:-1] <
```

```
>>> pl.plot(t[index]/2.0*4000, out[index] )
```

freqz



IIR

WeaveC

PythonPythonCNumPy, SciPyCFortranPython
PythonSciPyC++-- WeaveNumPy

```
1  # -*- coding: utf-8 -*-
2  import scipy.weave as weave
3  import numpy as np
4  import time
5
6  def my_sum(a):
7      n=int(len(a))
8      code=""""
9          int i;
10
11         double counter;
12         counter =0;
13         for(i=0;i<n;i++){
14             counter=counter+a(i);
15         }
16         return_val=counter;
17     """
18
19     err=weave.inline(
20         code,['a','n'],
21         type_converters=weave.converters.blitz,
22         compiler="gcc"
23     )
24     return err
25
26 a = np.arange(0, 10000000, 1.0)
27 # my_sumweaveC
28 my_sum(a)
29
30 start = time.clock()
31 for i in xrange(100):
32     my_sum(a)  #
33 print "my_sum:", (time.clock() - start) / 100.0
34
35 start = time.clock()
36 for i in xrange(100):
37     np.sum( a ) # numpysumC
```

```
38 print "np.sum:", (time.clock() - start) / 100.0
39
40 start = time.clock()
41 print sum(a) # Python sum a
42 print "sum:", time.clock() - start
```

```
my_sum: 0.0294527349146
np.sum: 0.0527649547638
sum: 9.11022322669
```

Weave Cnumpysum Python sum Python Weave 1/300

weave.inline C++ weave Python an C++
converters.blitz numpy C++ blitz C++ a blitz a[i]
compiler weave gcc C++ python(x,y) gcc(mingw32) gcc
Visual C++

Note: Visual C++ 2008 Express Visual C++ Python
gcc

weave Python

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

SymPy-

SymPyPython^{sympy}

```
>>> from sympy import *
```

$$e^{i\pi} + 1 = 0$$

ei π 5SymPy

Eipi

```
>>> E**(I*pi)+1  
0
```

$$e^{ix} = \cos x + i \sin x$$

SymPyxSymPySymbol

```
>>> x = Symbol('x')
```

expandE**(I*pi)

```
>>> expand(E**(I*x))  
exp(I*x)
```

expmath.expm numpy.expsympy.exp

expandcomplexTrueexpand

```
>>> expand(exp(I*x), complex=True)  
I*exp(-im(x))*sin(re(x)) + cos(re(x))*exp(-im(x))
```

sin, cos, re, imsympyreimxxx

```
>>> x = Symbol("x", real=True)  
>>> expand(exp(I*x), complex=True)
```

```
I*sin(x) + cos(x)
```

```
>>> tmp = series(exp(I*x), x, 0, 10)
>>> pprint(tmp)
      2      3      4      5      6      7      8      9
      x      I*x     x      I*x     x      I*x     x      I*x
1 + I*x - ----- + ----- + ----- - ----- + ----- + ----- + 0
      2      6      24     120     720    5040   40320   362880
```

seriespprinttmpcos(x)sin(x)

```
>>> pprint(re(tmp))
      2      4      6      8
      x      x      x      x
1 + re(0(x**10)) - --- + --- - --- + -----
      2      24     720    40320
```

```
>>> pprint(series( cos(x), x, 0, 10 ) )
      2      4      6      8
      x      x      x      x
1 - --- + --- - --- + ----- + 0(x**10)
      2      24     720    40320
```

```
>>> pprint(im(tmp))
      3      5      7      9
      x      x      x      x
x + im(0(x**10)) - --- + --- - --- + -----
      6      120    5040   362880
```

```
>>> pprint(series(sin(x), x, 0, 10))
      3      5      7      9
      x      x      x      x
x - --- + --- - --- + ----- + 0(x**10)
      6      120    5040   362880
```

SciPy SymPy integrate integrate

```
>>> integrate(x*sin(x), x)
-x*cos(x) + sin(x)
```

xintegrate

```
>>> integrate(x*sin(x), (x, 0, 2*pi))
-2*pi
```

rY

$$y(x) = \sqrt{r^2 - x^2}$$

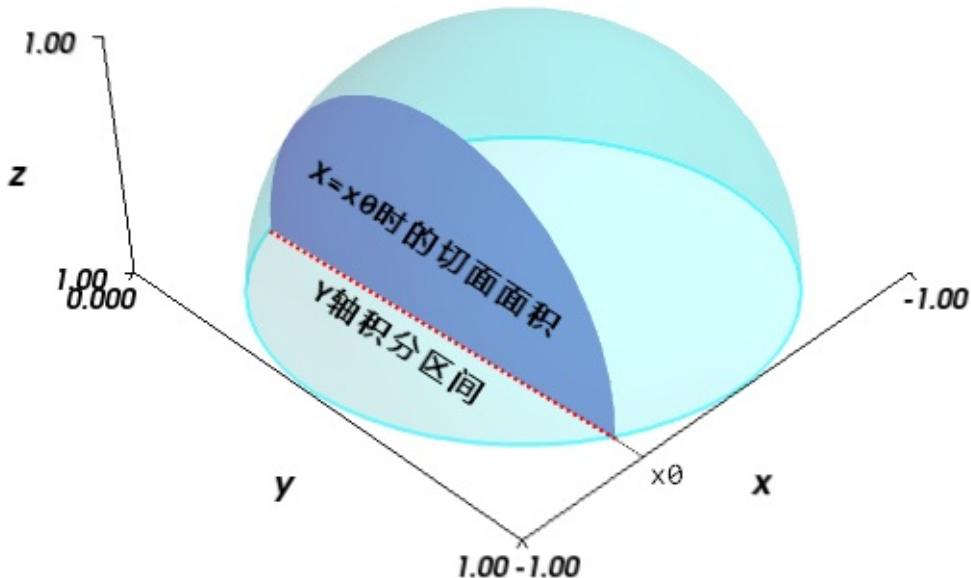
-rrsymbols

```
>>> x, y, r = symbols('x,y,r')
>>> 2 * integrate(sqrt(r*r-x**2), (x, -r, r))
2*Integral((r**2 - x**2)**(1/2), (x, -r, r))
```

integrateSymPyrOr

```
>>> r = symbols('r', positive=True)
>>> circle_area = 2 * integrate(sqrt(r**2-x**2), (x, -r, r))
>>> circle_area
pi*r**2
```

XXxrxy(x)



circle_arear

```
>>> circle_area = circle_area.subs(r, sqrt(r**2-x**2))
>>> circle_area
pi*(r**2 - x**2)
```

subs

subs3

- `expression.subs(x, y)` : xy
- `expression.subs({x:y,u:v})` :
- `expression.subs([(x,y),(u,v)])` :

```
expression.sub([(x,y),(y,x)])
```

x,y

circle_areax-rr

```
>>> integrate(circle_area, (x, -r, r))  
4*pi*r**3/3
```

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

matplotlib-

[matplotlib](#) [python](#)[matlab](#)[API](#)[GUI](#)

[Gallery](#) //

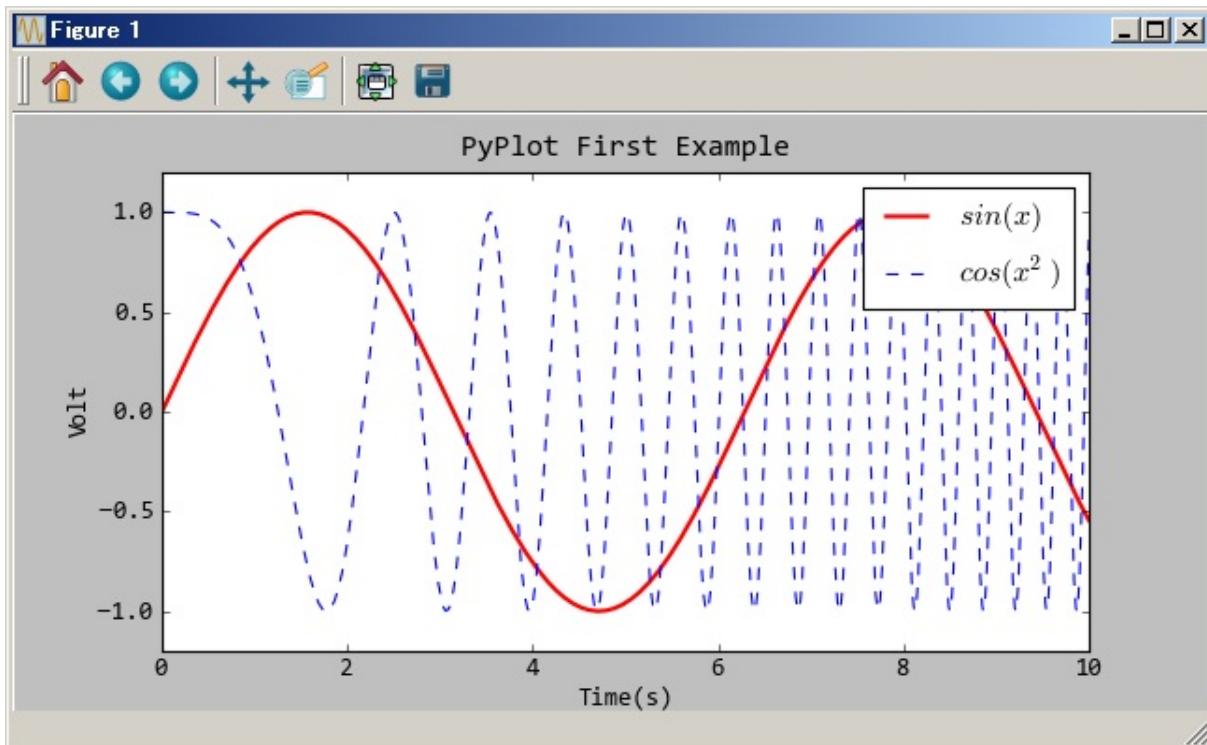
[matplotlib](#)[matplotlib](#)

matplotlib pyplot matlab API 2D

```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x)
z = np.cos(x**2)

plt.figure(figsize=(8,4))
plt.plot(x,y,label="$\sin(x)$",color="red",linewidth=2)
plt.plot(x,z,"b--",label="$\cos(x^2)$")
plt.xlabel("Time(s)")
plt.ylabel("Volt")
plt.title("PyPlot First Example")
plt.ylim(-1.2,1.2)
plt.legend()
plt.show()
```





matplotlib

```
import matplotlib.pyplot as plt
```

pylab

matplotlibpylabnumpypyplotIPython

figure

```
plt.figure(figsize=(8,4))
```

plotmatplotlibfigure

figsizedpi808*80 = 640

png800*400savefigDPIsavefigdpimatplotlib

```
>>> import matplotlib
>>> matplotlib.rcParams["savefig.dpi"]
100
```

plot

```
plt.plot(x,y,label="$\sin(x)$",color="red",linewidth=2)
plt.plot(x,z,"b--",label="$\cos(x^2)$")
```

plotx,yplot

- **label** : (legend)\$\$"matplotlib latex"
- **color** :
- **linewidth** :

"b--"b"--"IPython "plt.plot?"

```
plt.xlabel("Time(s)")  
plt.ylabel("Volt")  
plt.title("PyPlot First Example")  
plt.ylim(-1.2,1.2)  
plt.legend()
```

- **xlabel** : X
- **ylabel** : Y
- **title** :
- **ylim** : Y
- **legend** :

```
plt.show()
```

```
matplotlibset_*pyplotsetpplot matplotlib.lines.Line2D  
Line2D
```

```
>>> import numpy as np  
>>> import matplotlib.pyplot as plt  
>>> x = np.arange(0, 5, 0.1)  
>>> line, = plt.plot(x, x*x) # plotline,  
>>> # Line2Dset_*
```



```
>>> line.set_antialiased(False)  
  
>>> # sincoslinesLine2D  
>>> lines = plt.plot(x, np.sin(x), x, np.cos(x)) #  
>>> # setpLine2D  
>>> plt.setp(lines, color="r", linewidth=2.0)
```

```
Line2Dlineset_antialiasedplt.setpLine2D
```

```
Line2Dget_*plt.getp
```

```
>>> line.get_linewidth()  
1.0
```

```
>>> plt.getp(lines[0], "color") # color
'r'
>>> plt.getp(lines[1]) #
alpha = 1.0
animated = False
antialiased or aa = True
axes = Axes(0.125,0.1;0.775x0.8)
...
...
```

getp

•
•

matplotlibFigureplt.figureplt.gcf

```
>>> f = plt.gcf()
>>> plt.getp(f)
alpha = 1.0
animated = False
...
```

FigureaxesAxesSubplotAxesSubplotplt.gca

```
>>> plt.getp(f, "axes")
[<matplotlib.axes.AxesSubplot object at 0x05CDD170>]
>>> plt.gca()
<matplotlib.axes.AxesSubplot object at 0x05CDD170>
```

plt.getpAxesSubplotlines Line2D

```
>>> alllines = plt.getp(plt.gca(), "lines")
>>> alllines
<a list of 3 Line2D objects>
>>> alllines[0] == line #
True
```

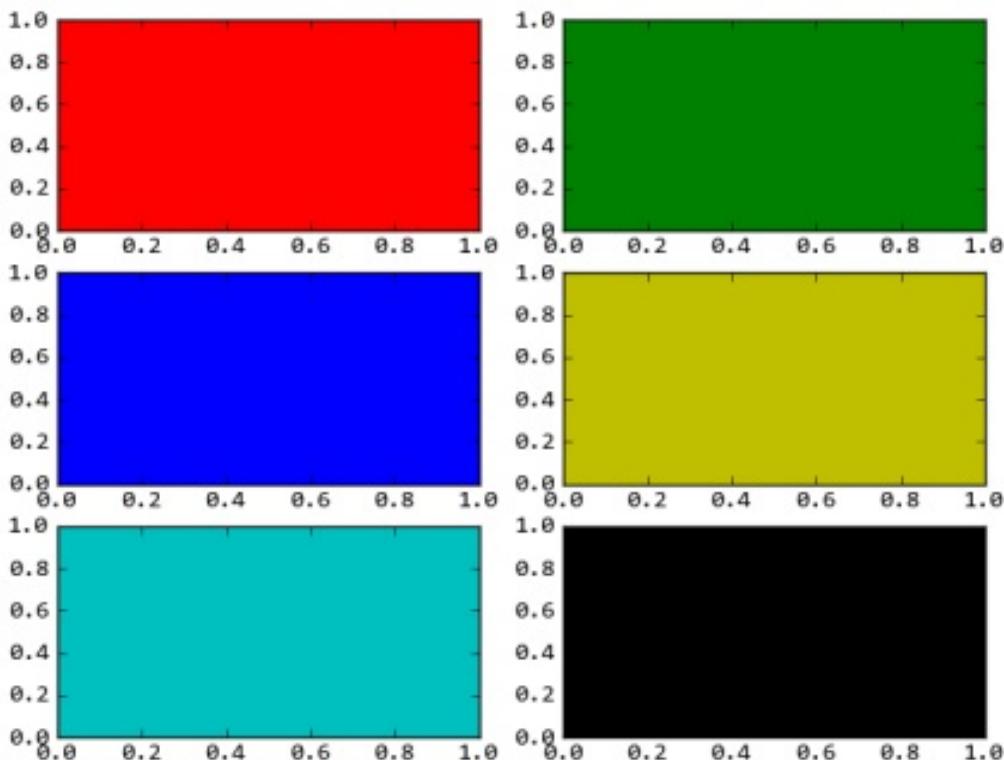
```
(figure)(axis)Matplotlib()subplotsubplot
```

```
    subplot(numRows, numCols, plotNum)
```

```
subplotnumRows * numCols1numRowsnumCols  
plotNum10subplot(323)subplot(3,2,3)subplotplotNum
```

```
326axisbg
```

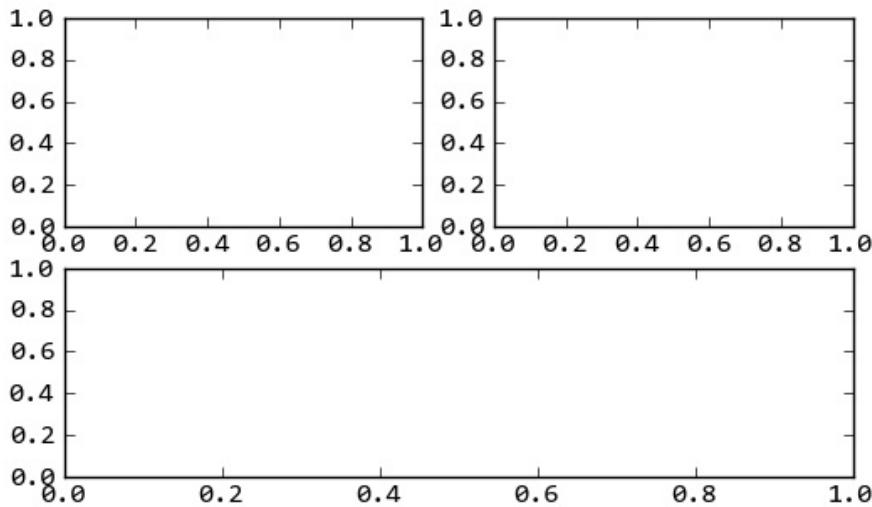
```
for idx, color in enumerate("rgbyck"):  
    plt.subplot(320+idx+1, axisbg=color)  
plt.show()
```



```
subplotFigure
```

```
subplot
```

```
plt.subplot(221) #
plt.subplot(222) #
plt.subplot(212) #
plt.show()
```



Figure

Configure Subplotssubplots_adjustleft, right,
bottom, top, wspace, hspace011

Matplotlib

matplotlibrc

- :
- : HOME/.matplotlib/MATPLOTLIBRC
- : matplotlib mpl-data

```
>>> import matplotlib
>>> matplotlib.get_configdir()
'C:\\Documents and Settings\\zhang\\.matplotlib'
```

```
>>> import matplotlib
>>> matplotlib.matplotlib_fname()
'C:\\Python26\\lib\\site-packages\\matplotlib\\mpl-data\\matplotlibrc'
```

matplotlibrc

```
>>> import os
>>> os.getcwd()
'C:\\zhang\\doc'
```

:

```
>>> matplotlib.matplotlib_fname()
'C:\\zhang\\doc\\matplotlibrc'
```

rc_params

```
>>> matplotlib.rcParams()
{'agg.path.chunksize': 0,
 'axes.axisbelow': False,
 'axes.edgecolor': 'k',
 'axes.facecolor': 'w',
 ... ...
```

matplotlibrc_paramsrcParams

```
>>> matplotlib.rcParams
{'agg.path.chunksize': 0,
 'axes.axisbelow': False,
 ... ...
```

matplotlibrcParams

```
>>> matplotlib.rcParams["lines.marker"] = "o"
>>> import pylab
>>> pylab.plot([1,2,3])
>>> pylab.show()
```

rc

```
>>> matplotlib.rc("lines", marker="x", linewidth=2, color="red")
```

(matplotlib) rcdefaults

```
>>> matplotlib.rcParams.update( matplotlib.rcParams() )
```

Artist

matplotlib API

- **backend_bases.FigureCanvas** :
- **backend_bases.Renderer** : FigureCanvas
- **artist.Artist** : RendererFigureCanvas

FigureCanvasRendererwxPythonPostScriptPDFArtist
Artist

ArtistsArtistsLine2D Rectangle TextAxesImage
ArtistsAxis AxesFigure

Artists

- Figure
- FigureAxesSubplot
- AxiesArtists

pyplot.figureFigureFigureadd_axesAxesadd_axes[left, bottom,
width, height]Axesfig01

```
>>> import matplotlib.pyplot as plt
>>> fig = plt.figure()
>>> ax = fig.add_axes([0.15, 0.1, 0.7, 0.3])
```

axplot(Line2D)

```
>>> line, = ax.plot([1, 2, 3], [1, 2, 1])
>>> ax.lines
[<matplotlib.lines.Line2D object at 0x0637A3D0>]
>>> line
<matplotlib.lines.Line2D object at 0x0637A3D0>
```

ax.linesaxax.plot

AxesArtist.set_xlabelX

```
>>> ax.set_xlabel("time")
```

set_xlabel

```
self.xaxis.set_label_text(xlabel)
```

Axes.xaxis.XAxis

```
>>> ax.xaxis  
<matplotlib.axis.XAxis object at 0x06343230>
```

XAxislabelText

```
>>> ax.xaxis.label  
<matplotlib.text.Text object at 0x06343290>
```

Text_text

```
>>> ax.xaxis.label._text  
'time'
```

Artists

```
>>> ax.xaxis.label.get_text()  
'time'
```

Artist

matplotlibArtistArtistFigureAxespatchRectangleFigrue

```
>>> fig = plt.figure()  
>>> fig.show()  
>>> fig.patch.set_color("g")  
>>> fig.canvas.draw()
```

```
patchcolorset_colorfig.canvas.draw()
```

Artist

- alpha : 0101
- animated :
- axes : ArtistAxesNone
- clip_box :
- clip_on :
- clip_path :
- contains :
- figure : FigureNone
- label :
- picker : Artist
- transform :
- visible :
- zorder :

Artist get_* set_* alpha

```
>>> fig.set_alpha(0.5*fig.get_alpha())
```

set

```
>>> fig.set(alpha=0.5, zorder=2)
```

matplotlib.pyplot.getp Artist

```
>>> plt.getp(fig.patch)
aa = True
alpha = 1.0
animated = False
antialiased or aa = True
...
...
```

Figure

ArtistArtistArtist

Artistmatplotlib.figure.FigureRectangleFigure.patch
add_subplotadd_axes()Figure.axesaxesAxesSubplot
Axes

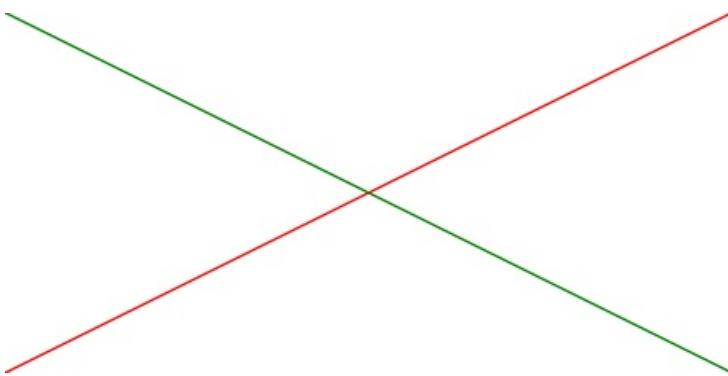
```
>>> fig = plt.figure()  
>>> ax1 = fig.add_subplot(211)  
>>> ax2 = fig.add_axes([0.1, 0.1, 0.7, 0.3])  
>>> ax1  
<matplotlib.axes.AxesSubplot object at 0x056BCA90>  
>>> ax2  
<matplotlib.axes.Axes object at 0x056BC910>  
>>> fig.axes  
[<matplotlib.axes.AxesSubplot object at 0x056BCA90>,  
<matplotlib.axes.Axes object at 0x056BC910>]
```

pylabgca()FigureFigure.axesadd_subplot, add_axes,
delaxesforaxes

```
>>> for ax in fig.axes: ax.grid(True)
```

FigureArtistArtisttransformFigure(0,0)(1,1)fig

```
>>> from matplotlib.lines import Line2D  
>>> fig = plt.figure()  
>>> line1 = Line2D([0,1],[0,1], transform=fig.transFigure, figure=fig)  
>>> line2 = Line2D([0,1],[1,0], transform=fig.transFigure, figure=fig)  
>>> fig.lines.extend([line1, line2])  
>>> fig.show()
```



Figure

```
Line2Dfigfig.TransFigureLine2DtransformLine2Dfigfigure  
figLine2Dfig.lines
```

FigureArtist

- axes : Axes
- patch : Rectangle
- images : FigureImage
- legends : Legend
- lines : Line2D
- patches : patch
- texts : Text

Axes

```
AxesmatplotlibArtistFigurepatchpatch  
RectanglepatchCircleAxes
```

```
>>> fig = plt.figure()  
>>> ax = fig.add_subplot(111)  
>>> ax.patch.set_facecolor("green")
```

```
AxesplotLine2DLine2DAxes.linesLine2D
```

```
>>> x, y = np.random.rand(2, 100)  
>>> line, = ax.plot(x, y, "-", color="blue", linewidth=2)  
>>> line  
<matplotlib.lines.Line2D object at 0x03007030>  
>>> ax.lines  
[<matplotlib.lines.Line2D object at 0x03007030>]
```

```
plotLine2DX,Y
```

```
plotbarhistPatchPatchRectanglePatchAxes.patches
```

```
>>> ax = fig.add_subplot(111)  
>>> n, bins, rects = ax.hist(np.random.randn(1000), 50, facecolor="red")  
>>> rects
```

```
<a list of 50 Patch objects>
>>> rects[0]
<matplotlib.patches.Rectangle object at 0x05BC2350>
>>> ax.patches[0]
<matplotlib.patches.Rectangle object at 0x05BC2350>
```

Axes.linesAxes.patchesadd_lineadd_patch

```
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111)
>>> rect = matplotlib.patches.Rectangle((1,1), width=5, height=
>>> print rect.get_axes() # rectaxes
None
>>> rect.get_transform() # recttransform
BboxTransformTo(Bbox(array([[ 1.,   1.],
                           [ 6.,  13.]])))
>>> ax.add_patch(rect) # rectax
<matplotlib.patches.Rectangle object at 0x05C34E50>
>>> rect.get_axes() # rectaxesax
<matplotlib.axes.AxesSubplot object at 0x05C09CB0>
```

```
>>> # recttransformaxtransData
>>> rect.get_transform()
...
>>> ax.transData
...
#
```

```
>>> ax.get_xlim() # axX01rect
(0.0, 1.0)
>>> ax.dataLim._get_bounds() # rect
(1.0, 1.0, 5.0, 12.0)
>>> ax.autoscale_view() #
>>> ax.get_xlim() # Xrect
(1.0, 6.0)
>>> plt.show()
```

add_patchrectaxestransform

AxesArtist

- artists : Artist
- patch : AxesPatchRectangleCircle
- collections : Collection

- images : AxesImage
- legends : Legend
- lines : Line2D
- patches : Patch
- texts : Text
- xaxis : XAxis
- yaxis : YAxis

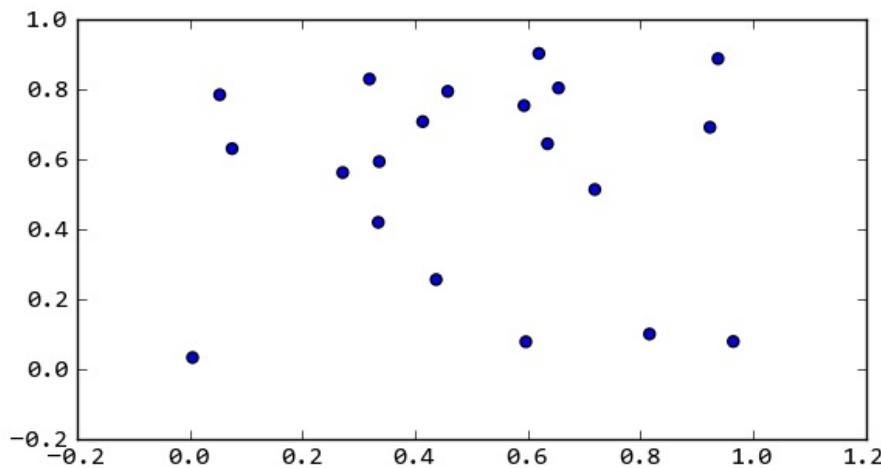
AxesArtist

Axes

annotate	Annotate	texts
bars	Rectangle	patches
errorbar	Line2D, Rectangle	lines, patches
fill	Polygon	patches
hist	Rectangle	patches
imshow	AxesImage	images
legend	Legend	legends
plot	Line2D	lines
scatter	PolygonCollection	Collections
text	Text	texts

(scatter)

```
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111)
>>> t = ax.scatter(np.random.rand(20), np.random.rand(20))
>>> t # CircleCollection
<matplotlib.collections.CircleCollection object at 0x06004230>
>>> ax.collections # collections
[<matplotlib.collections.CircleCollection object at 0x06004230>]
>>> fig.show()
>>> t.get_sizes() # Collection
20
```



scatter

Axis

Axis
Axis.get_major_ticks
Axis.get_minor_ticks
XTick
YTick
Axis.get_ticklabels
Axis.get_ticklines

```
>>> pl.plot([1,2,3],[4,5,6])
[<matplotlib.lines.Line2D object at 0x0AD3B670>]
>>> pl.show()
>>> axis = pl.gca().xaxis
```

```
>>> axis.get_ticklocs() #
array([ 1. ,  1.5,  2. ,  2.5,  3. ])
```

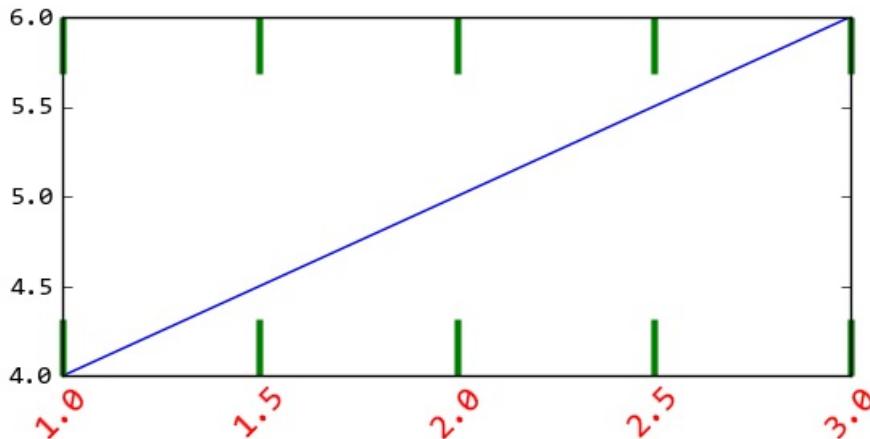
```
>>> axis.get_ticklabels() #
<a list of 5 Text major ticklabel objects>
>>> [x.get_text() for x in axis.get_ticklabels()] #
[u'1.0', u'1.5', u'2.0', u'2.5', u'3.0']
```

```
>>> axis.get_ticklines() # 10
<a list of 10 Line2D ticklines objects>
```

```
>>> axis.get_ticklines(minor=True) #
<a list of 0 Line2D ticklines objects>
```

```
>>> for label in axis.get_ticklabels():
...     label.set_color("red")
...     label.set_rotation(45)
...     label.set_fontsize(16)
... 
```

```
>>> for line in axis.get_ticklines():
...     line.set_color("green")
...     line.set_markersize(25)
...     line.set_markeredgewidth(3) 
```



NullLocatorAutoLocator

```
>>> axis.get_minor_locator() #
<matplotlib.ticker.NullLocator instance at 0x0A014300>
>>> axis.get_major_locator() #
<matplotlib.ticker.AutoLocator instance at 0x09281B20>
```

AxisLocatorFormatterXpi/4pi/20pi

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as pl
from matplotlib.ticker import MultipleLocator, FuncFormatter
import numpy as np
x = np.arange(0, 4*np.pi, 0.01)
```

```

y = np.sin(x)
pl.figure(figsize=(8,4))
pl.plot(x, y)
ax = pl.gca()

def pi_formatter(x, pos):
    """
    pi/4
    """
    m = np.round(x / (np.pi/4))
    n = 4
    if m%2==0: m, n = m/2, n/2
    if m%2==0: m, n = m/2, n/2
    if m == 0:
        return "0"
    if m == 1 and n == 1:
        return "\pi"
    if n == 1:
        return r"$%d \pi $" % m
    if m == 1:
        return r"\frac{\pi}{%d}" % n
    return r"\frac{%d \pi}{%d}" % (m,n)

#
# pl.ylim(-1.5,1.5)
# pl.xlim(0, np.max(x))

#
# pl.subplots_adjust(bottom = 0.15)

pl.grid() #

# pi/4
ax.xaxis.set_major_locator( MultipleLocator(np.pi/4) )

# pi_formatter
ax.xaxis.set_major_formatter( FuncFormatter( pi_formatter ) )

# pi/20
ax.xaxis.set_minor_locator( MultipleLocator(np.pi/20) )

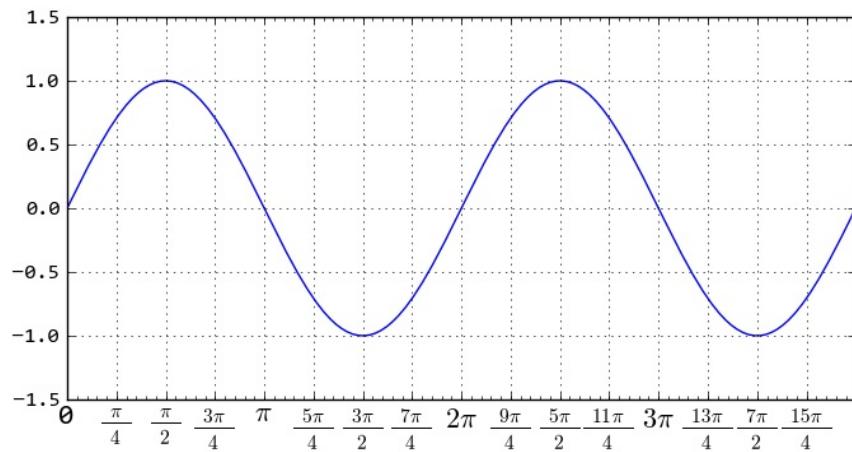
#
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
pl.show()

```

matplotlib.ticker

- **MultipleLocator** :
- **FuncFormatter** :

LocatorFormatterAPI



X

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

Traits-Python

PythonTraits

TraitsChaco(2D)

- 'red'
- 0xff0000
- (255, 0, 0)

Python"abc"0.5

Trait:

-
-
-

traits:

```
from enthought.traits.api import HasTraits, Color

class Circle(HasTraits):
    color = Color
```

enthought.traits.api: HasTraitsColortraitHasTraitsPython
traitColorTraitFactoryCirclecolor

PythonPythonclasscolorCircleCirclecolor__init__color =
ColortraitHasTraittrait:

```
>>> c = Circle()
>>> Circle.color
Traceback (most recent call last):
AttributeError: type object 'Circle' has no attribute 'color'
>>> c.color
wx.Colour(255, 255, 255, 255)
```

Circlecolorccolorwx.Colour(255, 255, 255, 255)

```
>>> c.color = "red"
>>> c.color
wx.Colour(255, 0, 0, 255)
>>> c.color = 0x00ff00
>>> c.color
wx.Colour(0, 255, 0, 255)
>>> c.color = (0, 255, 255)
>>> c.color
wx.Colour(0, 255, 255, 255)
>>> c.color = 0.5
Traceback (most recent call last):
  File "c:\python25\lib\site-packages\Traits-3.1.0-py2.5-win32.egd
    traits\trait_handlers.py", line 175, in error value
TraitError: The 'color' trait of a Circle instance must be a string
(r,g,b) or (r,g,b,a) where r, g, b, and a are integers from 0 to 255;
instance, an integer which in hex is of the form 0xRRGGBB, where RR is
green, and BB is blue or 'aquamarine' or 'black' or 'blue violet'
'brown' or 'cadet blue' or 'coral' or 'cornflower blue' or 'cyan'
... or 'yellow', but a value of 0.5 <type 'float'> was specified.
```

c.color"red"0x00ff00(0, 255, 255)0.5

:

```
>>> c.configure_traits()
True
>>> c.color
wx.Colour(64, 34, 117, 255)
```

c.configure_traits()OKconfigure_traitsTruec.color
iPython -wthreadspyder



 Trait

Traits

traitPython

- trait
- trait
- trait
- trait
- traittrait

trait

```
from enthought.traits.api import Delegate, HasTraits, Instance, Int, Str

class Parent ( HasTraits ):
    # : last_name'Zhang'
    last_name = Str( 'Zhang' )

class Child ( HasTraits ):
    age = Int

    # : fatherParent
    father = Instance( Parent )

    # : Childlast_namefatherlast_name
    last_name = Delegate( 'father' )

    # : age
    def _age_changed ( self, old, new ):
        print 'Age changed from %s to %s ' % ( old, new )
```

:

```
>>> p = Parent()
>>> c = Child()
```

cfatherlast_name:

```
>>> c.last_name
```

```
Traceback (most recent call last):
AttributeError: 'NoneType' object has no attribute 'last_name'
```

fatherclast_name:

```
>>> c.father = p
>>> c.last_name
'Zhang'
```

cage_age_changed:

```
>>> c.age = 4
Age changed from 0 to 4
```

configure_traits:

```
>>> c.configure_traits()
True
```

ctrait



Child

fatherParentfathertratis



ChildFatherfather

fatherLast namechildLast name

print_traitstrait:

```
>>> c.print_traits()
age:      4
father:    <__main__.Parent object at 0x13B49120>
last_name: u'Zhang'
```

gettraitdict:

```
>>> c.get()
{'age': 4, 'last_name': u'Zhang', 'father': <__main__.Parent obje
```

settraitsettrait:

```
>>> c.set(age = 6)
Age changed from 4 to 6
<__main__.Child object at 0x13B494B0>
```

Trait

TraitTraitPythonTraitsTrait

HasTraitsa, add_traitaxTraitFloat3.0

```
>>> from enthought.traits.api import *
>>> a = HasTraits()
>>> a.add_trait("x", Float(3.0))
>>> a.x
3.0
```

HasTraitsbadd_traitbaHasTraitsbab.a

```
>>> b = HasTraits()
>>> b.add_trait("a", Instance(HasTraits))
>>> b.a = a
```

bDelegate()ybaxb.yb.a.xDelegateb"x"
modify=Trueb.yb.a.xb.y10a.x

```
>>> b.add_trait("y", Delegate("a", "x", modify=True))
>>> b.y
3.0
>>> b.y = 10
>>> a.x
10.0
```

Property

PythonPropertyPropertyTraits

```
# -*- coding: utf-8 -*-
# filename: traits_property.py
from enthought.traits.api import HasTraits, Float, Property, cached_property

class Rectangle(HasTraits):
    width = Float(1.0)
    height = Float(2.0)

    #area width, height_get_area
    area = Property(depends_on=['width', 'height'])

    # cached_property decorator_get_area
    @cached_property
    def _get_area(self):
        """
        areagetProperty
        """
        print 'recalculating'
        return self.width * self.height
```

RectangleProperty()areaTraitsPropertyPythonTraits

area_get_areaarea_set_areadepends_onwidth

heightarea

_get_area@cached_property_get_areawidthheight

Rectangletraits_property.py!Python -wthread

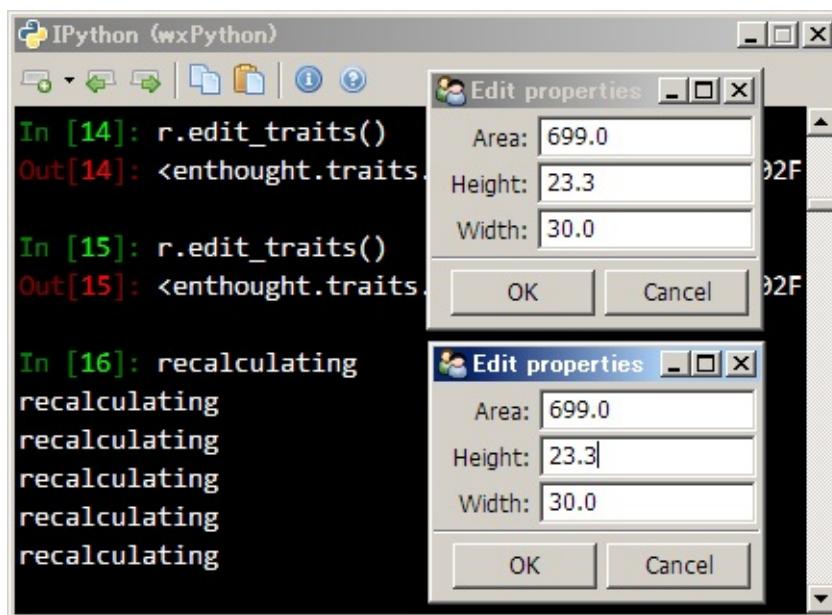
```
>>> run traits_property.py
>>> r = Rectangle()
>>> r.area  # <-- area
recalculating
2.0
>>> r.width = 10
>>> r.area # <--widtharea
recalculating
20.0
>>> r.area # <--widthheight
```

```
20.0
```

```
depends_on@cached_propertyarea_get_areaarea
r.width=10_get_arear.areaarea_get_area

r.edit_traits()depends_onedit_traits
```

```
>>> r.edit_traits()
<enthought.traits.ui.ui.UI object at 0x02FCD420>
>>> r.edit_traits()
<enthought.traits.ui.ui.UI object at 0x02FD68A0>
```



```
HeightWidthArea
```

```
widthheightAreaHeightWidth_get_arealPythonwidth
_get_area
```

```
>>> r.width = 25
recalculating
```

```
rr.widthr.arear.area_get_area
```

```
traits
```

```
# areaTraitarea
```

```
>>> t = r.traits().area
>>> type(t)
<class 'enthought.traits.traits.CTrait'> # <-- CTrait

# _notifiers are
>>> t._notifiers(True)
[<enthought.traits.trait_notifiers.FastUITraitChangeNotifyWrapper
 <enthought.traits.trait_notifiers.FastUITraitChangeNotifyWrapper
```

r.edit_traits()3

Property *Traits*

Trait

HasTraitstraittraitHasTraits

```
# -*- coding: utf-8 -*-
# filename: traits_listener.py
from enthought.traits.api import *

class Child( HasTraits ):
    name = Str
    age = Int
    doing = Str

    def __str__(self):
        return "%s<%x>" % (self.name, id(self))

    # : age
    def _age_changed( self, old, new ):
        print "%s.age changed: from %s to %s" % (self, old, new)

    def _anytrait_changed(self, name, old, new):
        print "anytrait changed: %s.%s from %s to %s" % (self, name, old, new)

    def log_trait_changed(obj, name, old, new):
        print "log: %s.%s changed from %s to %s" % (obj, name, old, new)

    if __name__ == "__main__":
        h = Child(name = "HaiYue", age=4)
        k = Child(name = "KaiYu", age=1)
        h.on_trait_change(log_trait_changed, name="doing")
```

Childage _age_changed _anytrait_changed
HasTraitstrait

log_trait_changedh.on_trait_changehdoinghdoing
log_trait_changed

IPython

```
>>> run traits_listener.py
anytrait changed: <201ba80>.age from 0 to 4
<201ba80>.age changed: from 0 to 4
anytrait changed: HaiYue<201ba80>.name from  to HaiYue
anytrait changed: <201bae0>.age from 0 to 1
<201bae0>.age changed: from 0 to 1
anytrait changed: KaiYu<201bae0>.name from  to KaiYu
```

hk

```
>>> h.age = 5
anytrait changed: HaiYue<5d87e70>.age from 4 to 5
HaiYue<5d87e70>.age changed: from 4 to 5
>>> h.doing = "sleeping"
anytrait changed: HaiYue<5d87e70>.doing from  to sleeping
log: HaiYue<5d87e70>.doing changed from  to sleeping
>>> k.doing = "playing"
anytrait changed: KaiYu<5d874e0>.doing from  to playing
```



Trait

- `_age_changed(self)`
- `_age_changed(self, new)`
- `_age_changed(self, old, new)`
- `_age_changed(self, name, old, new)`

- `observer()`
- `ovserver(new)`
- `ovserver(name, new)`
- `ovserver(obj, name, new)`

- ovserver(obj, name, old, new)

objnameoldnew

trait_xxx_changedTrait@on_trait_changed

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

TraitsUI-

PythonTkinterwxPythonpyQt4API

traitsModel-View-ControllerTraitUI

TraitsUITraitsTraitsHasTraitsconfigure_traitsTraitsUITrait:

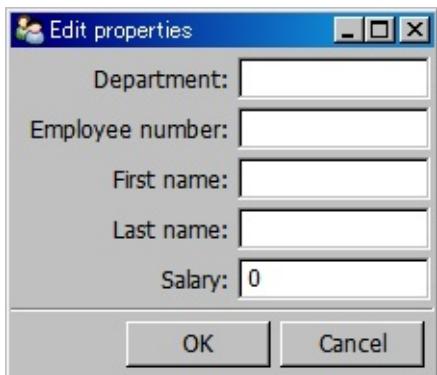
```
from enthought.traits.api import HasTraits, Str, Int

class SimpleEmployee(HasTraits):
    first_name = Str
    last_name = Str
    department = Str

    employee_number = Str
    salary = Int

sam = SimpleEmployee()
sam.configure_traits()
```

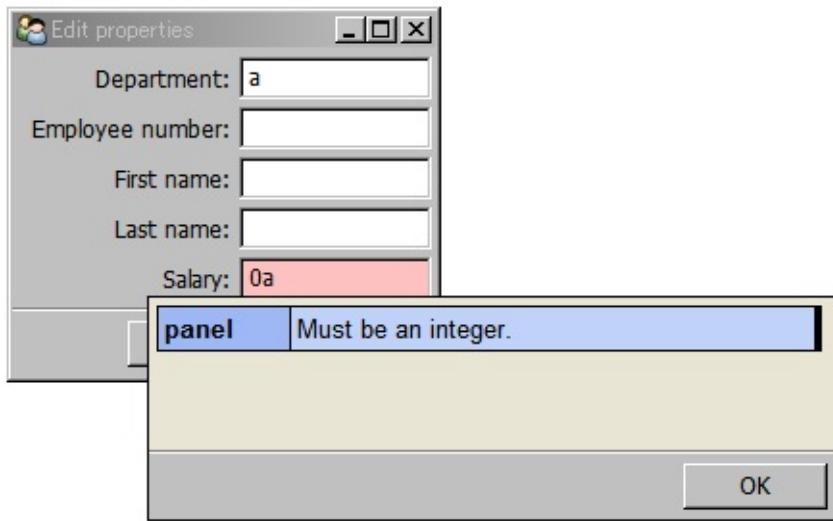
SimpleEmployeesamsam.configure_traits



SimpleEmployee

trait trait OK Cancel trait

salary salary Int Salary salary salary



view1configure_traits

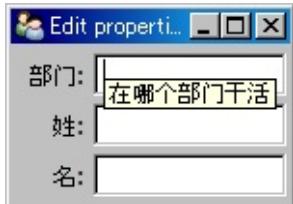
```
# -*- coding: utf-8 -*-
from enthought.traits.api import HasTraits, Str, Int
from enthought.traits.ui.api import View, Item

class SimpleEmployee(HasTraits):
    first_name = Str
    last_name = Str
    department = Str
    employee_number = Str
    salary = Int

    view1 = View(
        Item(name = 'department', label=u"" , tooltip=u"""),
        Item(name = 'last_name', label=u"" ),
        Item(name = 'first_name', label=u"" ))

sam = SimpleEmployee()
sam.configure_traits(view=view1)
```

traits.uiqt4wx -toolikt qt4 -toolikt wx wx



labeltooltip

traits.uiViewItemViewItem()ItemViewtrait

Item

ItemItemHasTraittraitItemItemnamename
SimpleEmployeetraitItemlabeltooltipItemItem
TraitsUiPythonItem??ItemItemtrait

```
class Item( ViewSubElement ):
    """ An element in a Traits-based user interface.

    # Trait definitions:

    # A unique identifier for the item. If not set, it defaults to
    # one of **name**.
    id = Str

    # User interface label for the item in the GUI. If this attribute
    # is set, the label is the value of **name** with slight modification:
    # underscores are replaced by spaces, and the first letter is
    # capitalized. If an item's **name** is not specified, its label is displayed
    # as static text, without any editor widget.
    label = Str

    # Name of the trait the item is editing:
    name = Str
```

ItemTraitsUIItem

- Label
- Heading
- Spring

Viewtrait

Group

ItemViewTraitUIGroup

```
# -*- coding: utf-8 -*-
from enthought.traits.api import HasTraits, Str, Int
from enthought.traits.ui.api import View, Item, Group
```

```

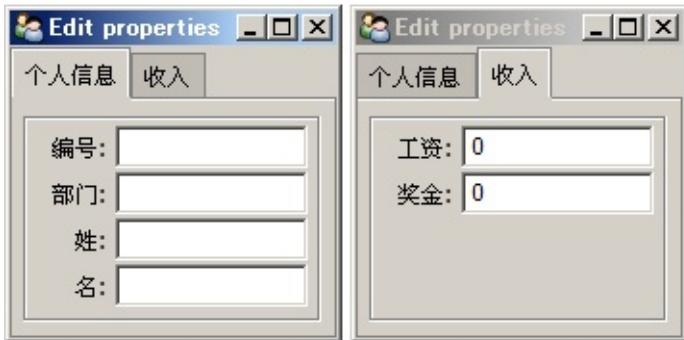
class SimpleEmployee(HasTraits):
    first_name = Str
    last_name = Str
    department = Str

    employee_number = Str
    salary = Int
    bonus = Int

view1 = View(
    Group(
        Item(name = 'employee_number', label=u' '),
        Item(name = 'department', label=u" ", tooltip=u""),
        Item(name = 'last_name', label=u" "),
        Item(name = 'first_name', label=u" "),
        label = u'',
        show_border = True
    ),
    Group(
        Item(name = 'salary', label=u" "),
        Item(name = 'bonus', label=u" "),
        label = u'',
        show_border = True
    )
)

sam = SimpleEmployee()
sam.configure_traits(view=view1)

```



Group

GroupViewGroupItemGrouplabelshow_borderViewGroup
GroupTabGroup

GroupGroupGroup

```
view2 = View( Group( view1.content ) )
```

view2GroupGroupview1(Group)view1

```
view2 = View( Group( Group( Item(name = 'employee_number', label=u''' ), Item(name = 'department', label=u''' , tooltip=u''' Item(name = 'last_name', label=u''' ), Item(name = 'first_name', label=u''' ), label = u''' , show_border = True ), Group( Item(name = 'salary', label=u''' ), Item(name = 'bonus', label=u''' ), label = u''' , show_border = True ) ) )
```

view2configure_traitsview2

```
sam.configure_traits(view=view2)
```



Group

GrouporientationlayoutGroupGroupGroupHSplit

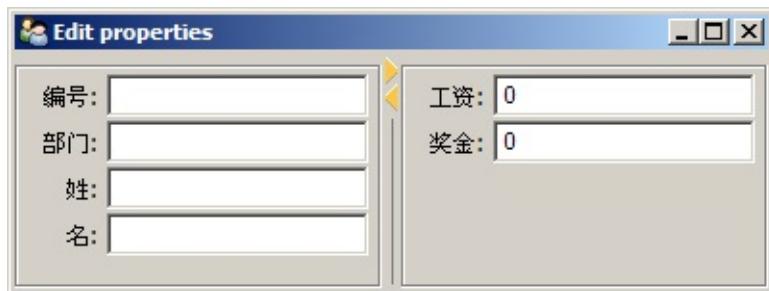
```
class HSplit ( Group ):
    # ...
    layout      = 'split'
    orientation = 'horizontal'
```

HSplitHSplit

```
Group( ... , layout = 'split', orientation = 'horizontal')
```

scrollable(item)

```
view1 = View(
    HSplit(
        VGroup(
            ... ,
            scrollable = True
        ),
        VGroup(
            ... ,
            ),
        resizable = True,
        width = 400,
        height = 150
    )
```



Group

GroupGroup

- **HGroup :**

```
Group(orientation= 'horizontal')
```

- **HFlow** :

```
Group(orientation= 'horizontal', layout='flow', show_labels=False)
```

- **HSplit** :

```
Group(orientation= 'horizontal', layout='split')
```

- **Tabbed** :

```
Group(orientation= 'horizontal', layout='tabbed')
```

- **VGroup** :

```
Group(orientation= 'vertical')
```

- **VFlow** :

```
Group(orientation= 'vertical', layout='flow', show_labels=False)
```

- **VFold** :

```
Group(orientation= 'vertical', layout='fold', show_labels=False)
```

- **VGrid** :

```
Group(orientation= 'vertical', columns=2)
```

- **VSplit**

```
Group(orientation= 'vertical', layout='split')
```

ItemGroup

kindView

- 'modal' : ,
- 'live' :
- 'livemodal' :
- 'nonmodal' :
- 'wizard' :
- 'panel' :
- 'subpanel'

'modal', 'live', 'livemodal', 'nonmodal' View
(OKApply)

'wizard'

'panel'"subpanel' panelsubpanel

OK, Cacel, Apply TraitsUIViewbuttons

TraitsUIUndoButton, ApplyButton, RevertButton, OKButton,
CancelButtonbuttonsButtonUndoButton"Undo"

enthought.tratis.ui.menu

```
OKCancelButtons = ``[OKButton, CancelButton ]``  
ModalButtons = ``[ ApplyButton, RevertButton, OKButton, CancelButton ]``  
LiveButtons = ``[ UndoButton, RevertButton, OKButton, CancelButton ]``
```

Python »

← (Alt+X) | |

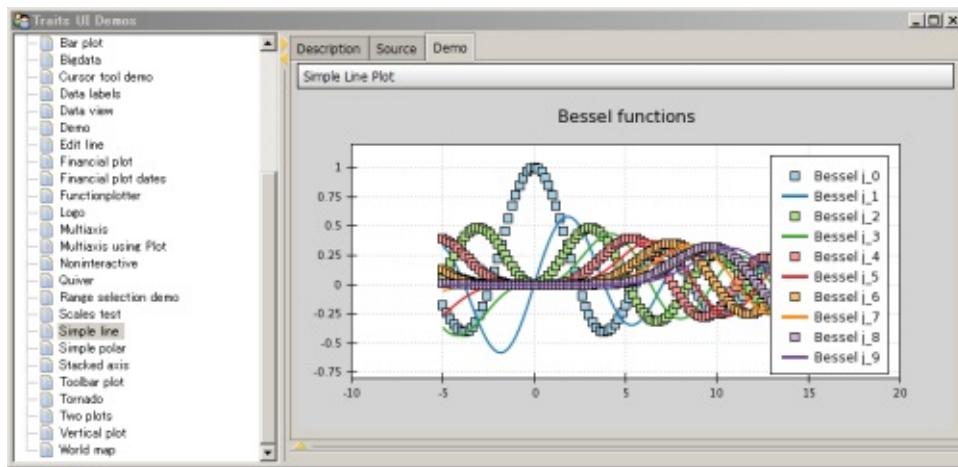
Python »

← (Alt+X) | |

Chaco-

Chaco2D
Python(x,y)pythonxyChacodemo:

```
\pythonxy\doc\Enthought Tool Suite\Chaco\examples\demo.py
```



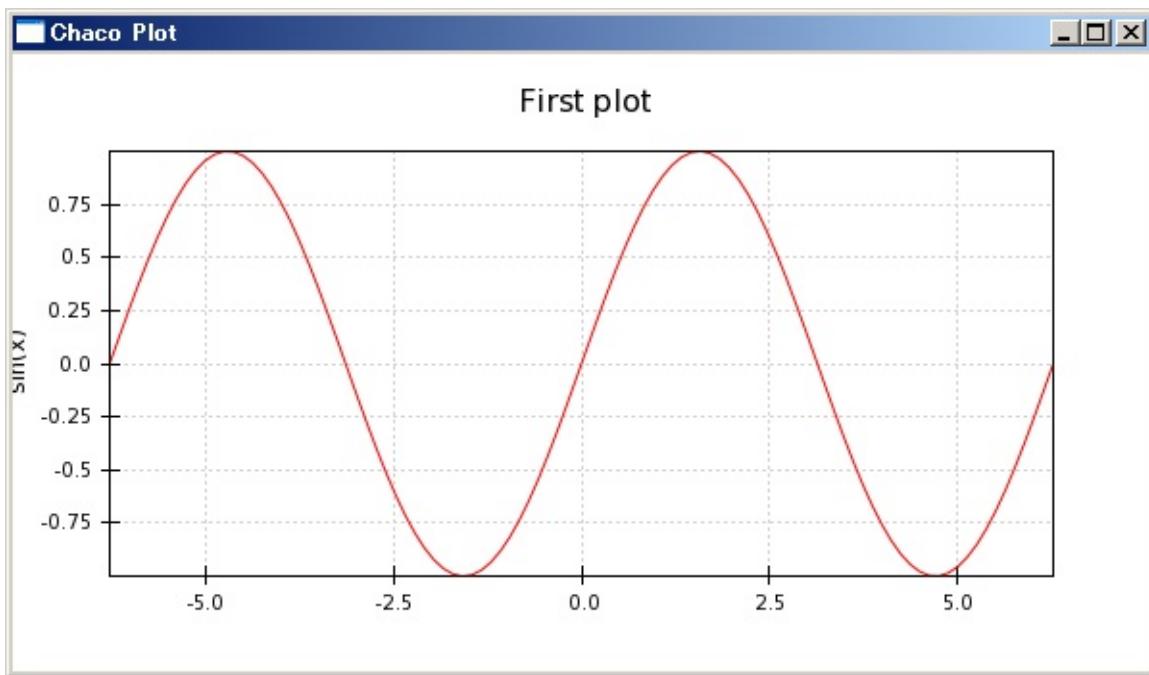
Chaco

SourceDemoDemoChaco

ChacoMatlabpylabenthought.chaco.shellnumpyx,yplot

:

```
1 import numpy as np
2 from enthought.chaco.shell import *
3
4 x = np.linspace(-2*np.pi, 2*np.pi, 100)
5 y = np.sin(x)
6
7 plot(x, y, "r-")
8 title("First plot")
9 ytitle("sin(x)")
10 show()
```



 Chaco

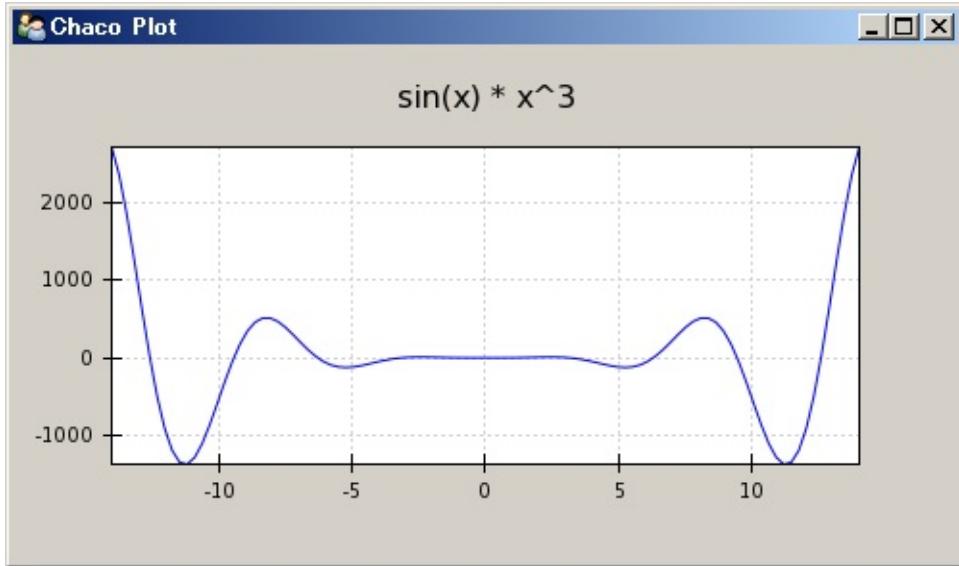
plot"r"""-titleytitleYshow()

ChacoAPIMatplotlibpylabChaco

Chaco:

```
1  from enthought.traits.api import HasTraits, Instance
2  from enthought.traits.ui.api import View, Item
3  from enthought.chaco.api import Plot, ArrayPlotData
4  from enthought.enable.component_editor import ComponentEditor
5  from numpy import linspace, sin
6
7  class LinePlot(HasTraits):
8      plot = Instance(Plot)
9      traits_view = View(
10          Item('plot', editor=ComponentEditor(), show_label=False,
11               width=500, height=500, resizable=True, title="Chaco Plot")
12
13      def __init__(self):
14          super(LinePlot, self).__init__()
15          x = linspace(-14, 14, 100)
16          y = sin(x) * x**3
17          plotdata = ArrayPlotData(x=x, y=y)
18          plot = Plot(plotdata)
19          plot.plot(("x", "y"), type="line", color="blue")
20          plot.title = "sin(x) * x^3"
21          self.plot = plot
22
23  if __name__ == "__main__":
24      LinePlot().configure_traits()
```

:



Chaco

import import

- HasTraits, Instance traits HasTraits Trait LinePlot
Instance Trait
- View, Item traits.ui View Item
- Plot, ArrayPlot Data chaco Plot HasTrait trait
ArrayPlot Data Plot ArrayPlot Data
- ComponentEditor enable Component Editor LinePlot plot
trait Int Str Float GUI Chaco Plot Component Editor Plot
- linspace, sin numpy linspace numpy sin

:

```
class LinePlot(HasTraits):
    plot = Instance(Plot)
```

LinePlot HasTrait trait plot Plot:

```
traits_view = View(
    Item('plot', editor=ComponentEditor(), show_label=False),
    width=500, height=500, resizable=True, title="Chaco Plot")
```

LinePlot configure_traits plot Item editor

```
ComponentEditor(show_label=False)Viewwidthheight  
resizabletitle
```

```
:
```

```
def __init__(self):  
    super(LinePlot, self).__init__()  
    x = linspace(-14, 14, 100)  
    y = sin(x) * x**3  
    plotdata = ArrayPlotData(x=x, y=y)
```

```
HasTraits
```

```
x,yArrayPlotDataArrayPlotData(dict())plot  
ArrayPlotDataArrayPlotDataArrayPlotData
```

```
plotArrayPlotDataplot:
```

```
plot = Plot(plotdata)
```

```
PlotChacoChacoPlotChaco
```

```
plotPlot():
```

```
plot.plot(("x", "y"), type="line", color="blue")
```

```
plotPlotArrayPlotData
```

```
plot:
```

```
plot.title = "sin(x) * x^3"  
self.plot = plot
```

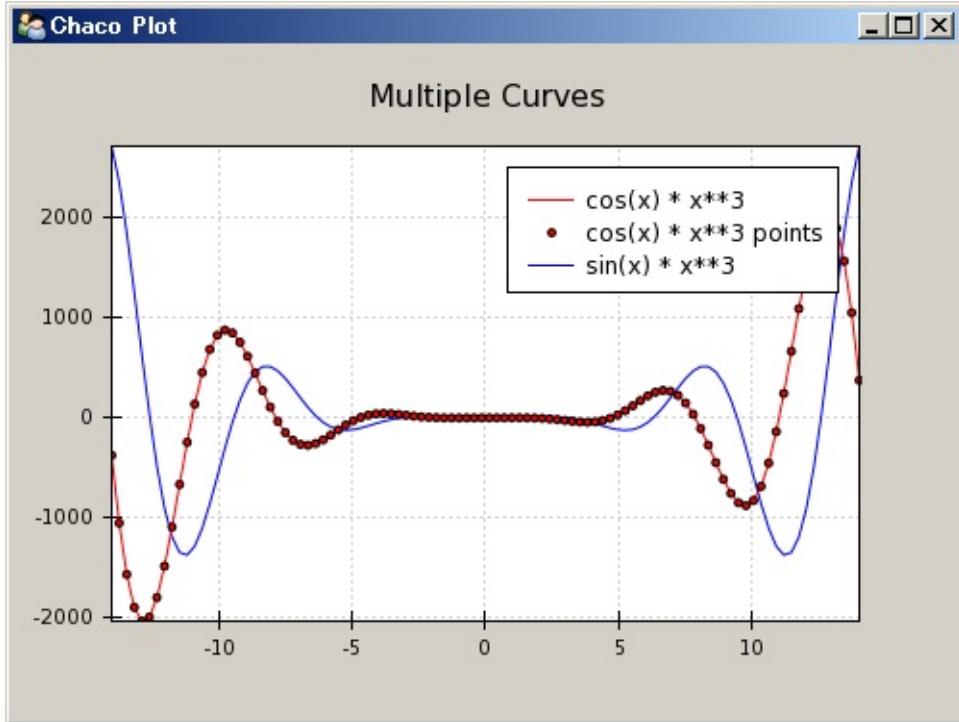
```
LinePlotconfigure_trait:
```

```
if __name__ == "__main__":  
    LinePlot().configure_traits()
```

```
configure_traitsLinePlottraits_viewLinePlottraitplot
traits_viewComponentEditor
```

LinePlot

```
1  from enthought.traits.api import HasTraits, Instance
2  from enthought.traits.ui.api import View, Item
3  from enthought.chaco.api import Plot, ArrayPlotData, Legend
4  from enthought.enable.component_editor import ComponentEditor
5  from numpy import linspace, sin, cos
6
7  class LinePlot(HasTraits):
8      plot = Instance(Plot)
9      traits_view = View(
10          Item('plot', editor=ComponentEditor(), show_label=False,
11               width=500, height=500, resizable=True, title="Chaco Plot")
12
13      def __init__(self):
14          super(LinePlot, self).__init__()
15          x = linspace(-14, 14, 100)
16          y1 = sin(x) * x**3
17          y2 = cos(x) * x**3
18          plotdata = ArrayPlotData(x=x, y1=y1, y2=y2)
19          plot = Plot(plotdata)
20          plot.plot(("x", "y1"), type="line", color="blue", name="sin(x) * x**3")
21          plot.plot(("x", "y2"), type="line", color="red", name="cos(x) * x**3")
22          plot.plot(("x", "y2"), type="scatter", color="red",
23                    marker_size = 2, name="cos(x) * x**3 points")
24          plot.title = "Multiple Curves"
25          self.plot = plot
26
27          legend = Legend(padding=10, align="ur")
28          legend.plots = plot.plots
29          plot.overlays.append(legend)
30
31 if __name__ == "__main__":
32     lineplot = LinePlot()
33     lineplot.configure_traits()
```



3plot.plottype="line" type="scatter" marker
marker_sizeplotname

(legend)chaco.apiLegend:

```
legend = Legend(component=plot, padding=10, align="ur")
legend.plots = plot.plots
plot.overlays.append(legend)
```

Legendpaddingalignpaddingalign: upper right

legendplotplotplotsiPythononlineplot.plot.plotsplots:

```
>>> lineplot.plot.plots
{'cos(x) * x**3 points': [<enthought.chaco.scatterplot.ScatterPlot object at 0x...>],
 'cos(x) * x**3': [<enthought.chaco.lineplot.LinePlot object at 0x...>],
 'sin(x) * x**3': [<enthought.chaco.lineplot.LinePlot object at 0x...>]}
```

plot.plotslegend.plotslegend

legendplot.overlaysoverlaysplotoverlays
TraitListObjectlistlistappendoverlays

(Container)

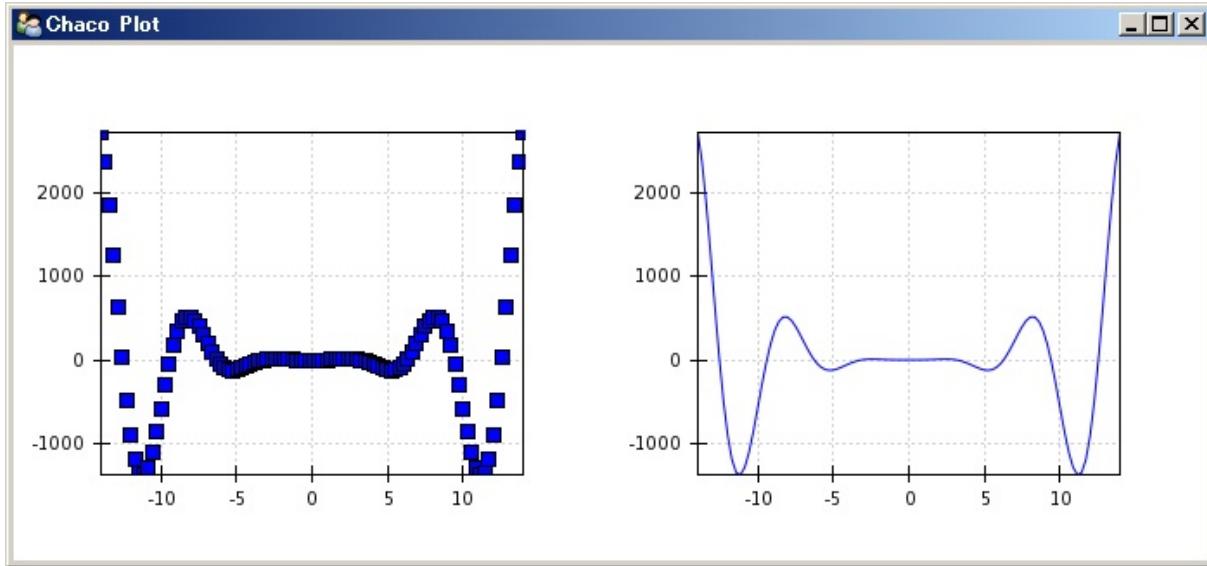
ChacoPlot DataView Overlay Plot Container Plot
Overlay Plot Container Overlay Plot Container

Overlay Plot Container Chaco

- HPlotContainer
- VPlotContainer
- GridPlotContainer

HPlotContainer

```
1  from enthought.traits.api import HasTraits, Instance
2  from enthought.traits.ui.api import View, Item
3  from enthought.chaco.api import HPlotContainer, ArrayPlotData
4  from enthought.enable.component_editor import ComponentEditor
5  from numpy import linspace, sin
6
7  class ContainerExample(HasTraits):
8      plot = Instance(HPlotContainer)
9      traits_view = View(Item('plot', editor=ComponentEditor(),
10                          width=1000, height=600, resizable=True))
11     def __init__(self):
12         super(ContainerExample, self).__init__()
13         x = linspace(-14, 14, 100)
14         y = sin(x) * x**3
15         plotdata = ArrayPlotData(x=x, y=y)
16         scatter = Plot(plotdata)
17         scatter.plot(("x", "y"), type="scatter", color="blue")
18         line = Plot(plotdata)
19         line.plot(("x", "y"), type="line", color="blue")
20         container = HPlotContainer(scatter, line)
21         self.plot = container
22
23 if __name__ == "__main__":
24     ContainerExample().configure_traits()
```

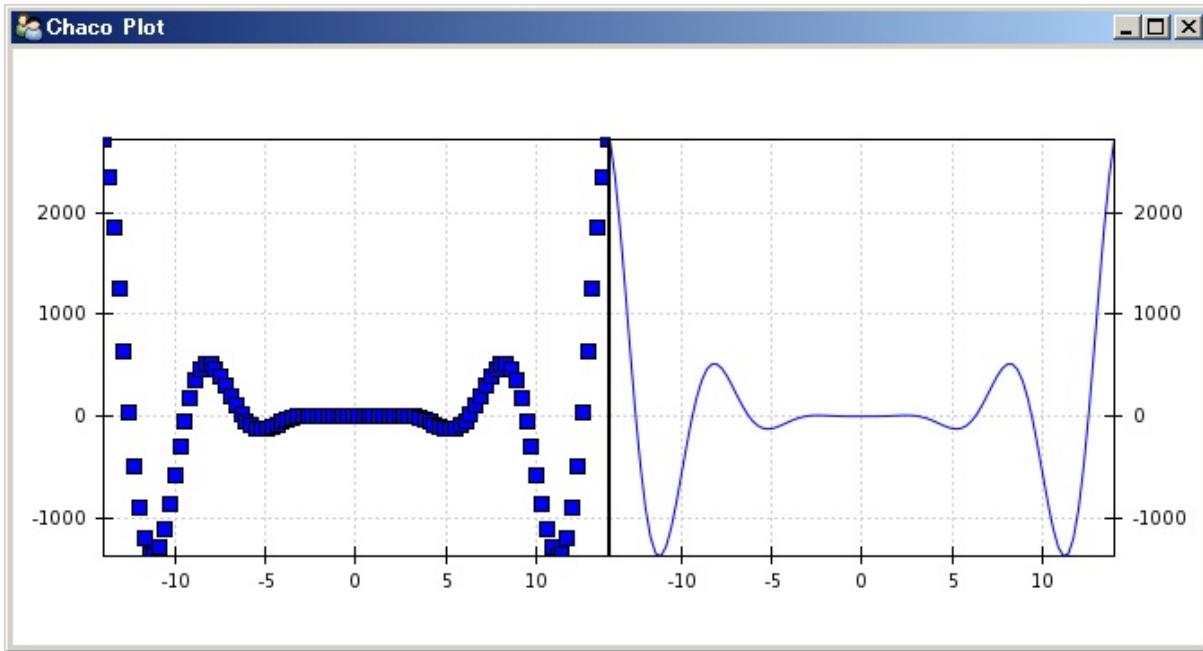


ContainerExampleplotPlotHPlotContainer

```
__init__ Plots scatter line HPlotContainer container plot  
container
```

```
__init__:
```

```
scatter.padding_right = 0  
line.padding_left = 0  
line.y_axis.orientation = "right"
```



padding

Plot(scatter)0(line)0 Y

ChacoTraitstraitTraitsTraitsUI

```
1  from enthought.traits.api import HasTraits, Instance, Int,
2  from enthought.traits.ui.api import View, Group, Item
3  from enthought.enable.component_editor import ComponentEditor
4  from enthought.chaco.api import marker_trait, Plot, ArrayPlot
5  from numpy import linspace, sin
6
7  class ScatterPlotTraits(HasTraits):
8
9      plot = Instance(Plot)
10     color = Color("blue")
11     marker = marker_trait
12     marker_size = Int(4)
13
14     traits_view = View(
15         Group(Item('color', label="Color"),
16               Item('marker', label="Marker"),
17               Item('marker_size', label="Size"),
```

```
18         Item('plot', editor=ComponentEditor(), show_]
19             orientation = "vertical"),
20             width=800, height=600, resizable=True, title=
21
22     def __init__(self):
23         super(ScatterPlotTraits, self).__init__()
24         x = linspace(-14, 14, 100)
25         y = sin(x) * x**3
26         plotdata = ArrayPlotData(x = x, y = y)
27         plot = Plot(plotdata)
28
29         self.renderer = plot.plot(("x", "y"), type="scatter")
30         self.plot = plot
31
32     def _color_changed(self):
33         self.renderer.color = self.color
34
35     def _marker_changed(self):
36         self.renderer.marker = self.marker
37
38     def _marker_size_changed(self):
39         self.renderer.marker_size = self.marker_size
40
41 if __name__ == "__main__":
42     ScatterPlotTraits().configure_traits()
```

traitflash

```
ScatterPlotTraits4traitplotcolormarkermarker_sizecolor  
Colormarker_sizeIntmarkerChacosscatter_makers.pyTrait
```

```
14ScatterPlotTraitstrait_traits_view4Item4traittrait3  
_color_changed_marker_changed_marker_size_changedtrait  
footrait_foo_changedtraittraittrait  
(render)
```

```
renderplot.plotplot.plotsplotrenderplot.plots.values()  
[0]plot.plots["plot0"]"plot0"
```

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

TVTK-

VTK (<http://www.vtk.org/>) C++PythonAPIC++

Pythonenthought.comTVTKVTKPythonAPITraitnumpy

TVTKVTKVTKTVKTVTK

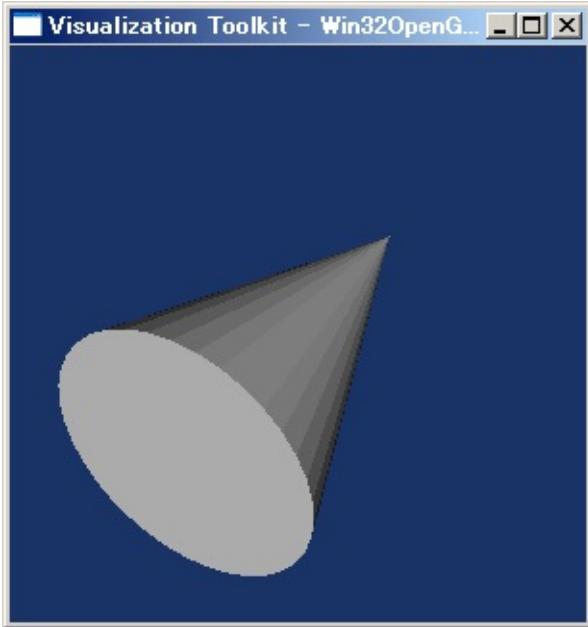
TVTK

a

:

```
# -*- coding: utf-8 -*-
from enthought.tvtk.api import tvtk

# (36)
cs = tvtk.ConeSource(height=3.0, radius=1.0, resolution=36)
# PolyDataMapper
m = tvtk.PolyDataMapper(input = cs.output)
# Actor
a = tvtk.Actor(mapper=m)
# RendererActor
ren = tvtk.Renderer(background=(0.1, 0.2, 0.4))
ren.add_actor(a)
# RenderWindow()Render
rw = tvtk.RenderWindow(size=(300,300))
rw.add_renderer(ren)
# RenderWindowInteractor
rwi = tvtk.RenderWindowInteractor(render_window=rw)
#
rwi.initialize()
rwi.start()
```



 TVTK

tvtk.apitvtktvtkvtk:

```
>>> from enthought.tvtk.api import tvtk
```

ConeSourcecsVTKtvktvkttraittrait(

36)print_traitstrait:

```
>>> cs = tvtk.ConeSource(height=3.0, radius=1.0, resolution=36)
>>> cs.print_traits()
...
angle:                                18.43494882292201
...
center:                               array([ 0.,  0.,  0.])
class_name:                            'vtkConeSource'
...
direction:                             array([ 1.,  0.,  0.])
...
height:                                3.0
...
radius:                                 1.0
...
resolution:                            36
...
```

VTKVTKVTK(Pipeline)

- (Visualization Pipeline)
- (Graphics Pipeline)

(Mapper)

ConeSource(PolyData)PolyDataPolyDataMapper

ConSourcePolyDataMapper

ActorRendererRenderWindow

- **Actor** : (mapper)
- **Renderer** : ActorActor
- **RenderWindow** : RenderRender
- **RenderWindowInteractor** : Actor(Camera)

PolyData

PolyData

- **points** : Points
- **verts** : CellArray points verts points
- **line** : CellArray points
- **polys** : CellArray poi

ivtk

tvtkTVTKivtkivtk

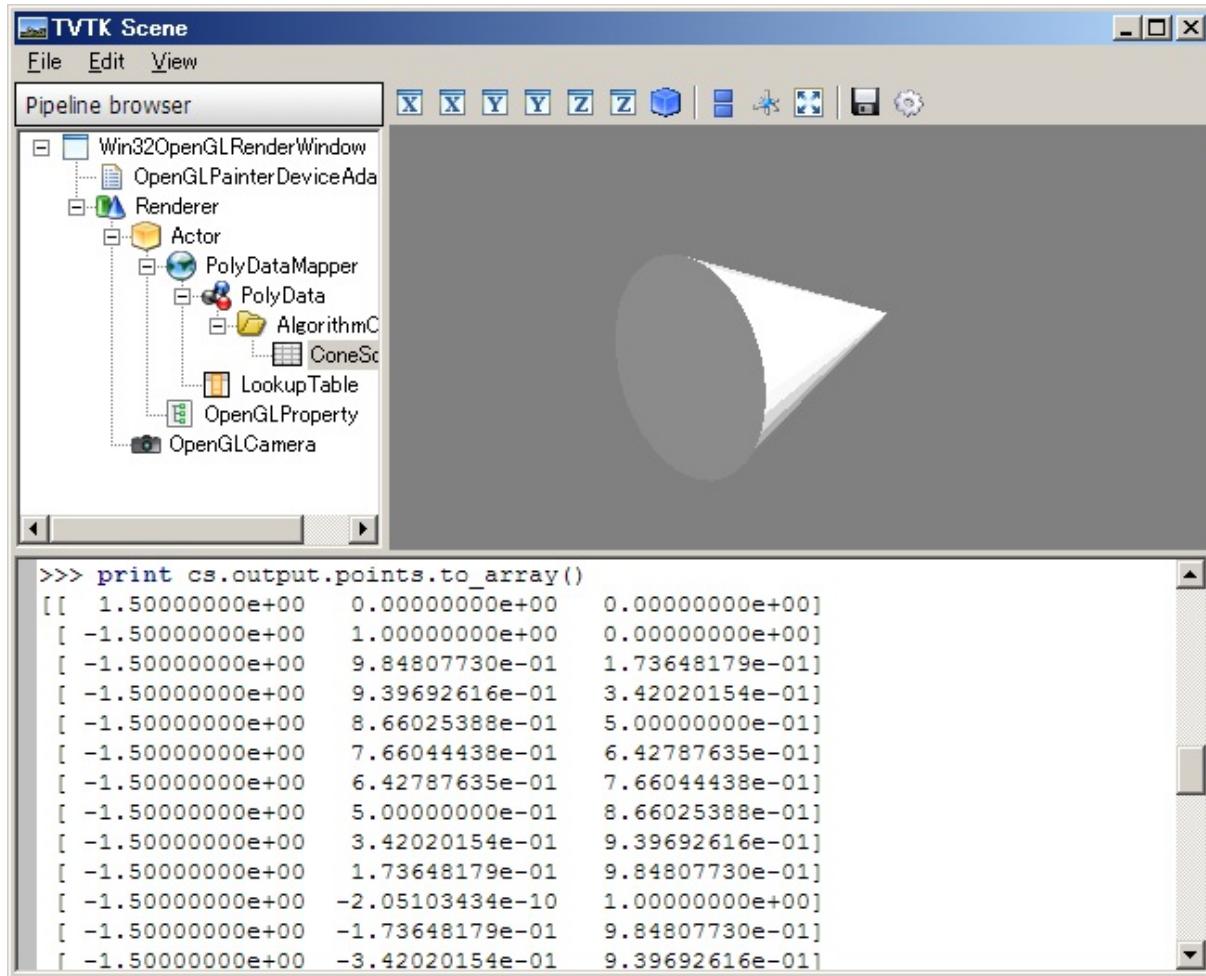
```
# -*- coding: utf-8 -*-
```

```
from enthought.tvtk.api import tvtk

# ivtk
from enthought.tvtk.tools import ivtk
from enthought.pyface.api import GUI

cs = tvtk.ConeSource(height=3.0, radius=1.0, resolution=36)
m = tvtk.PolyDataMapper(input = cs.output)
a = tvtk.Actor(mapper=m)

# GUICrust (Python shell) ivtk
gui = GUI()
window = ivtk.IVTKWithCrustAndBrowser(size=(800,600))
window.open()
window.scene.add_actor( a ) # actor
gui.start_event_loop()
#window.scene.reset_zoom()
```



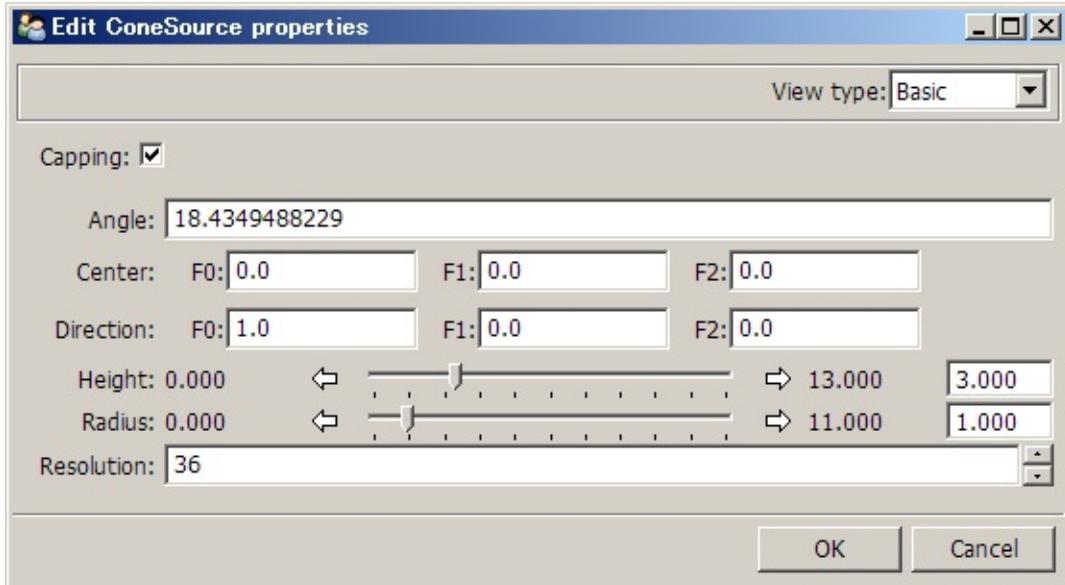
Python Shell

ivtk

- :
- : (ConeSource)RenderWindow
- **Python Shell** : Python ShellConeSourcePolyDatapoints

HasTraitstraitConeSourceConeSource

Note: ConeSource



ConeSource

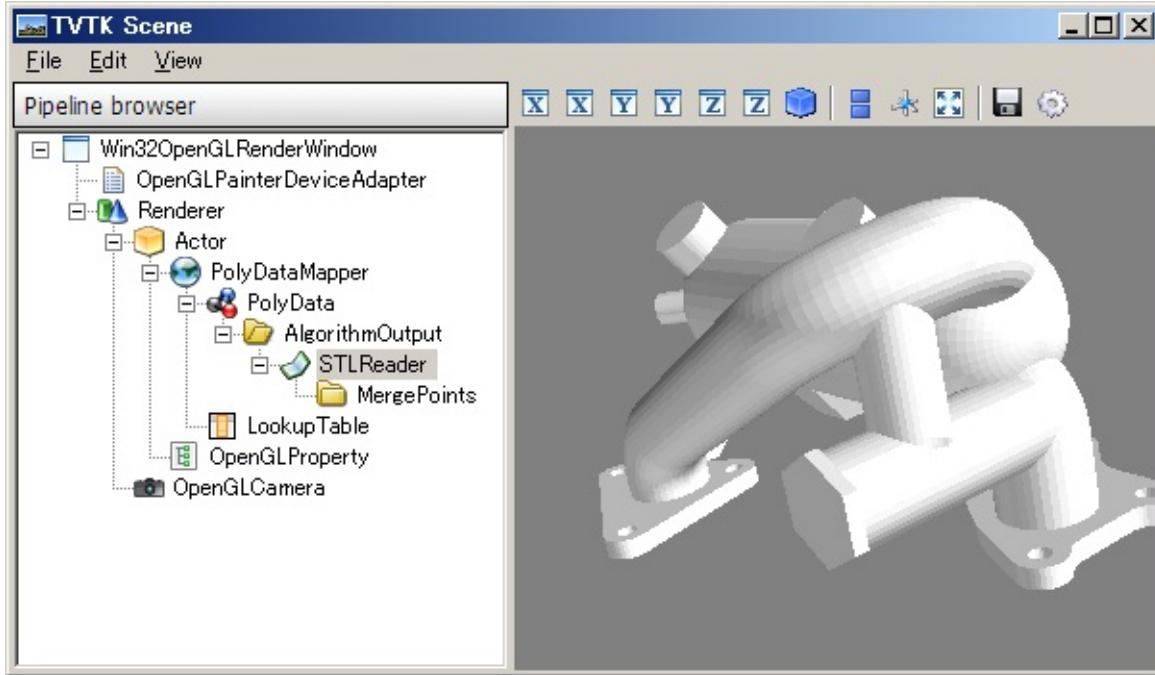
heightRadiusresolution

VTKVTKReaderVTK42400-IDGH.stl

```
# -*- coding: utf-8 -*-
from enthought.tvtk.api import tvtk
from enthought.tvtk.tools import ivtk
from enthought.pyface.api import GUI

part =tvtk.STLReader(file_name = "42400-IDGH.stl")
part_mapper = tvtk.PolyDataMapper( input = part.output )
part_actor = tvtk.Actor( mapper = part_mapper )

gui = GUI()
window = ivtk.IVTKWithBrowser(size=(800,600))
window.open()
window.scene.add_actor( part_actor )
gui.start_event_loop()
```



3D

PolyDataMapperConeSourceSTLReaderSTLReaderSTL

PolyDataPolyDataMapperivtk.IVTKWithBrowserPython

Shellivtk

STL

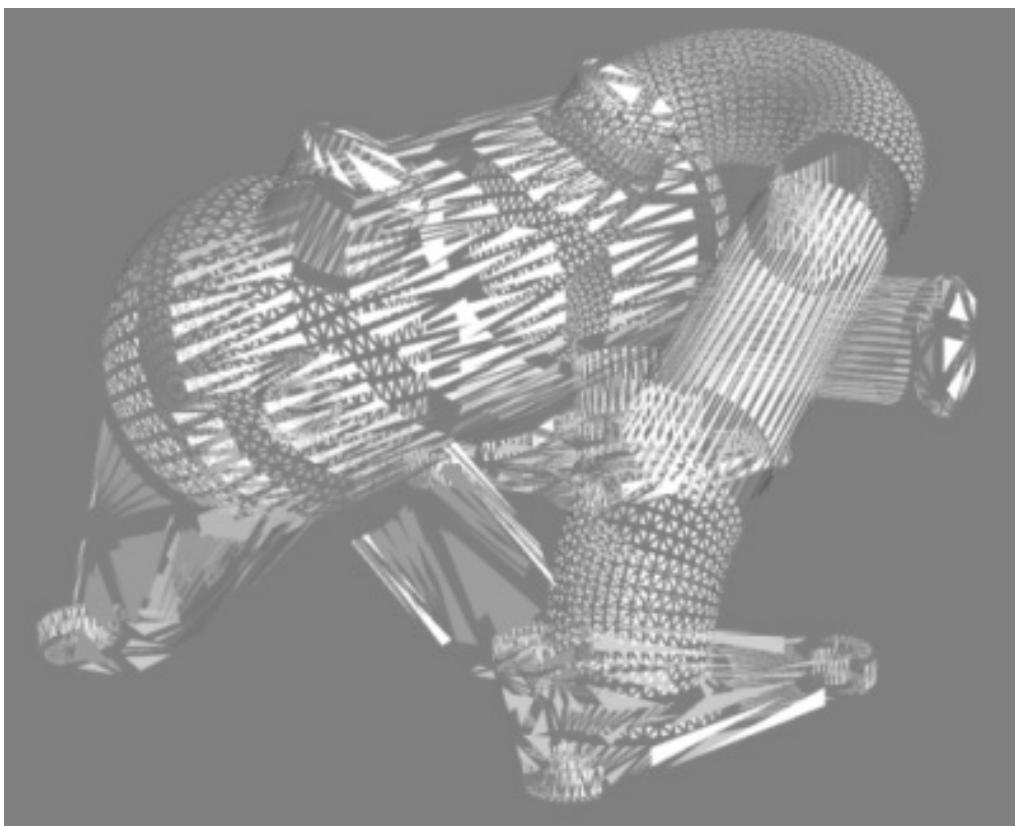
STLstereo-lithography3D SystemsCAD/CAM42400-

IDGH.stlVTKVTKVTK

mapperSTLReaderPolyDataMapperShrinkPolyData

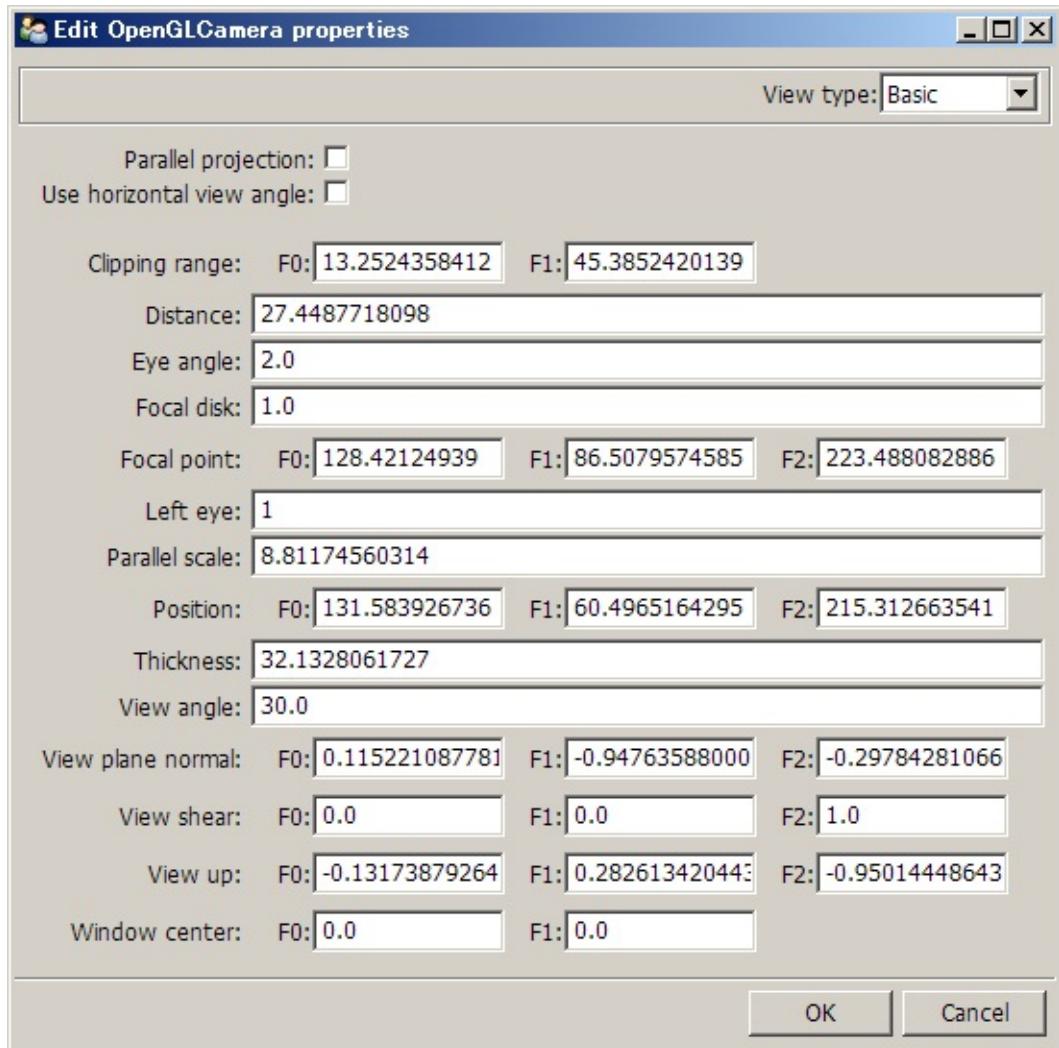
```
part = tvtk.STLReader(file_name = "42400-IDGH.stl")
shrink = tvtk.ShrinkPolyData(input = part.output, shrink_factor =
part_mapper = tvtk.PolyDataMapper( input = shrink.output )
```

ShrinkPolyDataPolyDataPolyData()



 ShrinkPolyData

ivtk3DOpenGLCamera



3D

```
>>> camera = window.scene.renderer.active_camera
```

```
>>> camera.clipping_range  
array([ 20.46912341,  51.21854284])  
>>> camera.view_up = 0,1,0
```

- **clipping_plane** :
- **position** :
- **focal_point** :
- **view_up** :
- **parallel_projection** : True3D2D

..TODO * **azimuth** :

- **elevation** :

view_up

ivtk

```
>>> camera = window.scene.renderer.active_camera
>>> light = tvtk.Light(color=(1,0,0))
>>> light.position=camera.position
>>> light.focal_point=camera.focal_point
>>> window.scene.renderer.add_light(light)
```

..TODO focal_point light positional True cone_angle

180

3D Props

3D prop prop Prop3D Actor3D prop Prop3D

TVTK

PythonVTK:

```
import vtk

# Source object .
cone = vtk.vtkConeSource( )
cone.SetHeight( 3.0 )
cone.SetRadius( 1.0 )
cone.SetResolution(10)
# The mapper .
coneMapper = vtk.vtkPolyDataMapper( )
coneMapper.SetInput( cone.GetOutput( ) )
# The actor .
coneActor = vtk.vtkActor( )
coneActor.SetMapper ( coneMapper )
# Set it to render in wireframe
coneActor.GetProperty( ).SetRepresentationToWireframe( )

# Renderer and render window .
ren1 = vtk.vtkRenderer( )
ren1.AddActor( coneActor )
ren1.SetBackground( 0.1 , 0.2 , 0.4 )
renWin = vtk.vtkRenderWindow( )
renWin.AddRenderer( ren1 )
renWin.SetSize(300 , 300)

# On screen interaction .
iren = vtk.vtkRenderWindowInteractor( )
iren.SetRenderWindow( renWin )
iren.Initialize( )
iren.Start( )
```

C++C++VTK APIVTK-PythonC++PythonVTK-
PythonPythonEnthought.comTVTKVTK:

- Trait
- Pickle
- APIPython
- numpyPython

- mlab APIvtk
- tvtkEnvisage

TVTK

TVTK:

```
from enthought.tvtk.api import tvtk

cone = tvtk.ConeSource( height=3.0, radius=1.0, resolution=10 )
cone_mapper = tvtk.PolyDataMapper( input = cone.output )
cone_actor = tvtk.Actor( mapper=cone_mapper )
cone_actor.property.representation = "w"

ren1 = tvtk.Renderer()
ren1.add_actor( cone_actor )
ren1.background = 0.1, 0.2, 0.4
ren_win = tvtk.RenderWindow()
ren_win.add_renderer( ren1 )
ren_win.size = 300, 300

iren = tvtk.RenderWindowInteractor( render_window = ren_win )
iren.initialize()
iren.start()
```

VTKTVTK

- tvtk from enthought.tvtk.api import tvtk
- tvtkVTK"vtk""vtk"vtk3DSImporterPythonvtk:
: ThreeDSImporter
- tvtkEnthoughtVTKAddItemtvtkadd_item
- VTKtvtktraitm.input = cs.outputVTK
m.SetInput(cs.GetOutput())p.representation = 'w'VTK
p.SetRepresentationToWireframe()VTKSet*, Get*TVTKTrait
Trait
- trait

tvtkVTKtvtkVTKVTKtvtktvtkVTKVTK

`tvtk.to_tvtk(p)pVTK`

Trait

`tvtktraits.HasStrictTraitsHasStrictTraits`

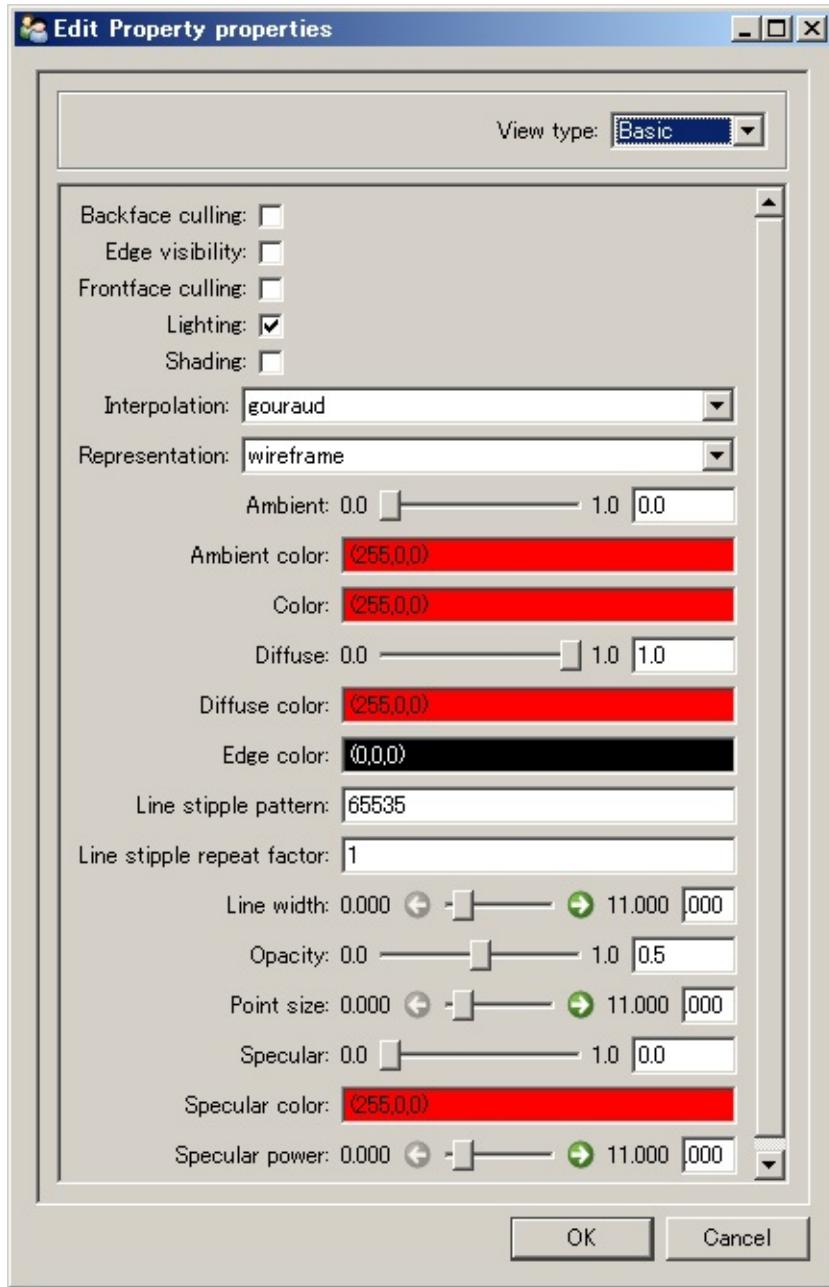
`VTKtvtktraittrait:`

- `settrait`

```
>>> p = tvtk.Property()  
>>> p.set(opacity=0.5, color=(1,0,0), representation="w")
```

- `edit_traitsconfigure_traitstraitVTKVTKtrait`

```
>>> p.edit_traits()
```



TVTK

- `tvtk.to_vtk(p).vtkTVK`

```
>>> print p.representation
wireframe
>>> p_vtk = tvtk.to_vtk(p)
>>> p_vtk.SetRepresentationToSurface()
>>> print p.representation
surface
```

(Pickling)

tvtktvtk

```
>>> import cPickle  
>>> p = tvtk.Property()  
>>> p.representation="w"  
>>> s = cPickle.dumps(p)  
>>> del p  
>>> q = cPickle.loads(s)  
>>> q.representation  
'wireframe'
```

VTK

pickle.load:

```
>>> p = tvtk.Property()  
>>> p.interpolation = "flat"  
>>> d = p.__getstate__()  
>>> del p  
>>> q = tvtk.Property()  
>>> q.interpolation  
'gouraud'  
>>> q.__setstate__(d)  
>>> q.interpolation  
'flat'
```

tvtk.CollectionPython:

```
>>> ac = tvtk.ActorCollection()  
>>> len(ac)  
0  
>>> ac.append(tvtk.Actor())  
>>> ac.append(tvtk.Actor())  
>>> len(ac)  
2  
>>> for a in ac:  
...     print a
```

```
....  
vtkOpenGLActor (06A99EB8)  
.....  
vtkOpenGLActor (069C4270)  
.....  
>>> del ac[0]  
>>> len(ac)  
1
```

ActorCollectionPythonlen, appendforVTKvtk:

```
>>> ac = vtk.vtkActorCollection()  
>>> ac.GetNumberOfItems()  
0  
>>> ac.AddItem(vtk.vtkActor())  
>>> ac.AddItem(vtk.vtkActor())  
>>> ac.GetNumberOfItems()  
2  
>>> ac.InitTraversal()  
>>> for i in range(ac.GetNumberOfItems()):  
...     print ac.GetNextItem()  
....  
vtkOpenGLActor (05E0A750)  
.....  
vtkOpenGLActor (05E0A8C0)  
.....  
>>> ac.RemoveItem(0)  
>>> ac.GetNumberOfItems()  
1
```

DataArrayPython __getitem__, __setitem__, __repr__,
append, extendnumpypython(from_array)DataArray
numpyPointsIdList

```
>>> pts = tvtk.Points()  
>>> p_array = np.eye(3)  
>>> p_array  
array([[ 1.,  0.,  0.],  
       [ 0.,  1.,  0.],  
       [ 0.,  0.,  1.]])  
>>> pts.from_array(p_array)
```

```

>>> pts.print_traits()
_in_set:          0
vtk_obj:         <vtkCommonPython.vtkPoints vtkobject at 0x142D4F60>
actual_memory_size: 1L
bounds:           (0.0, 1.0, 0.0, 1.0, 0.0, 1.0)
class_name:       'vtkPoints'
data:             [(1.0, 0.0, 0.0), (0.0, 1.0, 0.0), (0.0,
data_type:        'double'
...
number_of_points: 3
reference_count:  1
>>> pts.to_array()
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])

```

tvtkdataArray, Points, IdListCellArray:

```

>>> points = np.array([[0,0,0],[1,0,0],[0,1,0],[0,0,1]], 'f')
>>> triangles = np.array([[0,1,3],[0,3,2],[1,2,3],[0,2,1]])
>>> values = np.array([1.1, 1.2, 2.1, 2.2])
>>> mesh = tvtk.PolyData(points=points, polys=triangles)
>>> mesh.point_data.scalars = values
>>> mesh.points
[(0.0, 0.0, 0.0), (1.0, 0.0, 0.0), (0.0, 1.0, 0.0), (0.0, 0.0, 1.0)]
>>> mesh.polys
<tvtk_classes.cell_array.CellArray object at 0x142D4F60>
>>> mesh.polys.to_array()
array([3, 0, 1, 3, 3, 0, 3, 2, 3, 1, 2, 3, 0, 2, 1])
>>> mesh.point_data.scalars
[1.1000000000000001, 1.2, 2.1000000000000001, 2.2000000000000002]

```

CellArray(meshpolys): array([3, 0, 1, 3, 3, 0, 3, 2, 3, 1, 2, 3, 3, 0, 2, 1])[Cell, Cell..., Cell, Cell...]

CellArray(Cell)Cell

TVTK

```

from enthought.tvtk.api import tvtktvktvtkVTK
TVTK

```

tvtk

- tvtkTVTK
- TVTK
- TVTK

tvtktvtk_classes.ziptvtktvtk_helper.pyTVTKtvtk

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

Mayavi-

VTK 3DPythonTVTKVTKParaView

VTKDesigner2Mayavi2

Mayavi2PythonPythonPythonAPImlab

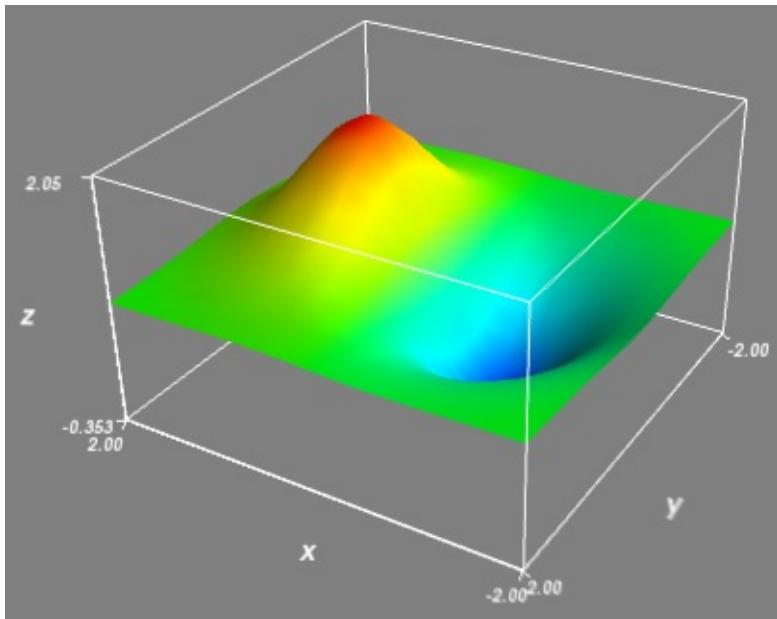
mlab

ChacoshellmatplotlibpylabmayavimlabmlabIPython

```
import numpy as np
from enthought.mayavi import mlab

x, y = np.ogrid[-2:2:20j, -2:2:20j]
z = x * np.exp( - x**2 - y**2)

pl = mlab.surf(x, y, z, warp_scale="auto")
mlab.axes(xlabel='x', ylabel='y', zlabel='z')
mlab.outline(pl)
```



 Mayavi3D

mlab

```
from enthought.mayavi import mlab
```

```
mlab.surfsurfx,y,zx,yogridshapen*11*nzn*n
```

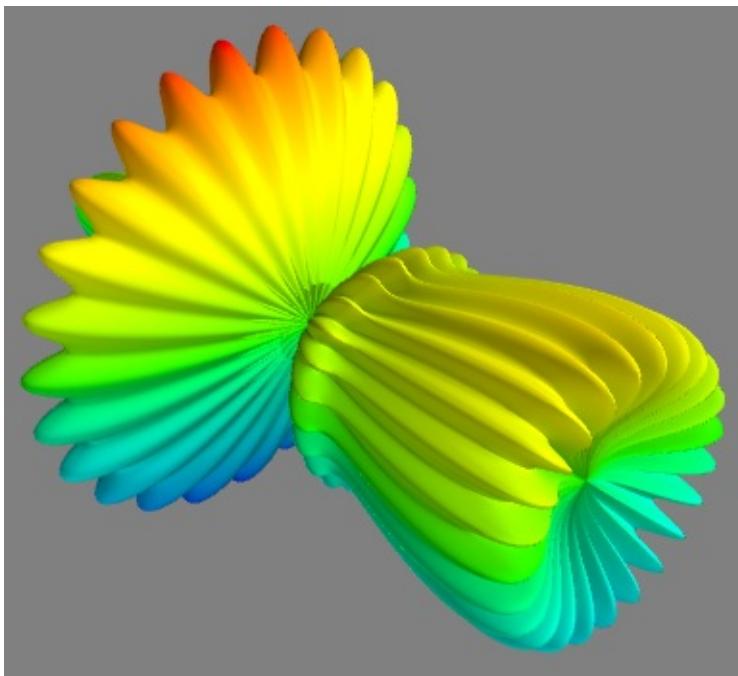
`mlab.axesmlab.outline`

`surfX-Ymeshmesh`

```
# -*- coding: utf-8 -*-
from numpy import *
from enthought.mayavi import mlab

# Create the data.
dphi, dtheta = pi/20.0, pi/20.0
[phi,theta] = mgrid[0:pi+dphi*1.5:dphi,0:2*pi+dtheta*1.5:dtheta]
m0 = 4; m1 = 3; m2 = 2; m3 = 3; m4 = 6; m5 = 2; m6 = 6; m7 = 4;
r = sin(m0*phi)**m1 + cos(m2*phi)**m3 + sin(m4*theta)**m5 + cos(m6*theta)**m7
x = r*sin(phi)*cos(theta)
y = r*cos(phi)
z = r*sin(phi)*sin(theta)

# View it.
s = mlab.mesh(x, y, z, representation="wireframe", line_width=1.0
mlab.show()
```



`mesh3D`

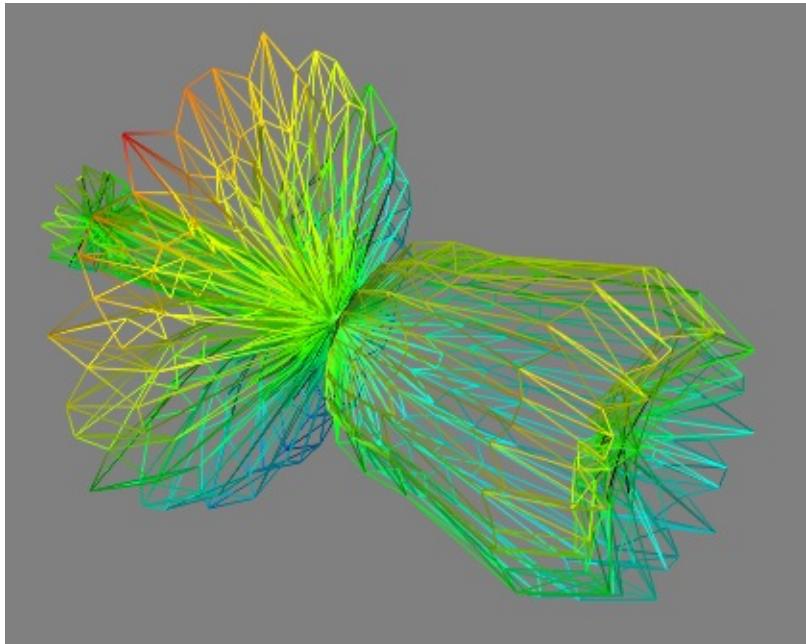
`meshsurf(x, y, z)`

X-Y-Z

representationmesh

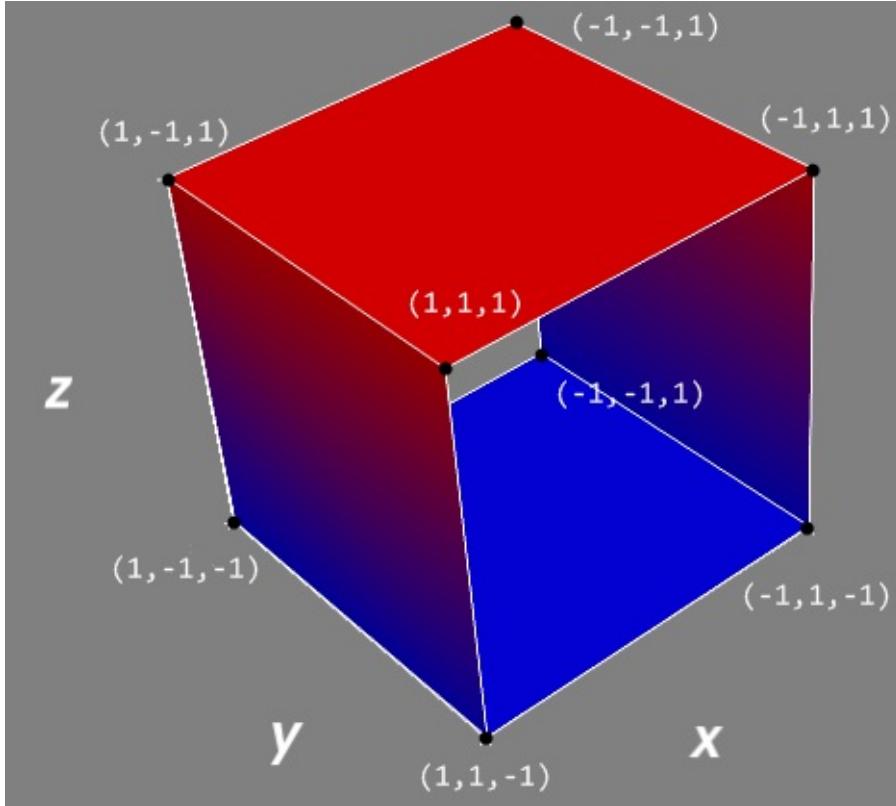
- **surface** :
- **wireframe** : $d\phi, d\theta \pi / 20$

```
s = mlab.mesh(x, y, z, representation="wireframe", line_width
```



mesh

mesh



X,y,z

```
x = [[-1, 1, 1, -1, -1],  
[-1, 1, 1, -1, -1]]
```

```
y = [[-1, -1, -1, -1, -1],  
[1, 1, 1, 1, 1]]
```

```
z = [[1, 1, -1, -1, 1],  
[1, 1, -1, -1, 1]]
```

X, y, z

```
[  
    [(-1, -1, 1), (1, -1, 1), (1, -1, -1), (-1, -1, -1), (-1, 1, -1),  
     [(-1, 1, 1), (1, 1, 1), (1, 1, -1), (-1, 1, -1), (-1, 1, 1)]  
]
```

```
( -1, -1, 1), (1, -1, 1), (-1, 1, 1), (1, 1, 1)
```

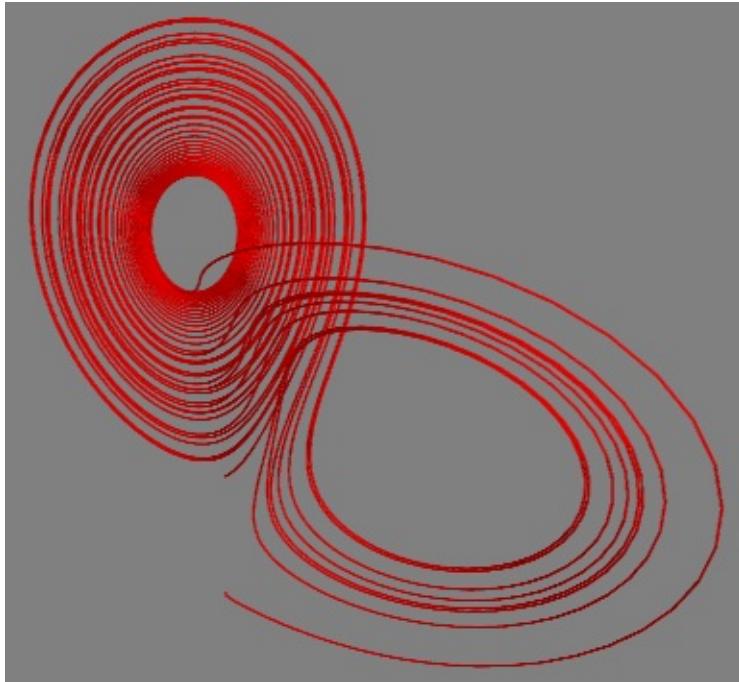
mesh

```
(1, -1, 1), (1, -1, -1), (1, 1, 1), (1, 1, -1)
```

4

mlab

- **points3d, plot3d** : 3x,y,z(points3d)(plot3d)plot3d



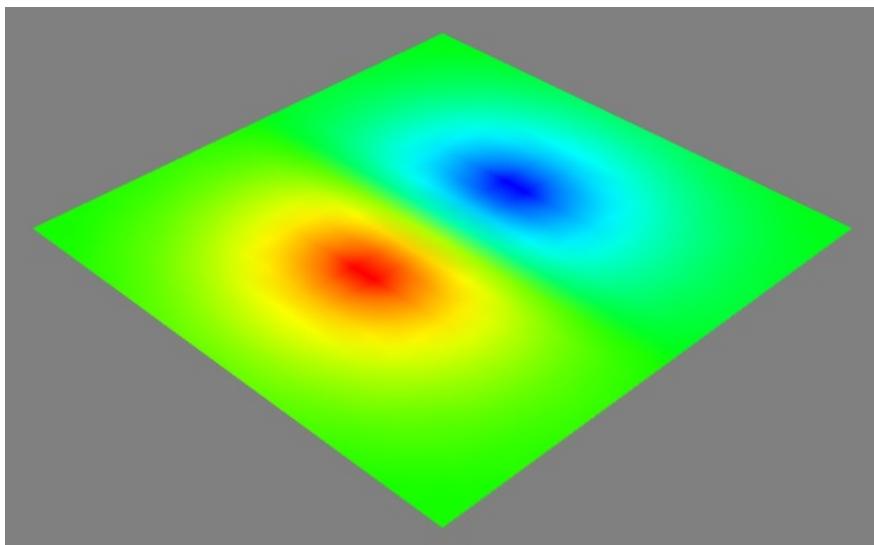
plot3d

```
mlab.plot3d(track1[:,0], track1[:,1], track1[:,2], color=(1,0,0),
```

track1X,Y,Zplot3dtube_radius

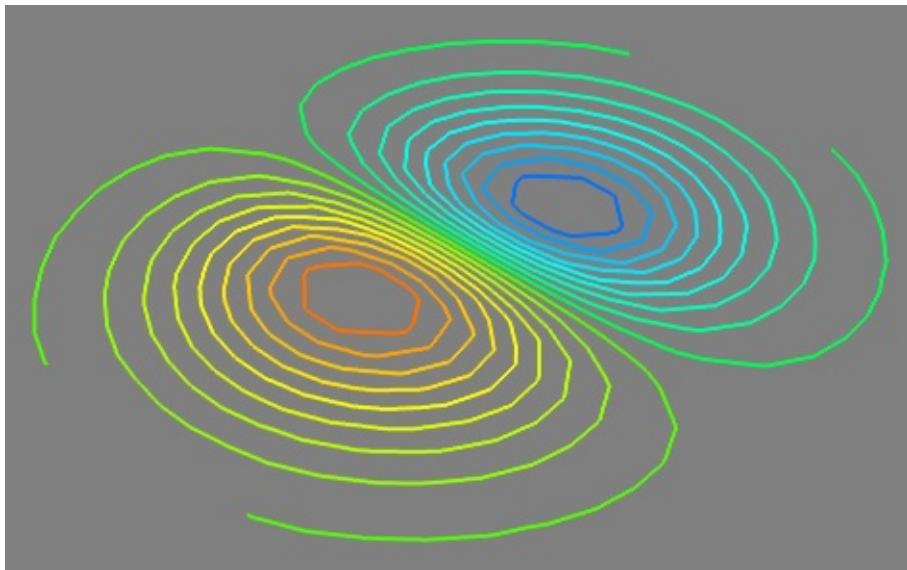
SciPy-

- **imshow, surf, contour_surf** : sXYimshows
surfZcontour_surfimshow(surf)



imshow

contour_surf



contour_surf

Mayavi

Mayavi

MayavitraitsTraitsUI

```
# -*- coding: utf-8 -*-
from enthought.traits.api import *
from enthought.traits.ui.api import *
from enthought.tvtk.pyface.scene_editor import SceneEditor
from enthought.mayavi.tools.mlab_scene_model import MlabSceneMode
from enthought.mayavi.core.ui.mayavi_scene import MayaviScene

class DemoApp(HasTraits):
    plotbutton = Button(u"" )
    scene = Instance(MlabSceneModel, ()) # mayavi

    view = View(
        VGroup(
            Item(name='scene',
                  editor=SceneEditor(scene_class=MayaviScene), # ma
                  resizable=True,
                  height=250,
                  width=400
                ),
            'plotbutton',
            show_labels=False
        ),
        title=u"TraitsUIMayavi"
    )

    def _plotbutton_fired(self):
        self.plot()

    def plot(self):
        g = self.scene.mlab.test_mesh()

app = DemoApp()
app.configure_traits()
```

traitstraitstraits.uiSceneEditorMlabSceneModelMayaviScene

MlabSceneModelmlab(Model):

```
scene = Instance(MlabSceneModel, ())
```

```
traitssceneMlabSceneModel(View)scene
```

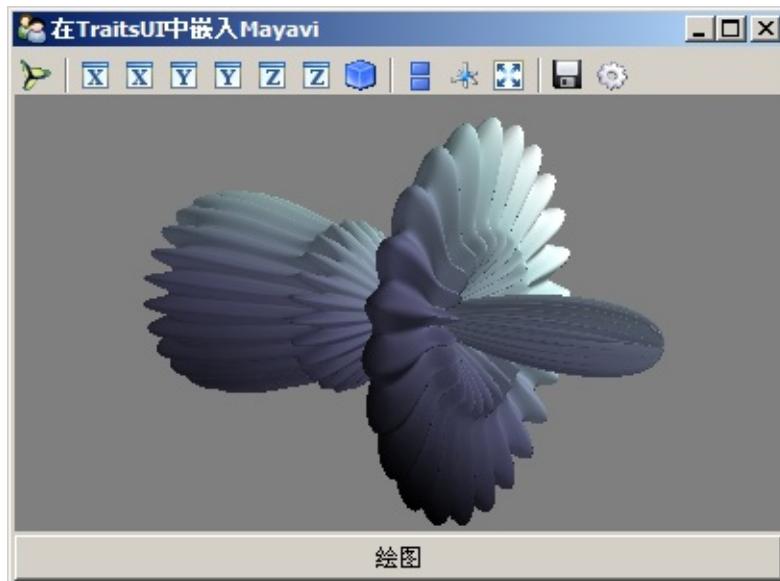
```
Item(name='scene',
      editor=SceneEditor(scene_class=MayaviScene), # mayavi
      resizable=True,
      height=250,
      width=400
    )
```

```
SceneEditorscene_classMayaviScene
```

```
plotbutton_plotbutton_firedplotplot
```

```
g = self.scene.mlab.test_mesh()
```

```
scene.mlabmlabtest_meshscene
```



 MayaviTraitsUI

```
x,y,zf(x,y,z)x*x+y*y+z*z
```

```
plotplotmgridx0, x1, y0, y1, z0, z1, points, functiontraits
```

```
#  
x, y, z = mgrid[  
    self.x0:self.x1:1j*self.points,  
    self.y0:self.y1:1j*self.points,  
    self.z0:self.z1:1j*self.points]  
scalars = eval(self.function) #
```

```
self.scene.mlab.clf() #
```

scene.mlabaxes, contour3d, pipeline.scalar_cut_plane

```
#  
g = self.scene.mlab.contour3d(x, y, z, scalars, contours=8, trans|  
g.contour.auto_contours = self.autocontour  
self.scene.mlab.axes() #  
  
# X-Y  
s = self.scene.mlab.pipeline.scalar_cut_plane(g)  
cutpoint = (self.x0+self.x1)/2, (self.y0+self.y1)/2, (self.z0+sel|  
s.implicit_plane.normal = (0,0,1) # x cut  
s.implicit_plane.origin = cutpoint
```

v0v1

```
self.g = g  
self.scalars = scalars  
#  
self.v0 = np.min(scalars)  
self.v1 = np.max(scalars)
```

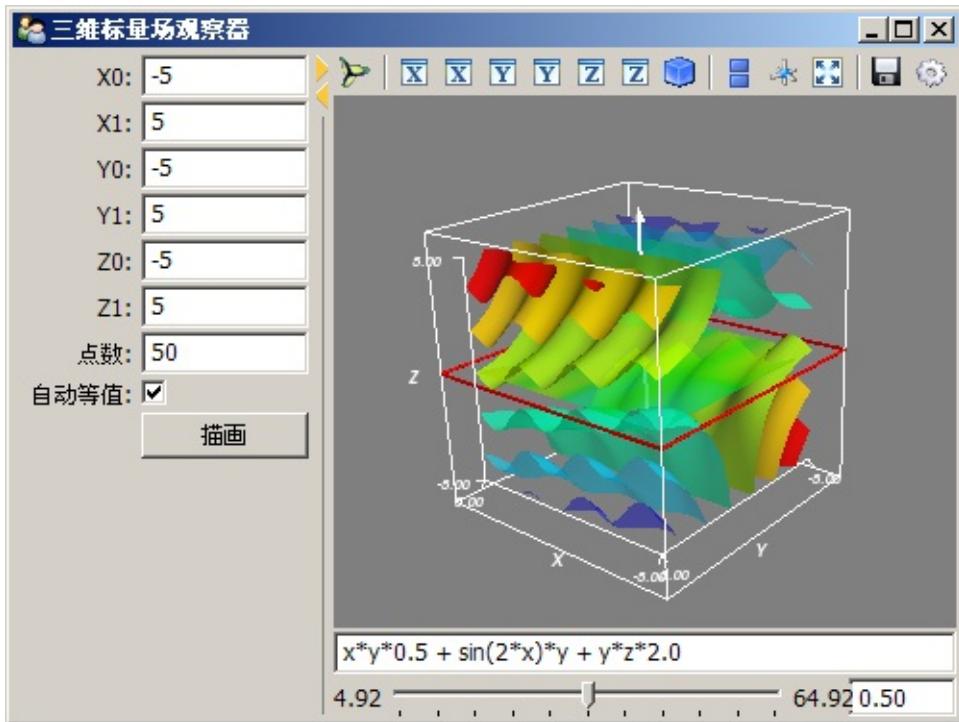
"""(autocontour)

```
self.g.contour.auto_contours = self.autocontour
```

(contour):

```
if not self.g.contour.auto_contours:  
    self.g.contour.contours = [self.contour]
```

traits TraitsUI *TraitsUI-*



█ $x*y*0.5 + \sin(2*x)*y + y*z*2.0$

Python »

← (Alt+X) | |

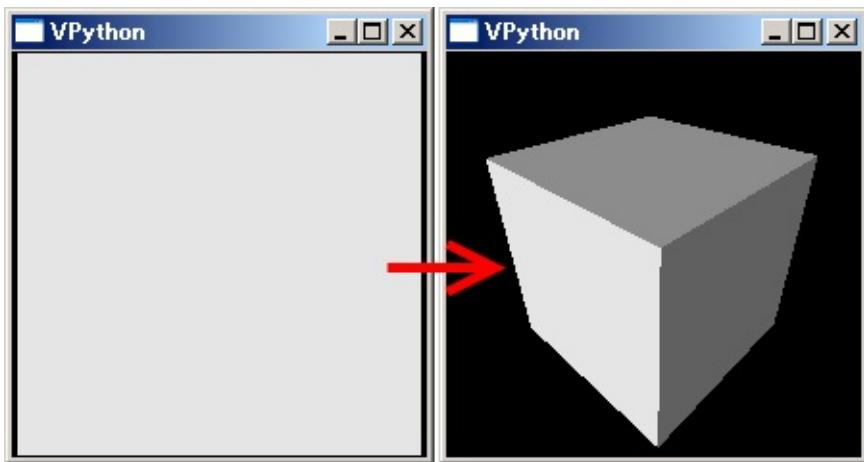
Python »

← (Alt+X) | |

Visual-3D

Visual Python3D3DTVTK3DTVTK3DVisual3D

```
from visual import *\nbox()
```



 VPython

visualbox()boxVPythonbox

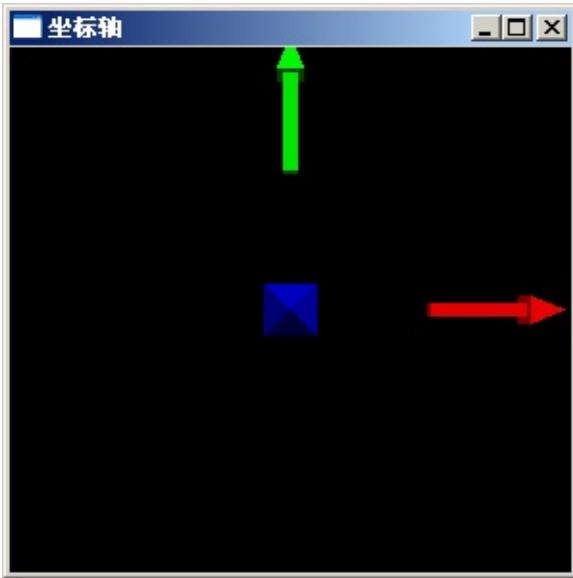
- 3D 0, 0, 0
- 1, 1, 1
-

Z

3D

box()

```
# -*- coding: utf-8 -*-
from visual import *
display(title=u" ".encode("gb2312"), width=300, height=300)
arrow(pos=(1,0,0), axis=(1,0,0), color=(1,0,0))
arrow(pos=(0,1,0), axis=(0,1,0), color=(0,1,0))
arrow(pos=(0,0,1), axis=(0,0,1), color=(0,0,1))
```



VPythonX,Y,Z

display()Windowsunicodegb2312

3arrow()

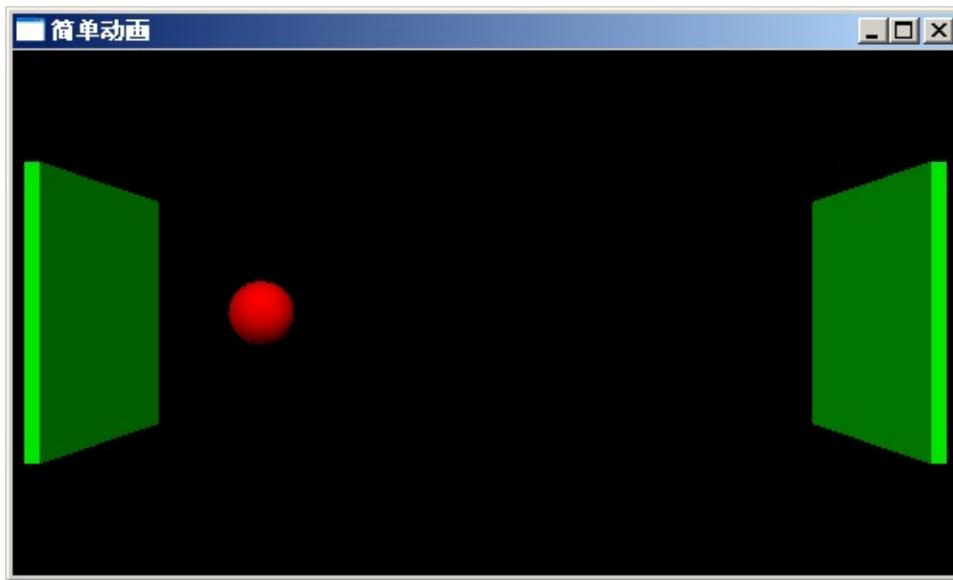
- pos(1,0,0), (0,1,0), (0,0,1)3
- axis3D1
- color01

-
- X
- y
- z

zz

visual3D

```
1 # -*- coding: utf-8 -*-
2 from visual import *
3
4 display(title=u"      .encode("gb2312"), width=500, height=300)
5
6 ball = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
7 wall_right = box(pos=(6,0,0), size=(0.1, 4, 4), color=color.green)
8 wall_left = box(pos=(-6,0,0), size=(0.1, 4, 4), color=color.green)
9
10 dt = 0.05
11 ball.velocity = vector(6, 0, 0)
12
13 while True:
14     rate(1/dt)
15     ball.pos = ball.pos + ball.velocity*dt
16     if ball.x > wall_right.x-ball.radius or ball.x < wall_left.x:
17         ball.velocity.x *= -1
```



6-8(box)(sphere)sphereradiusboxsize x, y, z axisbox

10dt11ballvelocity3Dvelocitysphereball

13ballposratedt0.0520rate

15ballposdtball16, 17pos

6122

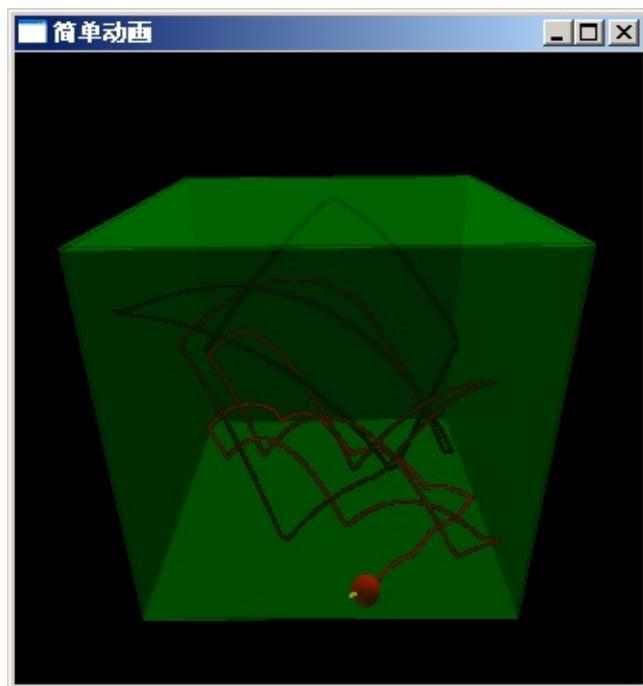
6(Z)

```
1 # -*- coding: utf-8 -*-
2 from visual import *
3
4 display(title=u"      .encode("gb2312"), width=500, height=500)
5
6 # 6
7 ball = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
8 wall_right = box(pos=(6,0,0), size=(0.1, 12, 12), color=color.white)
9 wall_left = box(pos=(-6,0,0), size=(0.1, 12, 12), color=color.white)
10 wall_front = box(pos=(0,-6,0), size=(12, 0.1, 12), color=color.white)
11 wall_back = box(pos=(0,6,0), size=(12, 0.1, 12), color=color.white)
12 wall_bottom = box(pos=(0,0,-6), size=(12, 12, 0.1), color=color.white)
13 wall_top = box(pos=(0,0,6), size=(12, 12, 0.1), color=color.white)
14
15 dt = 0.05
16 g = 9.8 #
17 f = 0.9 # 1.0
18 ball.velocity = vector(8, 6, 12)
19 bv = arrow(pos = ball.pos, axis=ball.velocity*0.2, color=ball.color)
20 ball.trail = curve(color=ball.color)
21 trail_color = 0 #
22
23 while True:
24     rate(1/dt)
25
26     # z
27     ball.velocity.z -= g * dt
28
29     #
30     ball.pos += ball.velocity * dt
31
32     ##
33     ##
34     #
35     if ball.velocity.x > 0 and ball.x >= wall_right.x - ball.radius:
36         ball.x = wall_right.x - ball.radius
37         ball.velocity.x *= -f
38     if ball.velocity.x < 0 and ball.x <= wall_left.x + ball.radius:
39         ball.x = wall_left.x + ball.radius
```

```

40         ball.velocity.x *= -f
41
42     #
43     if ball.velocity.y > 0 and ball.y >= wall_back.y - ball.radius:
44         ball.y = wall_back.y - ball.radius
45         ball.velocity.y *= -f
46     if ball.velocity.y < 0 and ball.y <= wall_front.y + ball.radius:
47         ball.y = wall_front.y + ball.radius
48         ball.velocity.y *= -f
49
50     #
51     if ball.velocity.z > 0 and ball.z >= wall_top.z - ball.radius:
52         ball.z = wall_top.z - ball.radius
53         ball.velocity.z *= -f
54     elif ball.velocity.z < 0 and ball.z <= wall_bottom.z + ball.radius:
55         ball.z = wall_bottom.z + ball.radius
56         ball.velocity.z *= -f
57
58     #
59     bv.pos = ball.pos
60     bv.axis = ball.velocity*0.2
61
62     ball.trail.append( pos = ball.pos, color = (trail_color.r,
63                             trail_color.g, trail_color.b) )
64     trail_color += 1.0/30.0*dt # 30
65     if trail_color > 1.0: trail_color = 1.0

```



8-13opacity0.2opacity=0.0opacity=1.0

19arrow()

```
bv = arrow(pos = ball.pos, axis=ball.velocity*0.1, color=color.ye
```

20cureve()trail

```
ball.trail = curve(color=ball.color)
```

2730

35-56x, y, zx

x()xxx

```
if ball.velocity.x > 0 and ball.x >= wall_right.x - ball.radius:  
    ball.x = wall_right.x - ball.radius  
    ball.velocity.x *= -f  
if ball.velocity.x < 0 and ball.x <= wall_left.x + ball.radius:  
    ball.x = wall_left.x + ball.radius  
    ball.velocity.x *= -f
```

59,606263,6430

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

OpenCV-

OpenCV Intel CSWIG Python PythonOpenCV

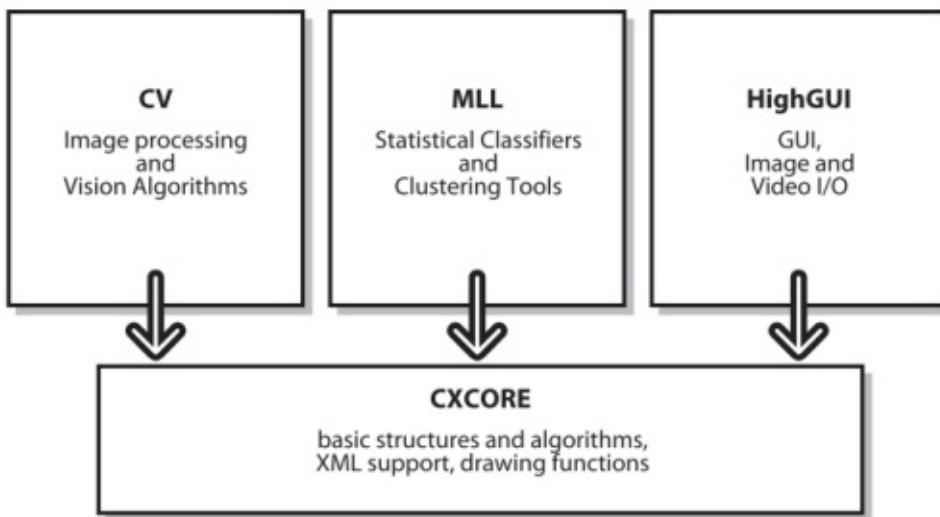
OpenCV Python CAPI Python Python CPython OpenCV API

OpenCV

```
# -*- coding: utf-8 -*-
from opencv.highgui import *
import sys

img = cvLoadImage( sys.argv[1] )
cvNamedWindow( "Example1", CV_WINDOW_AUTOSIZE)
cvShowImage("Example1", img)
cvWaitKey(0)
```

OpenCV54



OpenCV5

- **CV** : Pythonopencv.cv
- **ML** : opencv.ml
- **HighGUI** : GUIopencv.highgui
- **CXCore** : CXCoreopencv
- **CvAux** :

highguiopencvAPIcv

```
from opencv.highgui import *
```

```
cvLoadImageopencv.cv.cvMatcvMatOpenCV()cvMat
```

```
img = cvLoadImage( sys.argv[1] )
```

```
cvNamedWindow"Example1"CV_WINDOW_AUTOSIZE
```

```
cvNamedWindow( "Example1", CV_WINDOW_AUTOSIZE)
```

```
cvShowImageimg"Example1"OpenCVC".()highguiGUI"Example1"
```

```
cvWaitKey0
```

```
cvWaitKey(0)
```

```
IPython!Python!IPython!Python!opencv  
opencv.cv
```

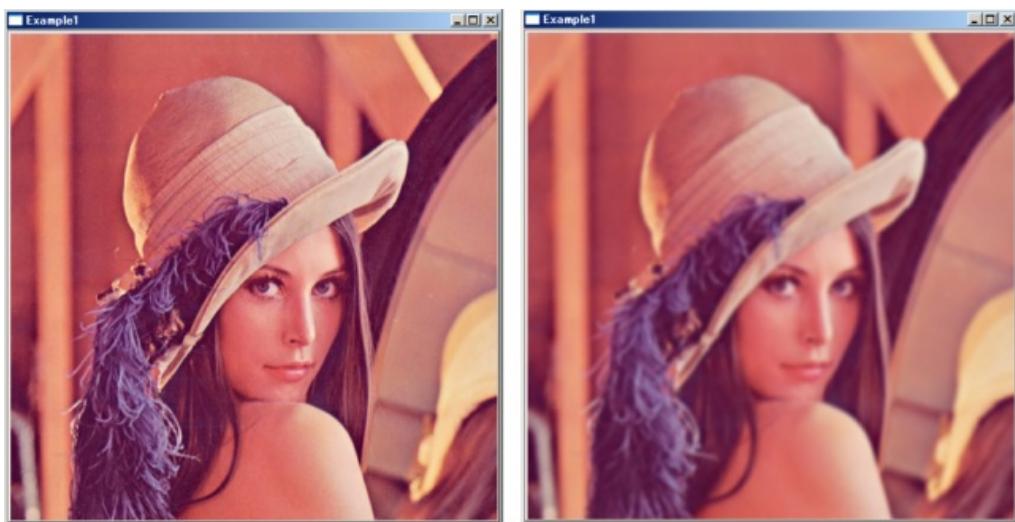
```
>>> from opencv.cv import *
```

```
cvSmoothimgcvSmoothimg
```

```
>>> cvSmooth(img, img, CV_GAUSSIAN, 11)
```

```
cvShowImage
```

```
>>> cvShowImage( "Example1", img )
```



[cvSmooth](#)

[Python »](#)

← (Alt+X) | |

Python »

← (Alt+X) | |

Traits

traits

- Traits
 - Traits
 -
 -
 - Thisself
 -
 - Trait
 -
 -
- Trait
 -

traits.ui

- Trait
 - Trait
 - matplotlib
 - CSV

Python »

← (Alt+X) | |

Traits

Python Traits

1. enthought.traits.api
2. traits
3. HasTraitstraitstrait

23traitstrait

```
from enthought.traits.api import HasTraits, Float

class Person(HasTraits):
    weight = Float(50.0)
```

HasTraitsPersonweighttrait50.0traittrait

```
>>> joe = Person()
>>> joe.weight
50.0
>>> joe.weight = 70.5
>>> joe.weight = 70
>>> joe.weight = "89"
Traceback (most recent call last):
  File "...trait_handlers.py", line 175, in error_value
TraitError: The 'weight' trait of a Person instance must be a float
but a value of '89' <type 'str'> was specified.
```

joePersonweighttrait50.0weightFloattrait

traitfloatattraittraitCasting traitCasting trait

C

```
from enthought.traits.api import HasTraits, CFloat

class Person(HasTraits):
    cweight = CFloat(50.0)
```

```
>>> bill = Person()
```

```
>>> bill.cweight = "90"
>>> bill.cweight
90.0
>>> bill.cweight = "abc"
Traceback (most recent call last)
...

```

CFloattraitcweight"90" "abc" CFloatfloat()
trait

traitstrait

```
from enthought.traits.api import HasTraits, Range

coefficient = Range(-1.0, 1.0, 0.0)

class quadratic(HasTraits):
    c2 = coefficient
    c1 = coefficient
    c0 = coefficient
    x = Range(-100.0, 100.0, 0.0)
```

traitRange()-1.01.00.0coefficient = Range(-1.0,
1.0, 0.0)traitsquadratictraitc0, c1, c2

Traits

TraitsPythontraitHasTraitstraittrait

```
class Person(HasTraits):
    age = Float
```

Personage0.0(Float)traittrait

```
age = Float(10.0)
```

trait .. TODO::

Pythontrait

- **Trait** : trait
- **Trait** : Pyton

Trait	Trait	Python		
Bool	CBool	Boolean	False	bool()
Complex	CComplex	Complex number	0+0j	complex()
Float	CFloat	Floating point number	0.0	float()
Int	CInt	Plain integer	0	int()
Long	CLong	Long integer	0L	int()
Str	CStr	String	"	str()
Unicode	CUnicode	Unicode	u"	unicode()

TraitTrait

```
>>> from enthought.traits.api import HasTraits, Float, CFloat
>>> class Person ( HasTraits ):
```

```
...     weight  = Float
...     cweight = CFloat
>>> bill = Person()
>>> bill.weight  = 180      # OK,  ()
>>> bill.cweight = 180      # OK,
>>> bill.weight  = '180'    # Error,
>>> bill.cweight = '180'    # OK,  float('180')
>>> print bill.cweight
180.0
```

TraitsTrait

- **Any** :

```
Any( [value = None, **metadata] )
```

- **Array** : numpy

```
Array( [dtype = None, shape = None, value = None, typecode =
```

- **Button** :

```
Button( [label = "", image = None, style = "button", orientati
```

- **Callable** :

```
Callable( [value = None, **metadata] )
```

- **CArray** : numpy Array

- **Class** : Python

```
class( [value, **metadata] )
```

- **Code** :

```
Code( [value = "", minlen = 0, maxlen = sys.maxint, regex = "
```

- **Color** :

```
Color( [*args, **metadata] )
```

- **CSet** :

```
CSet( [trait = None, value = None, items = True, **metadata] )
```

- **Constant** :

```
Constant( value*[, ***metadata] )
```

- **Dict** : traitDictStrAny, DictStrBool, ...

```
Dict( [key_trait = None, value_trait = None, value = None, it
```

- **Directory** :

```
Directory( [value = "", auto_set = False, entries = 10, exists
```

- **Disallow** : trait

- **Either** : trait Either(Str, Float)

```
Either( val1*[, *val2, ..., valN, **metadata] )
```

- **Enum** :

```
Enum( values*[, ***metadata] )
```

- **Event** :

```
Event( [trait = None, **metadata] )
```

- **Expression** : Python

```
Expression( [value ="0", **metadata] )
```

- **File :**

```
File( [value = "", filter = None, auto_set = False, entries =
```

- **Font :**

```
Font( [*args, **metadata] )
```

Thisself

traitsThisself()ThisNone

```
class Employee(HasTraits):  
    manager = self
```

EmployeemanagerEmployee

```
>>> e = Employee()  
>>> e.manager  
<__main__.Employee object at 0x05DB72A0>  
>>> e  
<__main__.Employee object at 0x05DB72A0>
```

ThisNone

```
manager = Instance(Employee)
```

managerEmployeeInstance("Employee")

Thisself

```
>>> from enthought.traits.api import HasTraits, This  
>>> class Employee(HasTraits):  
...     manager = This  
...
```

```
>>> class Executive(Employee):
...     pass
...
>>> fred = Employee()
>>> mary = Executive()
>>> fred.manager = mary
>>> mary.manager = fred
```

EnumEnumPythonlist

```
class Items(HasTraits):
    count = Enum(None, 0, 1, 2, 3, "many")
    #
# count = Enum([None, 0, 1, 2, 3, "many"])
```

```
>>> item = Items()
>>> item.count = 2
>>> item.count = "many"
>>> item.count = 5
```

values

```
class Items(HasTraits):
    count_list = List([None, 0, 1, 2, 3, "many"])
    count = Enum(values="count_list")
```

count_listEnumvaluescount_list

```
>>> item = Items()
>>> item.count = 5
Traceback (most recent call last)
#...
>>> item.count_list.append(5)
>>> item.count = 5
>>> item.count
5
```

Trait

TraitHasTraitstraittrait

```
from enthought.traits.api import *

class MetadataTest(HasTraits):
    i = Int(99)
    s = Str("test", desc="a string trait property")

test = MetadataTest()
```

IPythontest

```
>>> test.traits()
{'i': <enthought.traits.traits.CTrait object at 0x05D44EA0>,
'trait_added': <enthought.traits.traits.CTrait object at 0x05D1
's': <enthought.traits.traits.CTrait object at 0x05D44EF8>,
'trait_modified': <enthought.traits.traits.CTrait object at 0x0

>>> test.trait("i")
<enthought.traits.traits.CTrait object at 0x05D44EA0>

>>> test.trait("s").desc
'a string trait property'
```

HasTraitstraitstraittrait

- **trait** : test.i, test.s
- **trait** : traittest.trait("i"), test.trait("s")

traittraittraittesttraitistrait_addedtrait_modifiedHasTraits

- : trait
- : trait

- :

Traits

- **array** : trait
- **default** : trait traittraitdefault

```
>>> test.trait("i").default  
99
```

- **default_kind** : value, list, dict, self, factory, method

```
>>> test.trait("i").default_kind  
'value'
```

- **inner_traits** : traitList, DictListDict
- **trait_type** : traittraitInt

```
>>> test.trait("i").trait_type  
<enthought.traits.trait_types.Int object at 0x05DBD2D0>
```

- **type** : traitconstant, delegate, event, property, trait

```
>>> test.trait("i").type  
'trait'
```

HasTraits

- **desc** : trait
- **editor** : TraitEditortrait
- **label** : trait

- **rich_compare** : traitTrue()Flase
- **trait_value** : traitTraitValuFalseTruetraitTraitValue()
trait
- **transient** : ()traittraitTrue

Python » Traits »

← (Alt+X) | |

Trait

Traitstrait

trait

- : trait
- : @on_trait_change
- : on_trait_change()on_trait_event()trait()

Python » Traits »

← (Alt+X) | |

Trait

trait
traitView
pyQt
wxPython
simple, custom,
text, readonly

traitsUI

trait
traitmatplotlib
CSV

Trait

```
TestStrEditortesttraitStrviewItemtest(editor)
t.configure_traits()traitstest
```

```
from enthought.traits.api import *
from enthought.traits.ui.api import *

class TestStrEditor(HasTraits):
    test = Str
    view = View(Item("test"))

t = TestStrEditor()
t.configure_traits()
```



Traits

traits

traits: site-packages\ Traits-3.2.0-py2.6-win32.egg\enthought\traits,
%traits%

traitsUI: site-packages\ Traits-3.2.0-py2.6-win32.egg\enthought\traits\UI, %ui%

wx: site-packages\ TraitsBackendWX-3.2.0-py2.6.egg\enthought\traitsui\wx, %wx%

Strcreate_editor

```
>>> from enthought.traits.api import *
```

```
>>> s = Str()
>>> ed = s.create_editor()
>>> type(ed)
<class 'enthought.traits.ui.editors.text_editor.ToolkitEditorFacto
>>> ed.get()
{'auto_set': True,
 'custom_editor_class': <class 'enthought.traits.ui.wx.text_edito
 'enabled': True,
 'enter_set': False,
 'evaluate': <enthought.traits.ui.editors.text_editor._Identity o
 'evaluate_name': '',
 'format_func': None,
 'format_str': '',
 'invalid': '',
 'is_grid_cell': False,
 'mapping': {},
 'multi_line': True,
 'password': False,
 'readonly_editor_class': <class 'enthought.traits.ui.wx.text_ed
 'simple_editor_class': <class 'enthought.traits.ui.wx.text_edito
 'text_editor_class': <class 'enthought.traits.ui.wx.text_editor.
 'view': None}
```

create_editor%traits%trait_types.pyBaseStrcreate_editor
text_editor.ToolkitEditorFactory

```
enthought.traits.ui.editors.text_editor.ToolkitEditorFactory
```

%ui%editorstext_editor.pyEditorFactoryEditorFactory
%ui%editor_factory.pyEditorFactoryTraitsEditorFactory

```
class EditorFactory ( HasPrivateTraits ):
    #
    simple_editor_class = Property
    custom_editor_class = Property
    text_editor_class = Property
    readonly_editor_class = Property

    # simple_editor_class
    def simple_editor ( self, ui, object, name, description, parent,
                       factory = self,
                       ui = ui,
```

```

object      = object,
name        = name,
description = description

#
@classmethod
def _get_toolkit_editor(cls, class_name):
    editor_factory_classes = [factory_class for factory_class
                               if issubclass(factory_class, EditorFactory)]
    for index in range(len(editor_factory_classes)):
        try:
            factory_class = editor_factory_classes[index]
            editor_file_name = os.path.basename(
                sys.modules[factory_class.__module__])
            return toolkit_object(':' . join([editor_file_name,
                                              class_name]), True)
        except Exception, e:
            if index == len(editor_factory_classes)-1:
                raise e
    return None

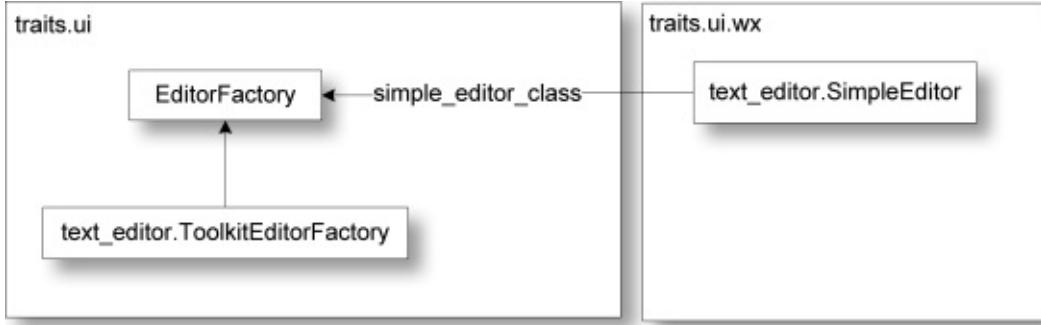
# simple_editor_class get
def _get_simple_editor_class(self):
    try:
        SimpleEditor = self._get_toolkit_editor('SimpleEditor')
    except:
        SimpleEditor = toolkit_object('editor_factory:SimpleEditor')
    return SimpleEditor

```

EditorFactoriesimple_editor_class, custom_editor_class,
text_editor_class, readonly_editor_classedsimple_editor_class
<class 'enthought.traits.ui.wx.text_editor.SimpleEditor'>wx
text_editorSimpleEditor

EditorFactory _get_toolkit_editor _get_toolkit_editorcls
text_editor.ToolkitEditorFactory._get_toolkit_editor()cls
text_editor.ToolkitEditorFactorycls.mroclstoolkit_object
%ui%toolkit.py

traits.uitraits.uiEditorFactorytraits.uiEditorFactory



`traits.uiEditorFactory`

```
wx %wx%text_editor.py text_editor.SimpleEditor
simple, custom, text, readonly() text_editor.SimpleEditor
```

```

class SimpleEditor ( Editor ):

    # Flag for window styles:
    base_style = 0

    # Background color when input is OK:
    ok_color = OKColor

    # Function used to evaluate textual user input:
    evaluate = evaluate_trait

    def init ( self, parent ):
        """ Finishes initializing the editor by creating the underlying
           widget.
        """
        factory      = self.factory
        style        = self.base_style
        self.evaluate = factory.evaluate
        self.sync_value( factory.evaluate_name, 'evaluate', 'from'
                        evaluate )

        if (not factory.multi_line) or factory.password:
            style &= ~wx.TE_MULTILINE

        if factory.password:
            style |= wx.TE_PASSWORD

        multi_line = ((style & wx.TE_MULTILINE) != 0)
        if multi_line:
            self.scrollable = True

```

```
if factory.enter_set and (not multi_line):
    control = wx.TextCtrl( parent, -1, self.str_value,
                          style = style | wx.TE_PROCESS_ENTER)
    wx.EVT_TEXT_ENTER( parent, control.GetId(), self.update_object )
else:
    control = wx.TextCtrl( parent, -1, self.str_value, style = style)
wx.EVT_KILL_FOCUS( control, self.update_object )

if factory.auto_set:
    wx.EVT_TEXT( parent, control.GetId(), self.update_object )

self.control = control
self.set_tooltip()
```

initinit__init__

matplotlib

EnthoughtChacomatplotlibmatplotlibTraitsUImatplotlib

```
# -*- coding: utf-8 -*-
# file name: mpl_figure_editor.py
import wx
import matplotlib
# matplotlibWXAggwxtraitsUI
matplotlib.use("WXAgg")
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigureCanvas
from matplotlib.backends.backend_wx import NavigationToolbar2Wx
from enthought.traits.ui.wx.editor import Editor
from enthought.traits.ui.basic_editor_factory import BasicEditorFactory

class _MPLFigureEditor(Editor):
    """
    WX
    """
    scrollable = True

    def init(self, parent):
        self.control = self._create_canvas(parent)
        self.set_tooltip()
        print dir(self.item)

    def update_editor(self):
        pass

    def _create_canvas(self, parent):
        """
        Panel, BoxSizer, panel
        FigureCanvas, NavigationToolbar2Wx, StaticText
        FigureCanvasmousemovedStaticText

        """
        panel = wx.Panel(parent, -1, style=wx.CLIP_CHILDREN)
        def mousemoved(event):
            panel.info.SetLabel("%s, %s" % (event.xdata, event.ydata))
        panel.mousemoved = mousemoved
        sizer = wx.BoxSizer(wx.VERTICAL)
        panel.SetSizer(sizer)
        mpl_control = FigureCanvas(panel, -1, self.value)
        mpl_control.mpl_connect("motion_notify_event", mousemoved)
```

```

toolbar = NavigationToolbar2Wx(mpl_control)
sizer.Add(mpl_control, 1, wx.LEFT | wx.TOP | wx.GROW)
sizer.Add(toolbar, 0, wx.EXPAND|wx.RIGHT)
panel.info = wx.StaticText(parent, -1)
sizer.Add(panel.info)

self.value.canvas.SetMinSize((10,10))
return panel

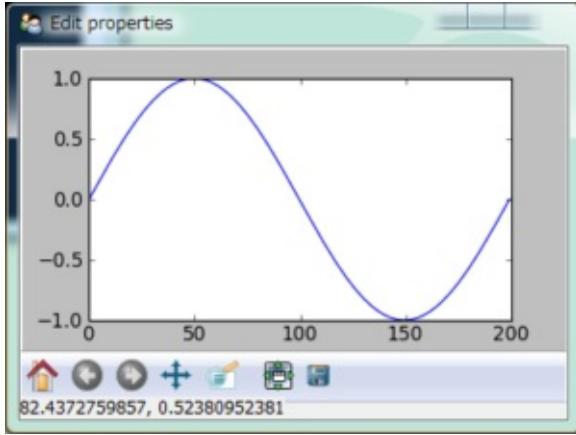
class MPLFigureEditor(BasicEditorFactory):
    """
    traits.uiEditorFactory
    """
    klass = _MPLFigureEditor

if __name__ == "__main__":
    from matplotlib.figure import Figure
    from enthought.traits.api import HasTraits, Instance
    from enthought.traits.ui.api import View, Item
    from numpy import sin, cos, linspace, pi

    class Test(HasTraits):
        figure = Instance(Figure, ())
        view = View(
            Item("figure", editor=MPLFigureEditor(), show_label=False,
                 width = 400,
                 height = 300,
                 resizable = True)
        def __init__(self):
            super(Test, self).__init__()
            axes = self.figure.add_subplot(111)
            t = linspace(0, 2*pi, 200)
            axes.plot(sin(t))

    Test().configure_traits()

```



TraitsUImatplotlib

simplewxtraits.uiBasicEditorFactory
%ui%basic_editor_factory.py

```
class BasicEditorFactory ( EditorFactory ):  
    klass = Any  
  
    def _get_simple_editor_class ( self ):  
        return self(klass  
        ...
```

EditorFactorysimple_editor_classklassMPLFigureEditor
BasicEditorFactory_MPLFigureEditor

text_editor.SimpleEditorEditor_MPLFigureEditorinit
Editorupdate_editortraitupdate_editor

matplotlibFigureCanvasFigure

```
mpl_control = FigureCanvas(panel, -1, self.value)
```

self.valueFigureMVCTest

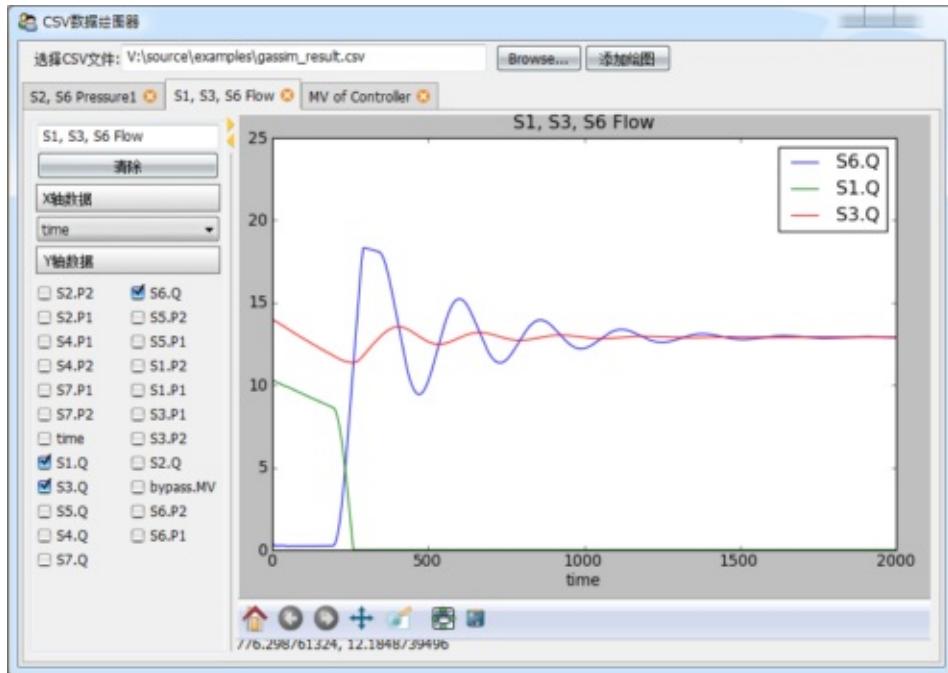
```
figure = Instance(Figure, ())
```

self.value_MPLFigureEditorself.valueTestself.figure

_create_canvaswxmatplotlibwxPython

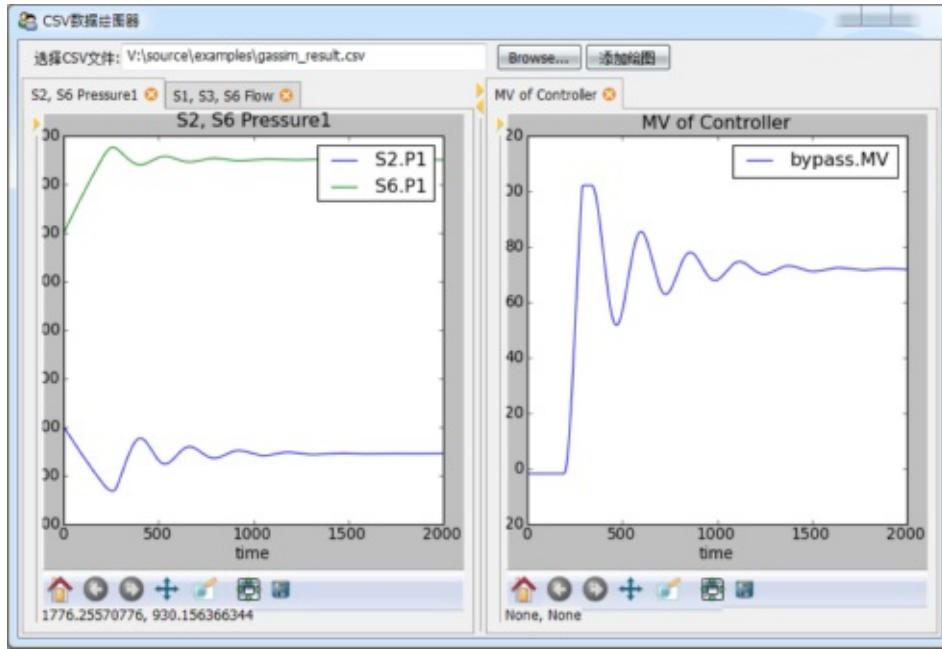
CSV

matplotlibCSVCSVX-YXY



CSV

XY



DOCK

matplotlibTraitsUI

mpl_figure_editor.py170

```

1  # -*- coding: utf-8 -*-
2  from matplotlib.figure import Figure
3  from mpl_figure_editor import MPLFigureEditor
4  from enthought.traits.ui.api import *
5  from enthought.traits.api import *
6  import csv
7
8  class DataSource(HasTraits):
9      """
10         data
11         namesdata
12         """
13         data = DictStrAny
14         names = List(Str)
15
16     def load_csv(self, filename):
17         """
18             CSVdatanames
19             """
20         f = file(filename)
```

```

21     reader = csv.DictReader(f)
22     self.names = reader.fieldnames
23     for field in reader.fieldnames:
24         self.data[field] = []
25     for line in reader:
26         for k, v in line.iteritems():
27             self.data[k].append(float(v))
28     f.close()
29
30 class Graph(HasTraits):
31     """
32
33     """
34     name = Str #
35     data_source = Instance(DataSource) #
36     figure = Instance(Figure) # Figure
37     selected_xaxis = Str # X
38     selected_items = List # Y
39
40     clear_button = Button(u"\u2605") # Y
41
42     view = View(
43         HSplit( # HSplit
44             #
45             VGroup(
46                 Item("name"), #
47                 Item("clear_button"), #
48                 Heading(u"\u2605"), #
49                 # XEnumEditorComboBox
50                 # data_sourcenames
51                 Item("selected_xaxis", editor=
52                     EnumEditor(name="object.data_source.name"),
53                     Heading(u"\u2605"), #
54                     # YYCheckBox
55                     Item("selected_items", style="custom",
56                         editor=CheckListEditor(name="object.selected_items",
57                         cols=2, format_str=u"%s")),
58                     show_border = True, #
59                     scrollable = True, #
60                     show_labels = False #
61                 ),
62                 #
63                 Item("figure", editor=MPLFigureEditor(), show=True),
64             )
65         )
66
67     def _name_changed(self):

```

```
68     """
69
70     """
71     axe = self.figure.axes[0]
72     axe.set_title(self.name)
73     self.figure.canvas.draw()
74
75     def _clear_button_fired(self):
76         """
77
78         """
79         self.selected_items = []
80         self.update()
81
82     def _figure_default(self):
83         """
84             figureFigure
85         """
86         figure = Figure()
87         figure.add_axes([0.05, 0.1, 0.9, 0.85]) #
88         return figure
89
90     def _selected_items_changed(self):
91         """
92             Y
93         """
94         self.update()
95
96     def _selected_xaxis_changed(self):
97         """
98             X
99         """
100        self.update()
101
102    def update(self):
103        """
104
105        """
106        axe = self.figure.axes[0]
107        axe.clear()
108        try:
109            xdata = self.data_source.data[self.selected_xa
110        except:
111            return
112        for field in self.selected_items:
113            axe.plot(xdata, self.data_source.data[field],
114            axe.set_xlabel(self.selected_xaxis)
```

```

115         axe.set_title(self.name)
116         axe.legend()
117         self.figure.canvas.draw()
118
119
120     class CSVGrapher(HasTraits):
121         """
122
123         """
124         graph_list = List(Instance(Graph)) #
125         data_source = Instance(DataSource) #
126         csv_file_name = File(filter=[u"*.csv"]) #
127         add_graph_button = Button(u"") #
128
129         view = View(
130             #
131             VGroup(
132                 # HGroup
133                 HGroup(
134                     #
135                     Item("csv_file_name", label=u"CSV" , wi
136                     #
137                     Item("add_graph_button", show_label=False)
138                 ),
139                 # ListEditor
140                 Item("graph_list", style="custom", show_label=
141                     editor=ListEditor(
142                         use_notebook=True, #
143                         deletable=True, #
144                         dock_style="tab", # dock
145                         page_name=".name") # Graphname
146                     )
147                 ),
148                 resizable = True,
149                 height = 0.8,
150                 width = 0.8,
151                 title = u"CSV"
152             )
153
154     def _csv_file_name_changed(self):
155         """
156         DataSource
157         """
158         self.data_source = DataSource()
159         self.data_source.load_csv(self.csv_file_name)
160         del self.graph_list[:]
161

```

```
162     def _add_graph_button_changed(self):
163         """
164         """
165         if self.data_source != None:
166             self.graph_list.append( Graph(data_source = self.data_source) )
167
168     if __name__ == "__main__":
169         csv_grapher = CSVGrapher()
170         csv_grapher.configure_traits()
```

traits

- View
- 141ListEditorgraph_listGraph
- 43GraphHSplitHSplit
- traitsUI3.2TraitsUIunicodeTraitsUI770str
unicode

```
File "C:\Python26\lib\site-packages\traitsbackendwx-3.2.0-p
traits\ui\wx\list_editor.py", line 770, in _create_page
name = str( name ) or '???
UnicodeEncodeError: 'ascii' codec can't encode characters i
ordinal not in range(128)
```

- matplotlib

Python »

← (Alt+X) | |

Visual

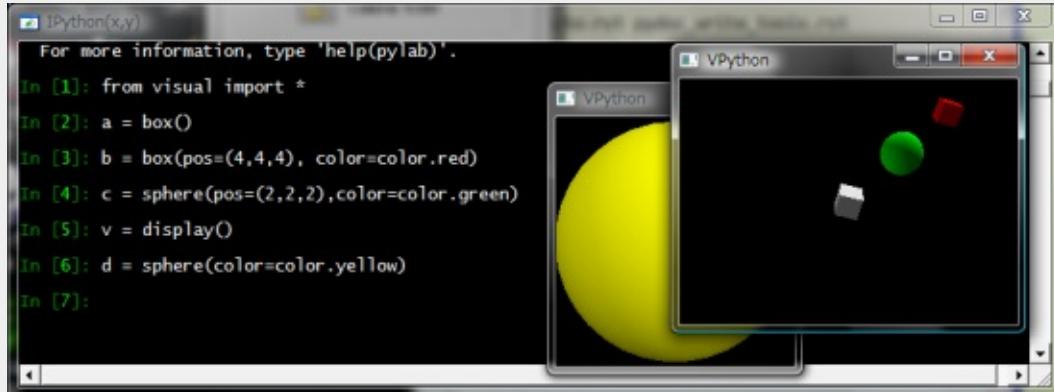
ipython

visualIPythonipython

```
>>> from visual import *
```

visualvisualipython

```
>>> scene.visible = False
```



IPythonvisual

IPythonvisualIPython

●

○

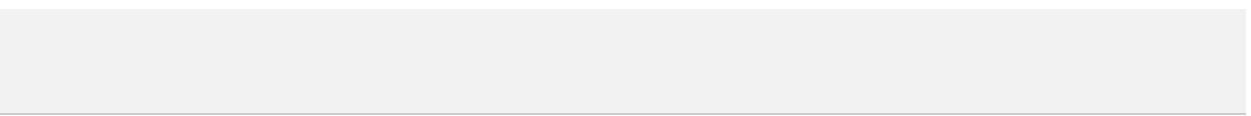
○

Python »

← (Alt+X) | |

Python » Visual »

← (Alt+X) | |



visual3Ddisplay

visualscene

```
>>> from visual import *
>>> scene
<visual.ui.display object at 0x032BF600>
```

visual.ui.displaydisplay()box3Ddisplay

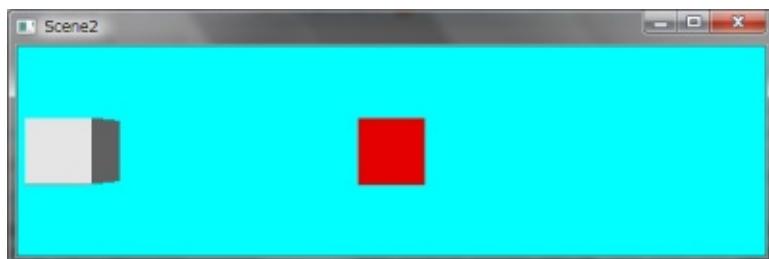
```
>>> scene2 = display(title='Scene2', x=0, y=0, width=600, height=400,
center=(5,0,0), background=(0,1,1))
```

Scene2(0,0)600200(5,0,0)3D(5,0,0)

```
>>> box()
>>> <visual.primitives.box object at 0x0334F090>
```

```
>>> box(pos=(5,0,0), color=color.red)
>>> <visual.primitives.box object at 0x0334F120>
```

(0,0,0)(5,0,0)center

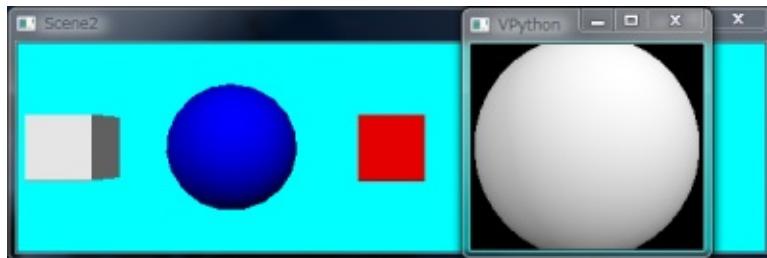


selectdisplay.get_selected()

```
>>> scene.select()
>>> sphere()
<visual.primitives.sphere object at 0x0331D7B0>
>>> scene2.select()
>>> sphere(pos=(2.5,0,0), color=color.blue)
```

```
<visual.primitives.sphere object at 0x0331D810>
>>> display.get_selected() == scene2
True
```

```
scenescene2(2.5,0,0)display.get_selected()
scene2VPythonScene2
```



- **foreground** : `scene.foreground = color.green`

- **background** :

- **ambient** : `color.gray(0.2)`
`visual 3scene.ambient=0.2`
`scene.ambient=color.gray(0.2)`

- **lights** :

```
[distant_light(direction=(0.22, 0.44, 0.88), color=color.gr
distant_light(direction=(-0.88, -0.22, -0.44), color=color
```

```
>>> scene.lights[0].direction
vector(0.218217890235992, 0.436435780471985, 0.872871560943
```

- **cursor.visible** : `False`
- **objects** : `visiblebox`

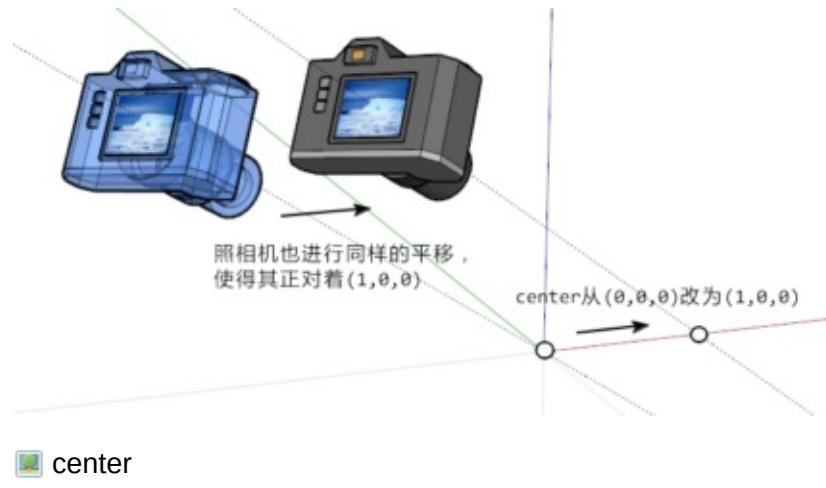
```
for obj in scene2.objects:  
    if isinstance(obj, box):  
        obj.color = color.red
```

- **show_rendertime** : true "cycle:27: 5" 275
Python22
- **stero** : 3Dscene.stereo="redcyan"-
"redblue""yellowblue""crosseyed"
"active"shutter glasses
- **stereodepth** : 02

x, y, width, height, titlefullscreenvisible = False

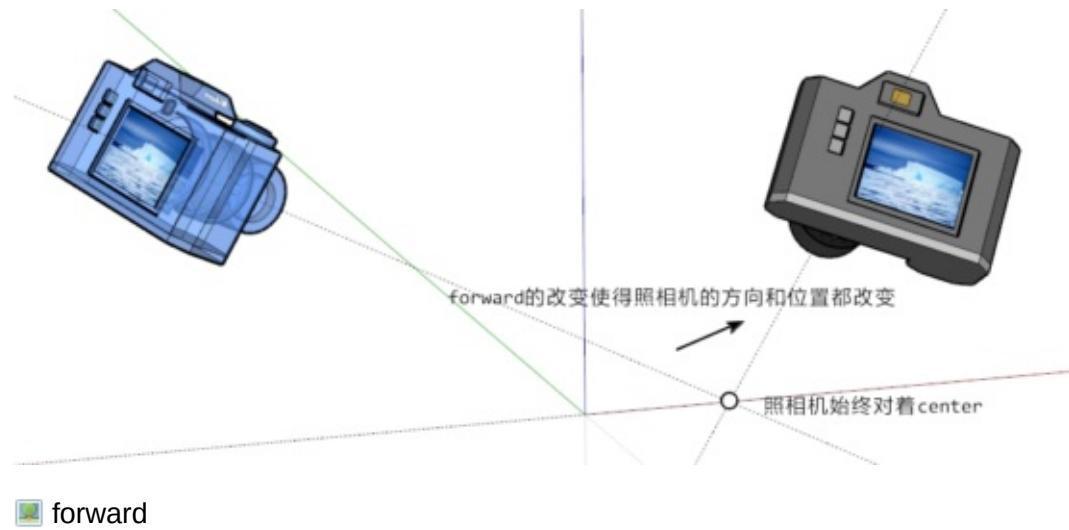
- **x, y** :
- **width, height** :
- **title** : windowsscene.title = u"".encode("gb2312")
- **fullscreen** : scene.fullscreen=TrueEsc
- **visible** :
- **exit** : exitFalseTrue

- **center** : 3Dcentercentercenter(0,0,0)



- **autocenter** : Truecenterx, y, zcenter

- **forward** : centerscene.mouse.cameraforward
forwardforwardcenterforward(0,0,-1)



Python » Visual »

← (Alt+X) | |

Python »

← (Alt+X) | |

WAVPythonWAV

pyAudio(<http://people.csail.mit.edu/hubert/pyaudio>)

pyMedia(<http://pymedia.org>)Mp3

Wave

WAV Microsoft PCM) WAV

-
- 8kHz, 16kHz, 32kHz, 48kHz, 11.025kHz, 22.05kHz, 44.1kHz
- bit8bit16bit24bit32bit

CD 44.1kHz 16bit

Audacity(<http://audacity.sourceforge.net>) Audacity
WAV Python

Wave

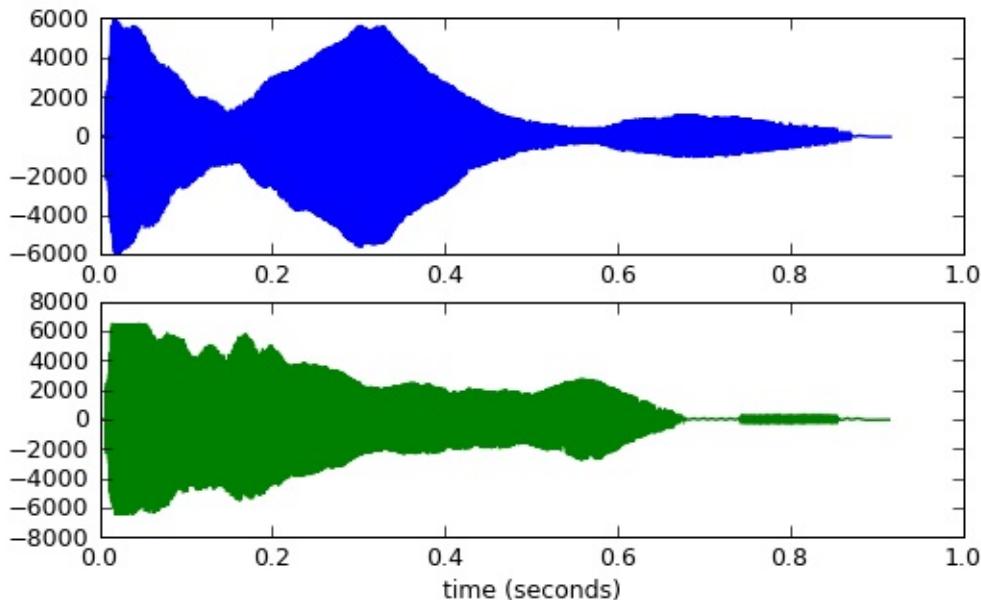
Python

```
1 # -*- coding: utf-8 -*-
2 import wave
3 import pylab as pl
4 import numpy as np
5
6 # WAV
7 f = wave.open(r"c:\WINDOWS\Media\ding.wav", "rb")
8
9 #
10 # (nchannels, sampwidth, framerate, nframes, comptype, comp
11 params = f.getparams()
12 nchannels, sampwidth, framerate, nframes = params[:4]
13
14 #
15 str_data = f.readframes(nframes)
16 f.close()
17
18 #
19 wave_data = np.fromstring(str_data, dtype=np.short)
20 wave_data.shape = -1, 2
```

```

21 wave_data = wave_data.T
22 time = np.arange(0, nframes) * (1.0 / framerate)
23
24 #
25 pl.subplot(211)
26 pl.plot(time, wave_data[0])
27 pl.subplot(212)
28 pl.plot(time, wave_data[1], c="g")
29 pl.xlabel("time (seconds)")
30 pl.show()

```



WindowsXP""

PythonWAVwave.open("ding.wav", "rb")

```

import wave
f = wave.open(r"c:\WINDOWS\Media\ding.wav", "rb")

```

openWave_readWAV

- getparamsWAV(tuple), byte, , , ,
wave

```

params = f.getparams()
nchannels, sampwidth, framerate, nframes = params[:4]

```

- getnchannels, getsampwidth, getframerate, getnframesWAV
- readframesreadframesbytes)Python

```
str_data = f.readframes(nframes)
```

```
wave_data = np.fromstring(str_data, dtype=np.short)
```

```
fromstringdtypeshortwave_datashort  
LRLRLRLR....LRLRwave_datasharp
```

```
wave_data.shape = -1, 2
```

```
wave_data = wave_data.T
```

```
time = np.arange(0, nframes) * (1.0 / framerate)
```

Wave

WAV

```
1 # -*- coding: utf-8 -*-
2 import wave
3 import numpy as np
4 import scipy.signal as signal
5
6 framerate = 44100
7 time = 10
8
```

```
9 # 1044.1kHz100Hz - 1kHz
10 t = np.arange(0, time, 1.0/framerate)
11 wave_data = signal.chirp(t, 100, time, 1000, method='linear')
12 wave_data = wave_data.astype(np.short)
13
14 # WAV
15 f = wave.open(r"sweep.wav", "wb")
16
17 #
18 f.setnchannels(1)
19 f.setsampwidth(2)
20 f.setframerate(framerate)
21 # wav_data
22 f.writeframes(wave_data.tostring())
23 f.close()
```

10-12scipy.signalchrip1044.1kHz100Hz1kHzchripfloat64
astypeshort

18-20WAVsetparams21writeframeswriteframesWAV
(nframes)

pyAudio

pyAudioWAVKaraoke

pyAudioPortAudio(<http://www.portaudio.com>)Python

pyAudio

```
1 # -*- coding: utf-8 -*-
2 import pyaudio
3 import wave
4
5 chunk = 1024
6
7 wf = wave.open(r"c:\WINDOWS\Media\ding.wav", 'rb')
8
9 p = pyaudio.PyAudio()
10
11 #
12 stream = p.open(format = p.get_format_from_width(wf.getsampwidth()),
13                   channels = wf.getnchannels(),
14                   rate = wf.getframerate(),
15                   output = True)
16
17 #
18 while True:
19     data = wf.readframes(chunk)
20     if data == "": break
21     stream.write(data)
22
23 stream.close()
24 p.terminate()
```

WAVopenWAVopen17-20whilepyAudio
stream.writestream.write

PyAudioopen

- **rate** -
- **channels** -
- **format** - (paFloat32, paInt32, paInt24, paInt16, paInt8 ...) get_format_from_widthwf.sampwidth()2paInt16
- **input** - True
- **output** - True
- **input_device_index** -
- **output_device_index** -
- **frames_per_buffer** - N
- **start** - True

pyAudio

```
1 # -*- coding: utf-8 -*-
2 from pyaudio import PyAudio, paInt16
3 import numpy as np
4 from datetime import datetime
5 import wave
6
7
8 # datafilenameWAV
9 def save_wave_file(filename, data):
10     wf = wave.open(filename, 'wb')
11     wf.setnchannels(1)
12     wf.setsampwidth(2)
13     wf.setframerate(SAMPLING_RATE)
14     wf.writeframes("".join(data))
15     wf.close()
16
17
18
19 NUM_SAMPLES = 2000      # pyAudio
20 SAMPLING_RATE = 8000    #
21 LEVEL = 1500            #
22 COUNT_NUM = 20          # NUM_SAMPLESCOUNT_NUMLEVEL
23 SAVE_LENGTH = 8         # SAVE_LENGTH * NUM_SAMPLES
```

```

24
25 #
26 pa = PyAudio()
27 stream = pa.open(format=paInt16, channels=1, rate=SAMPLING_
28                     frames_per_buffer=NUM_SAMPLES)
29
30 save_count = 0
31 save_buffer = []
32
33 while True:
34     # NUM_SAMPLES
35     string_audio_data = stream.read(NUM_SAMPLES)
36     #
37     audio_data = np.fromstring(string_audio_data, dtype=np.
38     # LEVEL
39     large_sample_count = np.sum( audio_data > LEVEL )
40     print np.max(audio_data)
41     # COUNT_NUMSAVE_LENGTH
42     if large_sample_count > COUNT_NUM:
43         save_count = SAVE_LENGTH
44     else:
45         save_count -= 1
46
47     if save_count < 0:
48         save_count = 0
49
50     if save_count > 0:
51         # save_buffer
52         save_buffer.append( string_audio_data )
53     else:
54         # save_bufferWAVWAV
55         if len(save_buffer) > 0:
56             filename = datetime.now().strftime("%Y-%m-%d_%H%
57             save_wave_file(filename, save_buffer)
58             save_buffer = []
59             print filename, "saved"

```

SAMPLING RATE
NUM_SAMPLES
COUNT_NUM
LEVEL
WAV
SAVE_LENGTH
WAV

WAV
paInt16(16bitshort)
dtype
np.short

pyMediaMp3

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

FIRIIR

FIR()IIR()

FIR

$$y[m] = b[0]x[m] + b[1]x[m-1] + \dots + b[P]x[n-P]$$

IIR(1)

$$\begin{aligned} y[m] &= b[0]x[m] + b[1]x[m-1] + \dots + b[P]x[n-P] \\ &\quad - a[1]y[m-1] - a[2]y[m-2] - \dots - a[Q]y[m-Q] \end{aligned}$$

xabyFIRIIRa0IIRFIR

FIRmy[m]mx[m]x[m-1] ... x[m-P]b[0] ... b[P]IIRy[m-1] ... y[m-Q]
a[1] ... a[m-Q]

PythonFIRCD441001004Python

scipysignalfilterC

```
signal.lfilter(b, a, x, axis=-1, zi=None)
```

baxlfilterIIR

$$y[m] = b[0]*x[m] + z[0, m-1] \quad (1)$$

$$z[0, m] = b[1]*x[m] + z[1, m-1] - a[1]*y[m] \quad (2)$$

...

$$z[n-3, m] = b[n-2]*x[m] + z[n-2, m-1] - a[n-2]*y[m]$$

$$z[n-2, m] = b[n-1]*x[m] - a[n-1]*y[m]$$

(2)m-1

$$z[0, m-1] = b[1]*x[m-1] + z[1, m-2] - a[1]*y[m-1] \quad (3)$$

(1)b[0]*x[m], b[1]*x[m-1], - a[1]*y[m-1]IIRIIR2

```
1my[m]mx[m]x[m-1]x[m-P]y[m-1]y[m-Q]P+Q2n-1  
z[0]z[n-2]n max(len(a), len(b)) max(P, Q)z
```

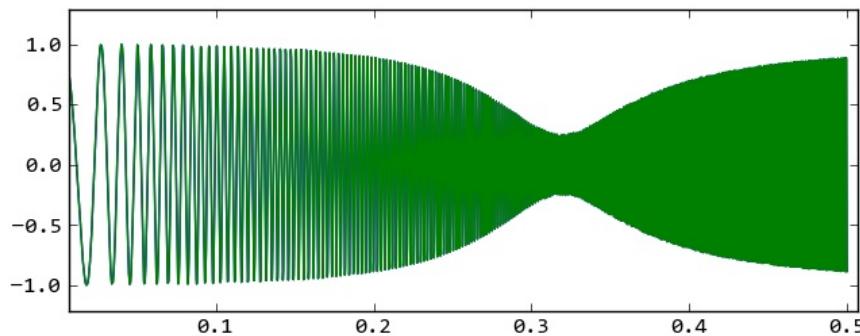
```
zifilterziNonelfilterzf(y, zf)ziNoney
```

```
lfilterxlfilterzffilter
```

```
1  # -*- coding: utf-8 -*-
2  import scipy.signal as signal
3  import numpy as np
4  import pylab as pl
5
6  #
7  a = np.array([1.0, -1.947463016918843, 0.9555873701383931])
8  b = np.array([0.9833716591860479, -1.947463016918843, 0.972
9
10 # 44.1kHz 1
11 t = np.arange(0, 0.5, 1/44100.0)
12 x= signal.chirp(t, f0=10, t1 = 0.5, f1=1000.0)
13
14 #
15 y = signal.lfilter(b, a, x)
16
17 # 50
18 x2 = x.reshape((-1,50))
19
20 # 0 -1
21 z = np.zeros(max(len(a),len(b))-1, dtype=np.float)
22 y2 = [] #
23
24 for tx in x2:
25     # z
26     ty, z = signal.lfilter(b, a, tx, zi=z)
27     #
28     y2.append(ty)
29
30 # y2
31 y2 = np.array(y2)
32 y2 = y2.reshape((-1, ))
33
34 # yy2
35 print np.sum((y-y2)**2)
36
37 #
```

```
38 pl.plot(t, y, t, y2)
39 pl.show()
```

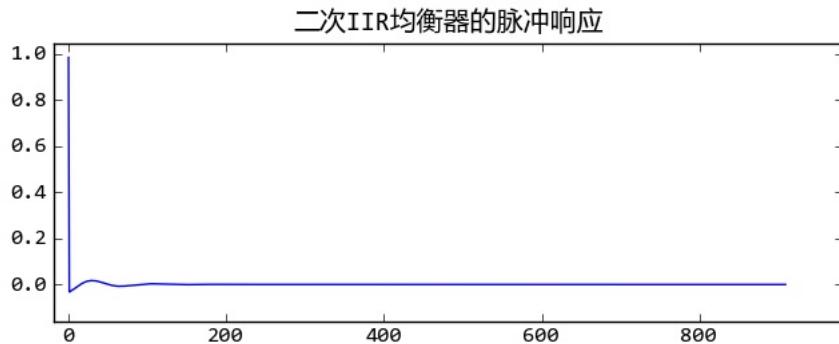
0



IIRfilterpyaudio

010FIRFIRIIRfilterIIRIIRIPythonSpyder

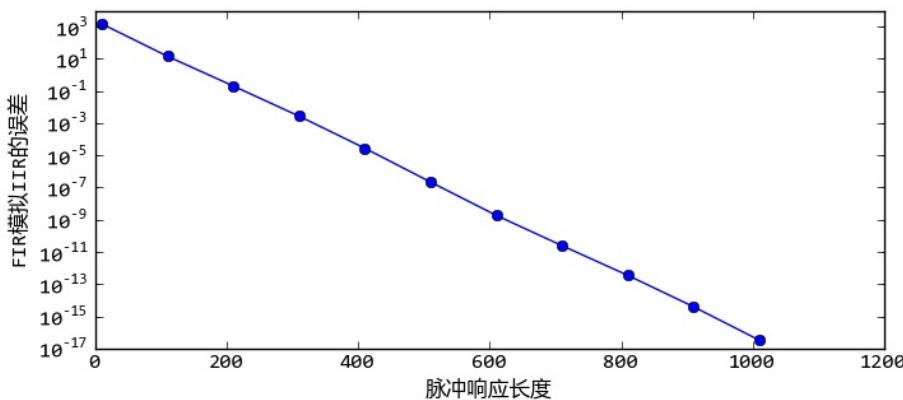
```
>>> impulse = np.zeros(1000, dtype=np.float)
>>> impulse[0] = 1
>>> h = signal.lfilter(b, a, impulse)
>>> h[-1]
-4.2666825205952273e-12
```



hh0hFIRx

```
>>> y3 = signal.lfilter(h, 1, x)
>>> np.sum((y-y3)**2)
3.7835244127856444e-17
```

hFIRIIRFIRIIRY

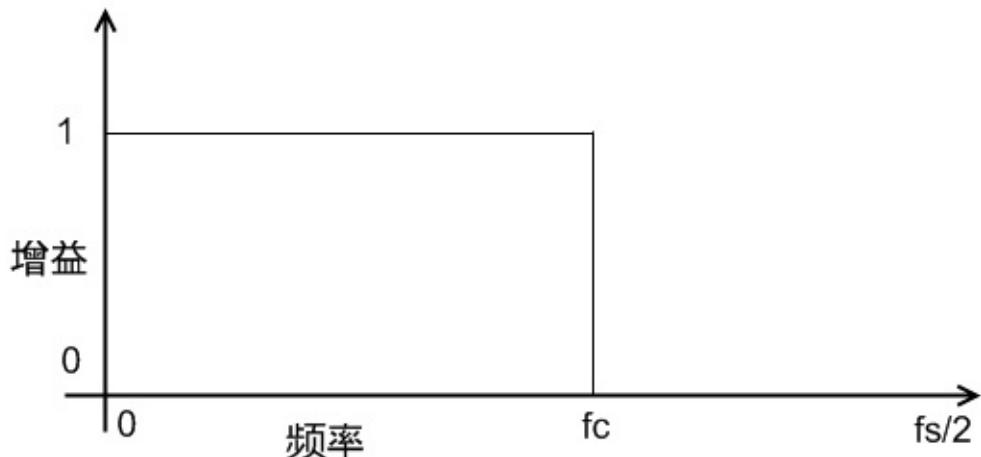


IfilterFIRa1

```
1 # -*- coding: utf-8 -*-
2 import scipy.signal as signal
3 import numpy as np
4 import pylab as pl
5
6 #
7 a = np.array([1.0, -1.947463016918843, 0.9555873701383931])
8 b = np.array([0.9833716591860479, -1.947463016918843, 0.972
9
10 # 44.1kHz 1
11 t = np.arange(0, 0.5, 1/44100.0)
12 x= signal.chirp(t, f0=10, t1 = 0.5, f1=1000.0)
13 y = signal.lfilter(b, a, x)
14 ns = range(10, 1100, 100)
15 err = []
16
17 for n in ns:
18     #
19     impulse = np.zeros(n, dtype=np.float)
20     impulse[0] = 1
21     h = signal.lfilter(b, a, impulse)
```

```
22
23     # FIR
24     y2 = signal.lfilter(h, 1, x)
25
26     # yy2
27     err.append(np.sum((y-y2)**2))
28
29 #
30 pl.figure(figsize=(8,4))
31 pl.semilogy(ns , err, "-o")
32 pl.xlabel(u"")
33 pl.ylabel(u"FIRIIR")
34 pl.show()
```

FIR



$$f_s \quad f_c \quad 1 \quad f_c \quad 0.10.110$$

$$h_{ideal}(n) = \frac{\sin(2\pi f_c)}{\pi n} = 2f_c \operatorname{sinc}(2f_c n)$$

n00

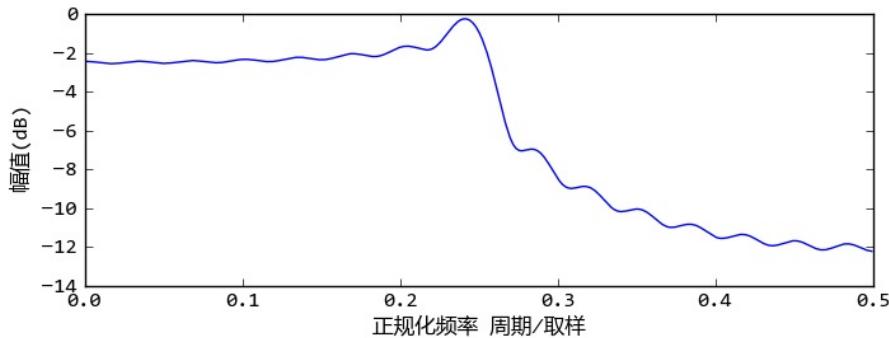
FIR h_{ideal} $0 \leq n < L$ LFIR

```
1 # -*- coding: utf-8 -*-
2 import scipy.signal as signal
3 import numpy as np
4 import pylab as pl
5
6 def h_ideal(n, fc):
7     return 2*fc*np.sinc(2*fc*np.arange(-n, n, 1.0))
8
9 b = h_ideal(30, 0.25)
10 w, h = signal.freqz(b)
11
12
```

```

13 pl.figure(figsize=(8,3))
14 pl.plot(w/2/np.pi, 20*np.log10(np.abs(h)))
15 pl.xlabel(u" / ")
16 pl.ylabel(u"(dB) ")
17 pl.show()

```



sinc

freqz

freqz

```
freqz(b, a=1, worN=None, whole=0, plot=None)
```

baworNwhole0piwhole12*pi

(w,h) ww/(2*pi)h

freqz 20*np.log10(np.abs(h)) hdB

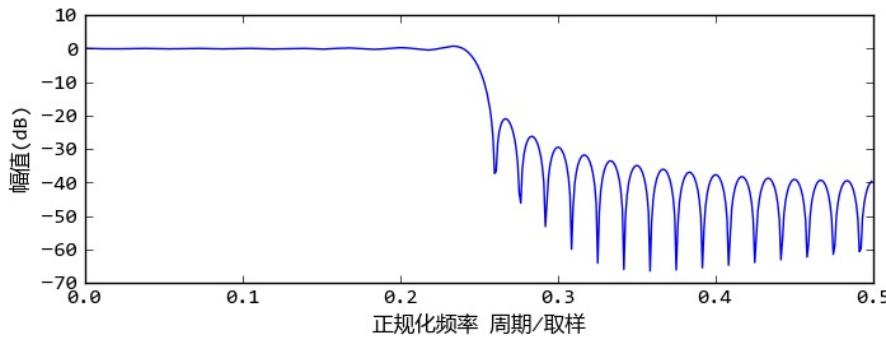
firwin

FIRn<0n>=0n<0h_ideal

h_{ideal}^{-nn}

```
def h_ideal(n, fc):
    return 2*fc*np.sinc(2*fc*np.arange(-n, n, 1.0))
```

n<0



sinc

SciPyfirwinfirwin

```
firwin(N, cutoff, width=None, window='hamming')
```

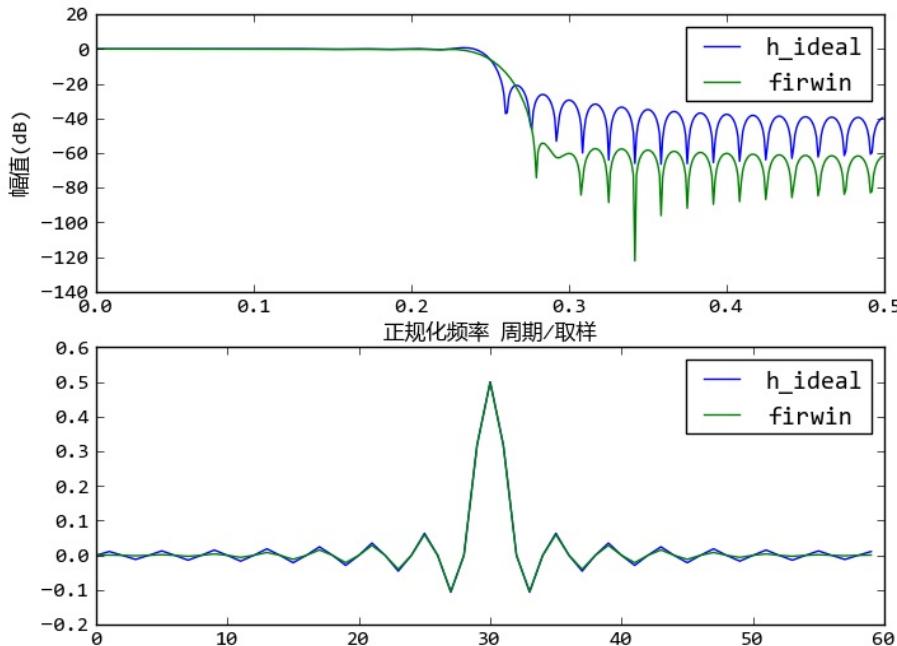
Ncutoff $f_s/2$ window

firwinfirwincutoff/2 f_c

```

1 # -*- coding: utf-8 -*-
2 import scipy.signal as signal
3 import numpy as np
4 import pylab as pl
5
6 def h_ideal(n, fc):
7     return 2*fc*np.sinc(2*fc*np.arange(-n, n, 1.0))
8
9 b = h_ideal(30, 0.25) # fs
10 b2 = signal.firwin(len(b), 0.5) # fs/2
11
12 w, h = signal.freqz(b)
13 w2, h2 = signal.freqz(b2)
14
15 pl.figure(figsize=(8,6))
16 pl.subplot(211)
17 pl.plot(w/np.pi, 20*np.log10(np.abs(h)), label=u"h_ideal")
18 pl.plot(w2/np.pi, 20*np.log10(np.abs(h2)), label=u"firwin")
19 pl.xlabel(u" / ")
20 pl.ylabel(u"(dB)")
```

```
21 pl.legend()
22 pl.subplot(212)
23 pl.plot(b, label=u"h_ideal")
24 pl.plot(b2, label=u"firwin")
25 pl.legend()
26 pl.show()
```



firwinFIR

remez

remez

```
remez(numtaps, bands, desired,
      weight=None, Hz=1, type='bandpass', maxiter=25, grid_density=:
```

- **numtaps** : FIR

- **bands** 0Hz/2Hz1
- **desired** : bandsbands
- **weight** : desireddesireddesired
- **type** : 'bandpass"ifferentiator'type'bandpass'

remez

remeznFIRremezFIR

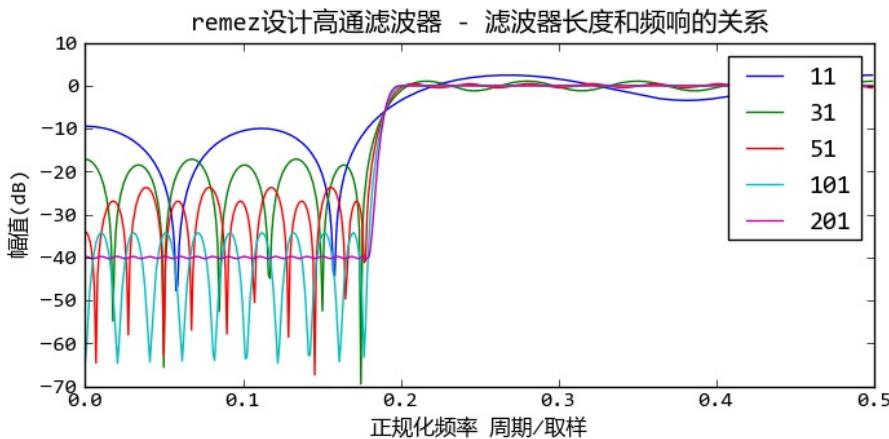
remezremezFIRfirwinnumtaps/20

remez

```

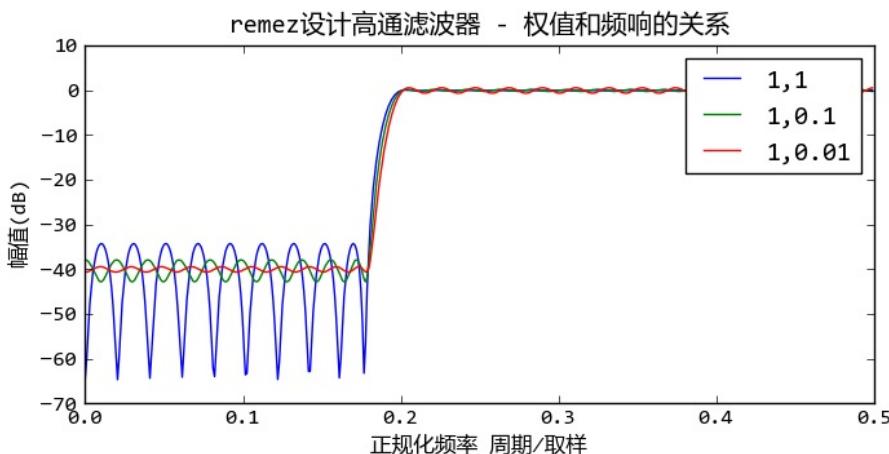
1 # -*- coding: utf-8 -*-
2 import scipy.signal as signal
3 import numpy as np
4 import pylab as pl
5
6 for length in [11, 31, 51, 101, 201]:
7     b = signal.remez(length, (0, 0.18, 0.2, 0.50), (0.01,
8         w, h = signal.freqz(b, 1)
9         pl.plot(w/2/np.pi, 20*np.log10(np.abs(h)), label=str(length))
10    pl.legend()
11    pl.xlabel(u" /")
12    pl.ylabel(u"(dB)")
13    pl.title(u"remez - ")
14    pl.show()
```

remezbands()0.180.20.5desired0.0110.2-40dB



remez

1011, 0.01()desired



remez

h1h2h1h2h1h2h1h2

remezh1h2h3

```

1 # -*- coding: utf-8 -*-
2 import scipy.signal as signal
3 import numpy as np
4 import pylab as pl

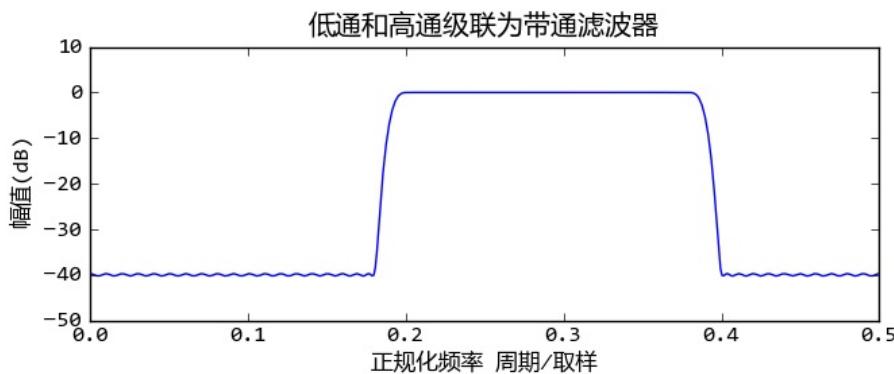
```

```

5
6 h1 = signal.remez(201, (0, 0.18, 0.2, 0.50), (0.01, 1))
7 h2 = signal.remez(201, (0, 0.38, 0.4, 0.50), (1, 0.01))
8 h3 = np.convolve(h1, h2)
9
10 w, h = signal.freqz(h3, 1)
11 pl.plot(w/np.pi, 20*np.log10(np.abs(h)))
12
13 pl.legend()
14 pl.xlabel(u" /")
15 pl.ylabel(u"(dB)")
16 pl.title(u"")
17 pl.show()

```

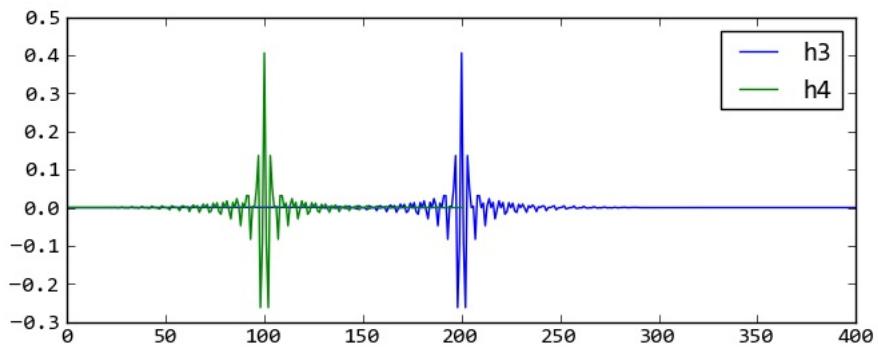
freqzh3



remez

```
>>> h4 = signal.remez(201, (0, 0.18, 0.2, 0.38, 0.4, 0.50), (0.01
```

h3h3h4



remz

h3h40h3h4

IIR

$$IIRsH(s)H(s)s$$

$$|H(j\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}$$

$$n = \omega_c 3 \text{dB}$$

- $\omega < \omega_c$ $\omega 1$
- $\omega > \omega_c$ $\omega 0$
- $n10n$
- $\omega = \omega_c 1/2-3 \text{dB}$

$$H(s) \quad s = \delta + j\omega$$

$$s = j\omega \quad H(s)H(-s) = |H(j\omega)|^2 \omega = s/j$$

$$H(s)H(-s) = \frac{|G_0|^2}{1 + \left(\frac{-s^2}{\omega_c^2}\right)^n}$$

$$2nnnH(s)nH(s)$$

$$H(s) = \frac{1}{\prod_{k=1}^n (s - s_k)/\omega_c}$$

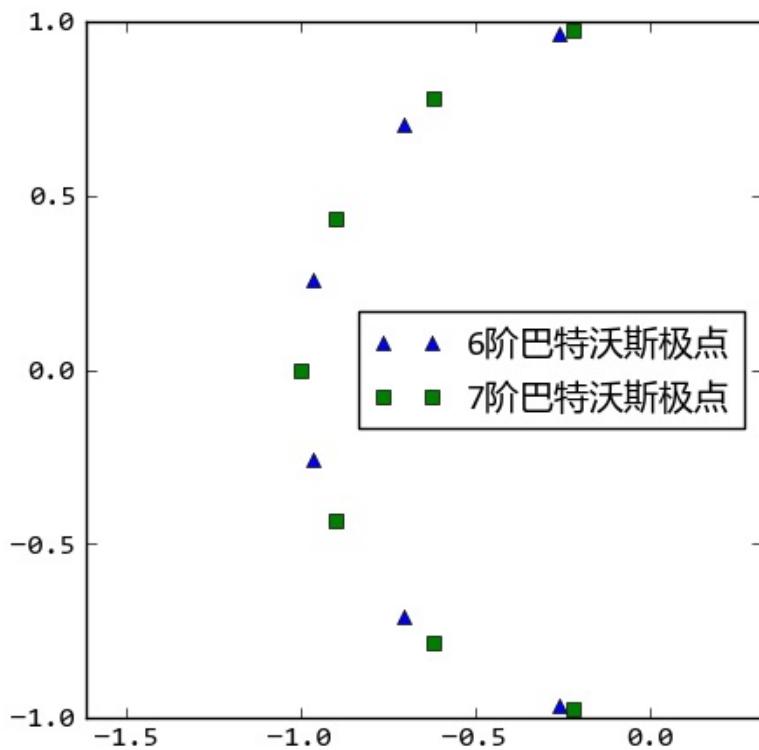
$$s_k$$

$$s_k = \omega_c e^{\frac{j(2k+n-1)\pi}{2n}} \quad k = 1, 2, 3, \dots, n$$

67S

```
1 # -*- coding: utf-8 -*-
2 from scipy import signal
3 import numpy as np
4 import matplotlib.pyplot as pl
5
6 pl.figure(figsize=(5,5))
7
8 b, a = signal.butter(6, 1.0, analog=1)
9 z,p,k = signal.tf2zpk(b, a)
10 pl.plot(np.real(p), np.imag(p), '^', label=u"6")
11
12 b, a = signal.butter(7, 1.0, analog=1)
13 z,p,k = signal.tf2zpk(b, a)
14 pl.plot(np.real(p), np.imag(p), 's', label=u"7")
15
16 pl.axis("equal")
17 pl.legend(loc="center right")
18 pl.show()
```

butteranalog=1batf2zpk



 S



$H(s)H(z)$

$$s \leftarrow \frac{2}{T} \frac{z-1}{z+1}$$

T

http://en.wikipedia.org/wiki/Bilinear_transform

SZSZSYz

```

1 # -*- coding: utf-8 -*-
2 import numpy as np
3 import pylab as pl

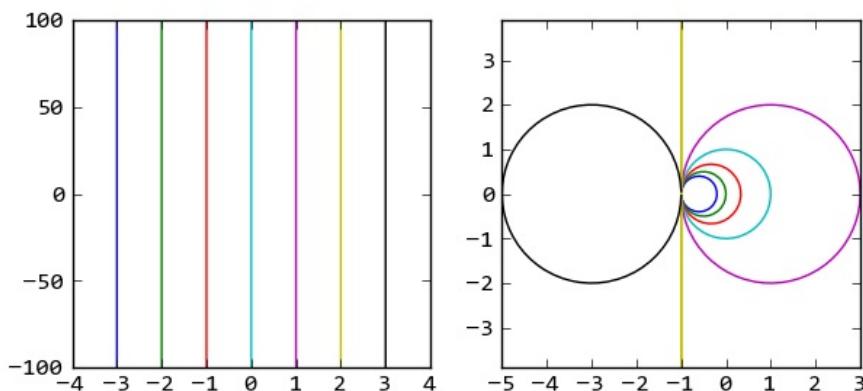
```

```

4
5  def stoz(s):
6      """
7          SZ
8          T=1
9      """
10     return (2+s)/(2-s)
11
12 def make_vline(x):
13     return x + 1j*np.linspace(-100.0,100.0,20000)
14
15 fig = pl.figure(figsize=(7,3))
16 axs = pl.subplot(121)
17 axz = pl.subplot(122)
18 for x in np.arange(-3, 4, 1):
19     s = make_vline(x)
20     z = stoz(s)
21     axs.plot(np.real(s), np.imag(s))
22     axz.plot(np.real(z), np.imag(z))
23
24 axs.set_xlim(-4,4)
25 axz.axis("equal")
26 axz.set_ylim(-3,3)
27
28 pl.show()

```

stozSZT=1



s()z()

$$H(z) \quad z = e^{j\omega T}$$

$$s = \frac{2}{T} \frac{z - 1}{z + 1} = \frac{2}{T} \frac{e^{j\omega T} - 1}{e^{j\omega T} + 1} = \frac{2}{T} \frac{e^{j\omega T/2} - e^{-j\omega T/2}}{e^{j\omega T/2} + e^{-j\omega T/2}} = j \frac{2}{T} \tan(\omega T/2)$$

$s = j\omega H(s) \quad \omega \omega_a$

$$\omega_a = \frac{2}{T} \tan\left(\omega \frac{T}{2}\right)$$

$$\omega = \frac{2}{T} \arctan\left(\omega_a \frac{T}{2}\right)$$

fs8kHz1kHz

```
>>> from scipy import signal
>>> from numpy import *
>>> fs = 8000.0
>>> f = 1000.0
```

butter3analog=1H(s)2*pi

```
>>> b, a = signal.butter(3, 2*pi*f, analog=1)
```

baH(s)

baMNb[0] $s^{M-1}a[0]$ $s^{N-1}b[-1]a[-1]$

$$H(s) = \frac{b_0 s^{M-1} + b_1 s^{M-2} + \dots + b_{M-1}}{a_0 s^{N-1} + a_1 s^{N-2} + \dots + a_{N-1}}$$

bilinearfs

```
>>> b2, a2 = signal.bilinear(b, a, fs=fs)
```

freqzwoN10000

```
>>> w2, h2 = signal.freqz(b2, a2, worN=1000)
```

$h2 = 10 \cdot \log_{10}(0.5) \cdot w_{idxw} / (2 \cdot \pi) \cdot fs$

```
>>> p2 = 20*log10(abs(h2))
>>> idx = argmin(abs(p2-10*log10(0.5)))
>>> w2[idx]/2/pi*8000
952.8
```

```
>>> 2*fs*arctan(2*pi*f/2/fs) /2/pi
952.8840223
```

scipy.signal.IIR2fs/21fsff/(fs/2)butte

```
>>> b3, a3 = signal.butter(3, 952.8840223/(fs/2))
>>> sum(abs(b3-b2))
1.3226225670237568e-13
>>> sum(abs(a3-a2))
7.0876637892069994e-13
```

baH(z)

$$baMNb[0]a[0]a[-1] z^{-(N-1)}b[-1]z^{-(M-1)}$$

$$H(s) = \frac{b_0 + b_1 z^{-1} + \dots + b_{M-1} z^{-(M-1)}}{a_0 + a_1 z^{-1} + \dots + a_{N-1} z^{-(N-1)}}$$

b3a3bilinearb2a2signalbilinear signal.iirfilter

$H(s)$

$1/$

```
>>> b, a = signal.butter(2, 1.0, analog=1)
```

```

>>> np.real(b)
array([ 1.])
>>> np.real(a)
array([ 1.           ,  1.41421356,  1.           ])

```

$$H(s) = \frac{1}{s^2 + 1.4142s + 1}$$

$$\omega_c$$

$$s \rightarrow \frac{s}{\omega_c}$$

$$s = j\omega H(s)H(s) \quad s = j3\text{dB} \quad H(s/\omega_c)s = j\omega_c 3\text{dB}2/2$$

```

>>> b2, a2 = signal.butter(2, 2.0, analog=1)
>>> np.real(b2)
array([ 4.])
>>> np.real(a2)
array([ 1.           ,  2.82842712,  4.           ])

```

$$s \rightarrow \frac{s}{2}H(s)$$

$$s \rightarrow \frac{\omega_c}{s}$$

- ω0000
- ω01

$$1/$$

```

>>> b3, a3 = signal.butter(2, 1.0, btype="high", analog=1)
>>> np.real(b3)
array([ 1.,  0.,  0.])
>>> np.real(a3)
array([ 1.           ,  1.41421356,  1.           ])

```

$$s \rightarrow \frac{1}{s}H(s) \quad s^2$$

$$\omega_2\omega_1$$

$$s \rightarrow \frac{\omega_0}{\Delta\omega} \left(\frac{s}{\omega_0} + \frac{\omega_0}{s} \right)$$

$$\Delta\omega = \omega_2 - \omega_1\omega_0 = \sqrt{\omega_1\omega_2}\Delta\omega \quad \omega_0$$

```

1  # -*- coding: utf-8 -*-
2  import numpy as np
3  from scipy import signal
4  import pylab as pl
5
6  b, a = signal.butter(2, 1.0, analog=1)
7
8  # ->
9  w1 = 1.0 #
10 w2 = 2.0 #
11 dw = w2 - w1 #
12 w0 = np.sqrt(w1*w2) #
13
14 # 10**-210**2
15 w = np.logspace(-2, 2, 1000)
16
17 #
18 nw = np.imag(w0/dw*(1j*w/w0 + w0/(1j*w)))
19
20 _, h = signal.freqs(b, a, worN=nw)
21 h = 20*np.log10(np.abs(h))
22
23 pl.figure(figsize=(8,5))
24
25 pl.subplot(221)
26 pl.semilogx(w, nw) # xlog
27 pl.xlabel(u"(/)") )
28 pl.ylabel(u"(/)") )
29
30 pl.subplot(222)
31 pl.plot(h, nw)
32 pl.xlabel(u"(dB)") )
33
34 pl.subplot(212)
35 pl.semilogx(w, h)

```

```
36 pl.xlabel(u"/")  
37 pl.ylabel(u"(dB)")  
38  
39 pl.subplots_adjust(wspace=0.3, hspace=0.3, top=0.95, bottom=0.1)  
40  
41 print "center:", w[np.argmin(np.abs(nw))]  
42 pl.show()
```

butter

```
b, a = signal.butter(2, 1.0, analog=1)
```

w1=1w2=2

```
# ->  
w1 = 1.0 #  
w2 = 2.0 #  
dw = w2 - w1 #  
w0 = np.sqrt(w1*w2) #
```

0.01100logspacew

```
w = np.logspace(-2, 2, 1000)
```

nw

```
nw = np.imag(w0/dw*(1j*w/w0 + w0/(1j*w)))
```

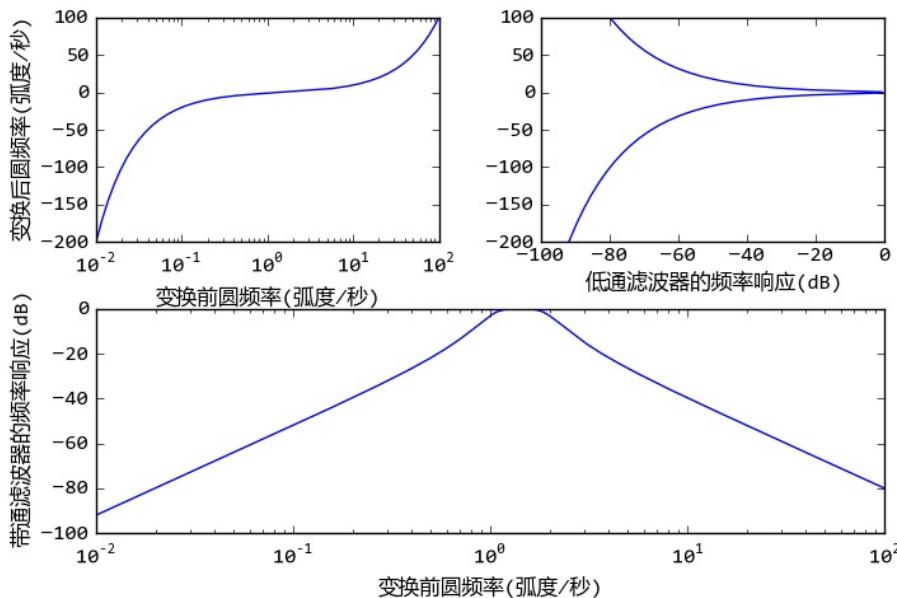
nwworNfreqs

```
_, h = signal.freqs(b, a, worN=nw)  
h = 20*np.log10(np.abs(h))
```

wh

```
pl.semilogx(w, h)
```

(XY)



00nw01nww0

```
>>> print w[np.argmin(np.abs(nw))]
1.4130259906
```

scipy.signal

- lp2lp :
- lp2hp :
- lp2bp :
- lp2bs :

lp2bp(b,a)

```
>>> b, a = signal.butter(2, 1.0, analog=1)
>>> b3, a3 = signal.lp2bp(b,a,np.sqrt(2), 1)
```

butter

```
>>> b4, a4 = signal.butter(2, [1,2], btype='bandpass', analog=1)
```

```
>>> np.all(b3==b4)
True
>>> np.all(a3==a4)
True
```

freqzfreqz!Python

```
>>> import scipy.signal  
>>> signal.freqz??
```

freqz():

```
1 def freqz(b, a=1, worN=None, whole=0, plot=None):  
2     b, a = map(atleast_1d, (b,a))  
3     if whole:  
4         lastpoint = 2*pi  
5     else:  
6         lastpoint = pi  
7     if worN is None:  
8         N = 512  
9         w = numpy.arange(0, lastpoint, lastpoint/N)  
10    elif isinstance(worN, types.IntType):  
11        N = worN  
12        w = numpy.arange(0, lastpoint, lastpoint/N)  
13    else:  
14        w = worN  
15    w = atleast_1d(w)  
16    zm1 = exp(-1j*w)  
17    h = polyval(b[::-1], zm1) / polyval(a[::-1], zm1)  
18    if not plot is None:  
19        plot(w, h)  
20    return w, h
```

3

```
w = numpy.arange(0,pi,pi/N)  
zm1 = exp(-1j*w)  
h = polyval(b[::-1], zm1) / polyval(a[::-1], zm1)
```

3

IIR

$$y[m] = b[0]x[m] + b[1]x[m-1] + \cdots + b[P]x[m-P] \\ - a[1]y[m-1] - a[2]y[m-2] - \cdots - a[Q]y[m-Q]$$

Z

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b[0] + z^{-1}b[1] + z^{-2}b[2] + \cdots + z^{-M}b[P]}{1 + z^{-1}a[1] + z^{-2}a[2] + \cdots + z^{-N}a[Q]}$$

$$\text{zz} \quad z = e^{j\omega} \quad H(\omega) \quad \omega \text{ 02*pi} \quad e^{j\omega} \text{ N}$$

```
w = numpy.arange(0,pi,pi/N)
```

$$w \quad z^{-1} zm1$$

```
zm1 = exp(-1j*w)
```

h

```
h = polyval(b[::-1], zm1) / polyval(a[::-1], zm1)
```

polyval(p, x)xp

```
p[0]*(x**N-1) + p[1]*(x**N-2) + ... + p[N-2]*x + p[N-1]
```

bapolyvalpb[::-1]zm1h

freqzZfreqz

```
1 # -*- coding: utf-8 -*-
2 import numpy as np
3 import pylab as pl
4 import scipy.signal as signal
5
6 def logfreqz(b, a, f0, f1, fs, N):
7     """
```

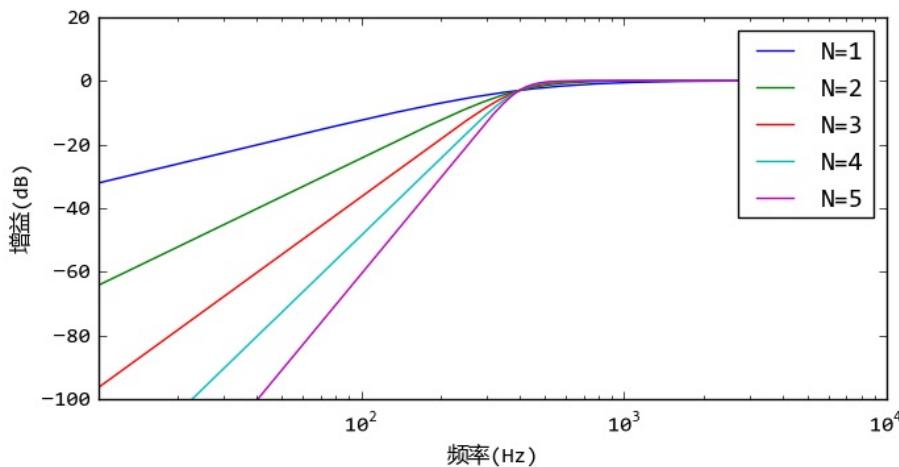
```

8      b, a
9      f0, f1:
10     fs:
11     """
12     w0, w1 = np.log10(f0/fs*2*np.pi), np.log10(f1/fs*2*np.pi)
13     #
14     w = np.logspace(w0, w1, N, endpoint=False)
15     zm1 = np.exp(-1j*w)
16     h = np.polyval(b[::-1], zm1) / np.polyval(a[::-1], zm1)
17     return w/2/np.pi*fs, h
18
19 for n in range(1, 6):
20     # n0.1*4000 = 400Hz
21     b, a = signal.iirfilter(n, [0.1, 1])
22     f, h = logfreqz(b, a, 10.0, 4000.0, 8000.0, 400)
23     gain = 20*np.log10(np.abs(h))
24     pl.semilogx(f, gain, label="N=%s" % n)
25     slope = (gain[100]-gain[10]) / (np.log2(f[100]) - np.log2(f[10]))
26     print "N=%s, slope=%s dB" % (n, slope)
27     pl.ylim(-100, 20)
28     pl.xlabel(u"(Hz)" )
29     pl.ylabel(u"(dB)" )
30     pl.legend()
31     pl.show()

```

logfreqzbaf0f1fsNf/fs*2*pilogsacefreqzpolyval

iirfilter5IIR0.18kHz0.1*4kHz=400Hz5IIR





IIR25

```
N=1, slope=5.9955865774 dB  
N=2, slope=12.0417201051 dB  
N=3, slope=18.0630802032 dB  
N=4, slope=24.0841135443 dB  
N=5, slope=30.1051375912 dB
```

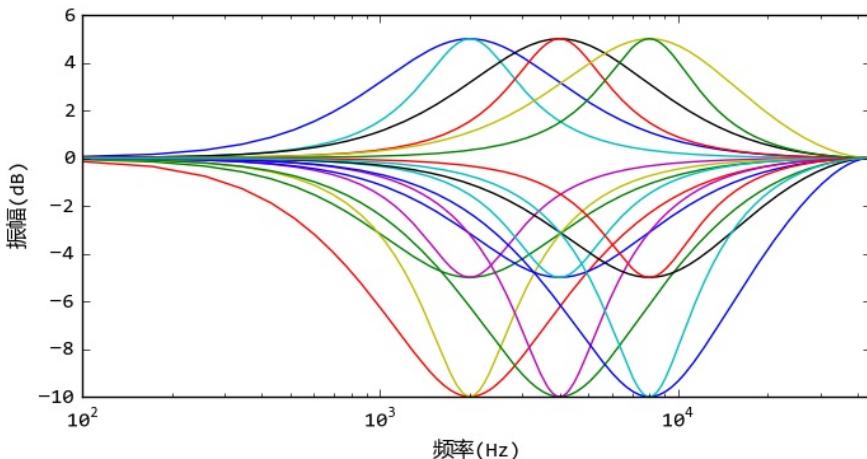
IIR1 6dB/oct (6dB)400Hz-3dB

IIR

IIR

$$H(z) = \frac{b0 + b1 \cdot z^{-1} + b2 \cdot z^{-2}}{a0 + a1 \cdot z^{-1} + a2 \cdot z^{-2}}$$

a01b0, b1, b2, a0, a1, a2a0a0=1 b0, b1, b2, a1, a25



f0peak

- 01
- (pi)1
- f0peak
- f00

4Q55

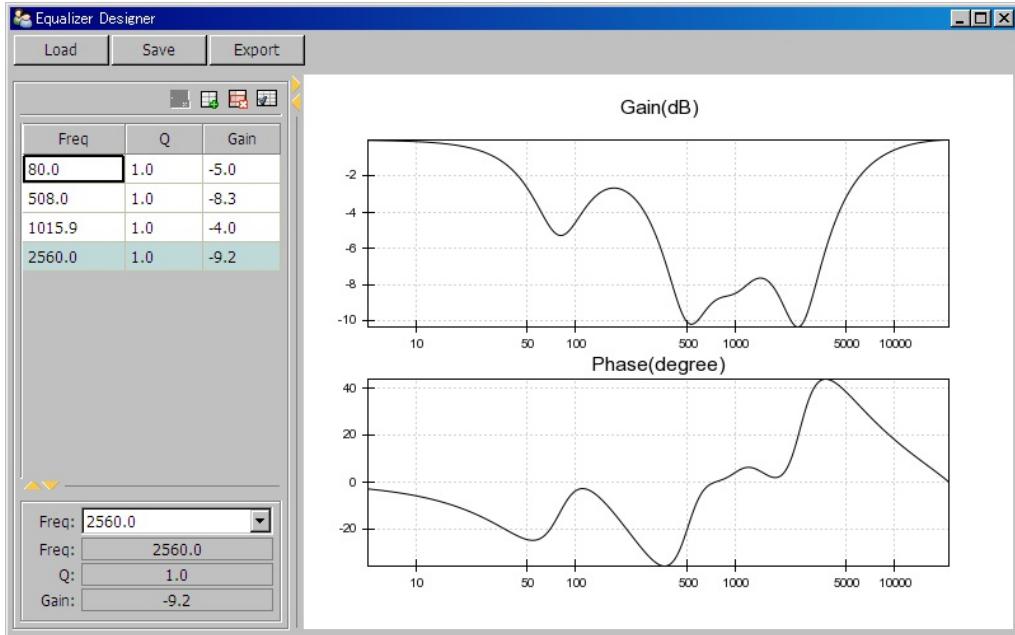
Audio EQ Cookbook (<http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt>)

```

1 # -*- coding: utf-8 -*-
2
3 import scipy.signal as signal
4 import pylab as pl
5 import math
6 import numpy as np
7
8 def design_equalizer(freq, Q, gain, Fs):
9     """
10     A = 10**(gain/40.0)
11     w0 = 2*math.pi*freq/Fs
12     alpha = math.sin(w0) / 2 / Q
13
14     b0 = 1 + alpha * A
15     b1 = -2*math.cos(w0)
16     b2 = 1 - alpha * A
17     a0 = 1 + alpha / A
18     a1 = -2*math.cos(w0)
19     a2 = 1 - alpha / A
20     return [b0/a0,b1/a0,b2/a0], [1.0, a1/a0, a2/a0]
21
22 pl.figure(figsize=(8,4))
23 for freq in [1000, 2000, 4000]:
24     for q in [0.5, 1.0]:
25         for p in [5, -5, -10]:
26             b,a = design_equalizer(freq, q, p, 44100)
27             w, h = signal.freqz(b, a)
28             pl.semilogx(w/np.pi*44100, 20*np.log10(np.abs(h)))
29             pl.xlim(100, 44100)
30             pl.xlabel(u"(Hz)")
31             pl.ylabel(u"(dB)")
32             pl.subplots_adjust(bottom=0.15)
33             pl.show()

```

TraitsUIChaco



ScrubberEditorBUG

ScrubberEditorsite-packages scrubber_editor.py

TraitsBackendWX-3.2.0-py2.6.egg\enthought\traits\ui\wx\scrubber_

```
# Establish the slider increment:
increment = self.factory.increment
if increment <= 0.0:
    if (low is None) or (high is None) or isinstance( low, int ):
        increment = 1.0
    else:
        increment = pow( 10, round( log10( high - low ) / 100.0 ) )
self.increment = increment # ** if
```

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

FFT

numpytraitsUIChacoFFTFFTFFTFFT

FFT

FFTN264, 128, 256 NFFTN

- ON/20
- iN-i

rand8xfft8

```
1 >>> x = np.random.rand(8)
2 >>> x
3 array([ 0.15562099,  0.56862756,  0.54371949,  0.06354358,
4          0.78360968,  0.90116887,  0.1588846 ])
5 >>> xf = np.fft.fft(x)
6 >>> xf
7 array([ 3.78195634+0.j           , -0.53575962+0.57688097j,
8          -0.68248579-1.12980906j, -0.36656155-0.13801778j,
9          0.63262552+0.j           , -0.36656155+0.13801778j,
10         -0.68248579+1.12980906j, -0.53575962-0.57688097j])
```

FFTIFFTFFT

```
>>> np.fft.ifft(xf)
array([ 0.15562099 +0.00000000e+00j,  0.56862756 +1.91940002e-16j
       0.54371949 +1.24900090e-16j,  0.06354358 -2.33573365e-16j
       0.60678158 +0.00000000e+00j,  0.78360968 +2.75206729e-16j
       0.90116887 -1.24900090e-16j,  0.15888460 -2.33573365e-16j]
```

ifftx

FFTIFFT8xf8

FFT

0ON/2N/2+1 ON/20N

FFT

- 0
- $ia+b^*jN/i \cos b \sin$

FFT

```
>>> x = np.ones(8)
>>> x
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
>>> np.fft.fft(x)/len(x)
array([ 1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
       0.+0.j])
```

1xFFT000(FFTFFT)1

8FFT

```
1 >>> x = np.arange(0, 2*np.pi, 2*np.pi/8)
2 >>> y = np.sin(x)
3 >>> np.fft.fft(y)/len(y)
4 array([- 1.42979161e-18 +0.00000000e+00j,
5        -4.44089210e-16 -5.00000000e-01j,
6        1.53075794e-17 -1.38777878e-17j,
7        3.87737802e-17 -1.11022302e-16j,
8        2.91853672e-17 +0.00000000e+00j,
9        0.00000000e+00 -9.71445147e-17j,
10       1.53075794e-17 +1.38777878e-17j,     3.44085112e-16
```

linspacex

$80-2\pi/8$ np.linspace(0, 2π , 8)

```
>>> np.linspace(0, 2*np.pi, 8)
array([ 0.          ,  0.8975979 ,  1.7951958 ,  2.6927937 ,
       3.5903916 ,  4.48798951,  5.38558741,  6.28318531])
>>> 2*np.pi / 0.8975979
7.000000079986666
```

```
linspace7np.linspace 9endpoint=False
```

```
>>> np.linspace(0, 2*np.pi, 9, endpoint=False)
array([ 0.          ,  0.6981317 ,  1.3962634 ,  2.0943951 ,  2.791
       3.4906585 ,  4.1887902 ,  4.88692191,  5.58505361])
```

FFT1-0.5()1-0.5*-2=1:

```
>>> np.fft.fft(np.cos(x))/len(x)
array([-4.30631550e-17 +0.00000000e+00j,
       5.00000000e-01 -2.52659764e-16j,
       1.53075794e-17 +0.00000000e+00j,
       1.11022302e-16 +1.97148613e-16j,
       1.24479962e-17 +0.00000000e+00j,
      -1.11022302e-16 +1.91429446e-16j,
       1.53075794e-17 +0.00000000e+00j,     5.00000000e-01 -1.359
```

10.510.5*2=12

```
1 >>> np.fft.fft(2*np.sin(2*x))/len(x)
2 array([ 6.12303177e-17 +0.00000000e+00j,
3           6.12303177e-17 +6.12303177e-17j,
4           -1.83690953e-16 -1.00000000e+00j,
5           6.12303177e-17 -6.12303177e-17j,
6           6.12303177e-17 +0.00000000e+00j,
7           6.12303177e-17 +6.12303177e-17j,
8           -1.83690953e-16 +1.00000000e+00j,    6.12303177e-17
9 >>> np.fft.fft(0.8*np.cos(2*x))/len(x)
10 array([-2.44921271e-17 +0.00000000e+00j,
11           -3.46370983e-17 +2.46519033e-32j,
12           4.00000000e-01 -9.79685083e-17j,
13           3.46370983e-17 -3.08148791e-32j,
14           2.44921271e-17 +0.00000000e+00j,
15           3.46370983e-17 -2.46519033e-32j,
16           4.00000000e-01 +9.79685083e-17j, -3.46370983e-17
```

4(N/2)FFT22,-10.80.4

:

•
•

```
1 >>> x = np.arange(0, 2*np.pi, 2*np.pi/128)
2 >>> y = 0.3*np.cos(x) + 0.5*np.cos(2*x+np.pi/4) + 0.8*np.co
3 >>> yf = np.fft.fft(y)/len(y)
4 >>> yf[:4]
5 array([ 1.00830802e-17 +0.00000000e+00j,
6          1.50000000e-01 +6.27820821e-18j,
7          1.76776695e-01 +1.76776695e-01j,   2.00000000e-01
8 >>> np.angle(yf[1])
9 4.1854721366992471e-017
10 >>> np.abs(yf[1]), np.rad2deg(np.angle(yf[1]))
11 (0.1500000000000008, 2.3980988870246962e-015)
12 >>> np.abs(yf[2]), np.rad2deg(np.angle(yf[2]))
13 (0.2500000000000011, 44.99999999999993)
14 >>> np.abs(yf[3]), np.rad2deg(np.angle(yf[3]))
15 (0.3999999999999991, -60.000000000000085)
```

- 128/1.00 0.3
- 64/2.045 0.5
- 128/3.0-600.8

yf[1], yf[2], yf[3]FFT

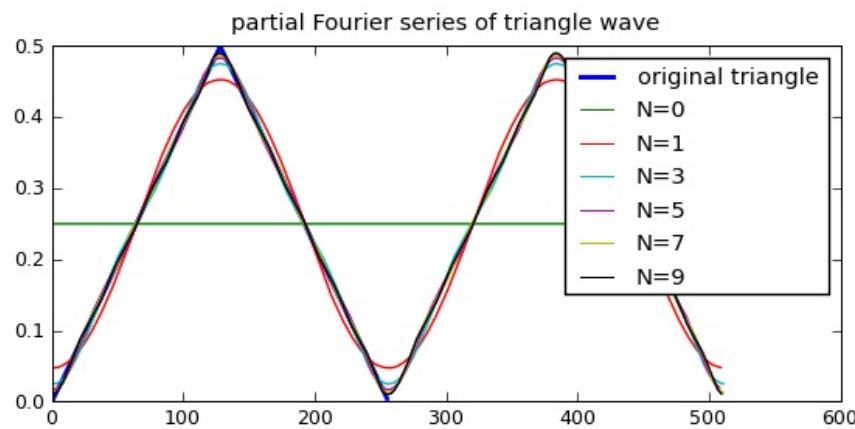
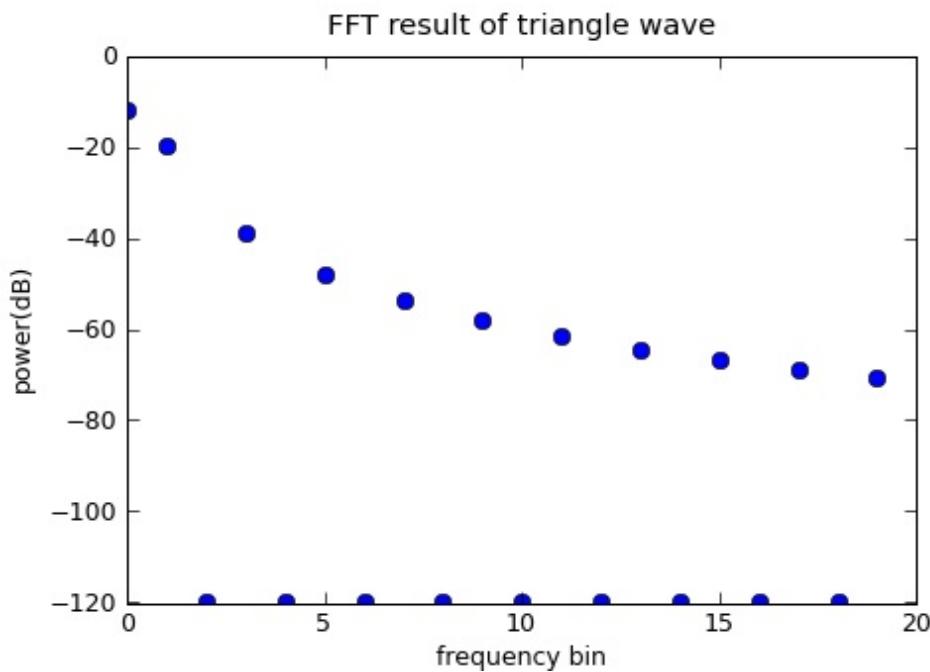
ifft:

```
1 # -*- coding: utf-8 -*-
2 #
3 import numpy as np
4 import pylab as pl
5
6 # FFTfreqsnloops
7 def fft_combine(freqs, n, loops=1):
8     length = len(freqs) * loops
9     data = np.zeros(length)
10    index = loops * np.arange(0, length, 1.0) / length * (2 * np.pi)
11    for k, p in enumerate(freqs[:n]):
12        if k != 0: p *= 2 # *2
13        data += np.real(p) * np.cos(k*index) #
14        data -= np.imag(p) * np.sin(k*index) #
15    return index, data
16
17 # size1
18 def triangle_wave(size):
19     x = np.arange(0, 1, 1.0/size)
20     y = np.where(x<0.5, x, 0)
21     y = np.where(x>=0.5, 1-x, y)
22     return x, y
23
24 fft_size = 256
25
26 # FFT
27 x, y = triangle_wave(fft_size)
28 fy = np.fft.fft(y) / fft_size
29
30 # FFT200
31 # lognp.clip
32 pl.figure()
33 pl.plot(np.clip(20*np.log10(np.abs(fy[:20])), -120, 120), 'r')
34 pl.xlabel("frequency bin")
35 pl.ylabel("power(dB)")
36 pl.title("FFT result of triangle wave")
37
38 # x
39 pl.figure()
```

```

40 pl.plot(y, label="original triangle", linewidth=2)
41 for i in [0,1,3,5,7,9]:
42     index, data = fft_combine(fy, i+1, 2) #
43     pl.plot(data, label = "N=%s" % i)
44 pl.legend()
45 pl.title("partial Fourier series of triangle wave")
46 pl.show()

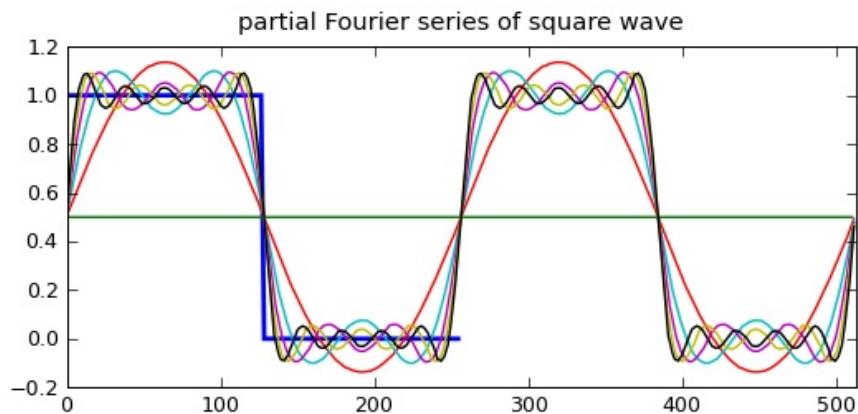
```



18triangle_wavenp.wheretrianglexyxx

7fft_combinefftfreqsnloops

```
def square_wave(size):
    x = np.arange(0, 1, 1.0/size)
    y = np.where(x<0.5, 1.0, 0)
    return x, y
```



FFT

TraitsUI200:

:

- 16ScrubberEditor77-80Itemeditor = scrubbertrait
editorRangetrait
- Range traitstrait(trait)trait(36, 37)trait

```
low = Float(0.02)
hi = Float(1.0)
```

- 190triangle_funcfrompyfuncufunc150frompyfuncufunc

(dtype)objectcast["float64"]float64

- trait: _trait_changedtrait@on_trait_change

:

```
1  # -*- coding: utf-8 -*-
2  from enthought.traits.api import \
3      Str, Float, HasTraits, Property, cached_property, Range
4
5  from enthought.chaco.api import Plot, AbstractPlotData, Array
6
7  from enthought.traits.ui.api import \
8      Item, View, VGroup, HSplit, ScrubberEditor, VSplit
9
10 from enthought.enable.api import Component, ComponentEditor
11 from enthought.chaco.tools.api import PanTool, ZoomTool
12
13 import numpy as np
14
15 #
16 scrubber = ScrubberEditor(
17     hover_color = 0xFFFFFFFF,
18     active_color = 0xA0CD9E,
19     border_color = 0x808080
20 )
21
22 # FFTfreqsnloops
23 def fft_combine(freqs, n, loops=1):
24     length = len(freqs) * loops
25     data = np.zeros(length)
26     index = loops * np.arange(0, length, 1.0) / length * (n - 1)
27     for k, p in enumerate(freqs[:n]):
28         if k != 0: p *= 2 # *2
29         data += np.real(p) * np.cos(k*index) #
30         data -= np.imag(p) * np.sin(k*index) #
31     return index, data
32
33 class TriangleWave(HasTraits):
34     # Rangetriplets
35     # trait
36     low = Float(0.02)
37     hi = Float(1.0)
38
```

```

39      #
40      wave_width = Range("low", "hi", 0.5)
41
42      # Cx
43      length_c = Range("low", "wave_width", 0.5)
44
45      # y
46      height_c = Float(1.0)
47
48      # FFTEnum
49      fftsize = Enum([(2**x) for x in range(6, 12)])
50
51      # FFTX
52      fft_graph_up_limit = Range(0, 400, 20)
53
54      # FFT
55      peak_list = Str
56
57      #
58      N = Range(1, 40, 4)
59
60      #
61      plot_data = Instance(AbstractPlotData)
62
63      #
64      plot_wave = Instance(Component)
65
66      # FFT
67      plot_fft = Instance(Component)
68
69      #
70      container = Instance(Component)
71
72      #
73      view = View(
74          HSplit(
75              VSplit(
76                  VGroup(
77                      Item("wave_width", editor = scrubber,
78                      Item("length_c", editor = scrubber, la
79                      Item("height_c", editor = scrubber, la
80                      Item("fft_graph_up_limit", editor = sc
81                      Item("fftsize", label=u"FFT" ),
82                      Item("N", label=u"" )
83                  ),
84                  Item("peak_list", style="custom", show_la
85          ),

```

```

86         VGroup(
87             Item("container", editor=ComponentEditor(s
88                 orientation = "vertical"
89             )
90         ),
91         resizable = True,
92         width = 800,
93         height = 600,
94         title = u"FFT"
95     )
96
97     #
98     #
99     def _create_plot(self, data, name, type="line"):
100        p = Plot(self.plot_data)
101        p.plot(data, name=name, title=name, type=type)
102        p.tools.append(PanTool(p))
103        zoom = ZoomTool(component=p, tool_mode="box", always_on=True)
104        p.overlays.append(zoom)
105        p.title = name
106        return p
107
108    def __init__(self):
109        #
110        super(TriangleWave, self).__init__()
111
112        # Plot
113        self.plot_data = ArrayPlotData(x=[], y=[], f=[])
114
115        #
116        self.container = VPlotContainer()
117
118        # (x,y)(x2,y2)
119        self.plot_wave = self._create_plot(("x","y"), "Triangle Wave")
120        self.plot_wave.plot(("x2","y2"), color="red")
121
122        # fp
123        self.plot_fft = self._create_plot(("f","p"), "FFT")
124
125        #
126        self.container.add( self.plot_wave )
127        self.container.add( self.plot_fft )
128
129        #
130        self.plot_wave.x_axis.title = "Samples"
131        self.plot_fft.x_axis.title = "Frequency pins"
132        self.plot_fft.y_axis.title = "(dB)"

```

```

133
134     # fftsize1024Enum
135     self.fftsize = 1024
136
137     # FFTx
138     def _fft_graph_up_limit_changed(self):
139         self.plot_fft.x_axis.mapper.range.high = self.fft_
140
141     def _N_changed(self):
142         self.plot_sin_combine()
143
144     # trait@on_trait_change
145     @on_trait_change("wave_width, length_c, height_c, ffts")
146     def update_plot(self):
147         #
148         global y_data
149         x_data = np.arange(0, 1.0, 1.0/self.fftsize)
150         func = self.triangle_func()
151         # funcfloat64
152         y_data = np.cast["float64"](func(x_data))
153
154         #
155         fft_parameters = np.fft.fft(y_data) / len(y_data)
156
157         #
158         fft_data = np.clip(20*np.log10(np.abs(fft_parameters)))
159
160         #
161         self.plot_data.set_data("x", np.arange(0, self.fftsize))
162         self.plot_data.set_data("y", y_data)
163         self.plot_data.set_data("f", np.arange(0, len(fft_data)))
164         self.plot_data.set_data("p", fft_data)
165
166         # x2
167         self.plot_data.set_data("x2", np.arange(0, 2*self.fftsize))
168
169         # x
170         self._fft_graph_up_limit_changed()
171
172         # -80dB
173         peak_index = (fft_data > -80)
174         peak_value = fft_data[peak_index][:20]
175         result = []
176         for f, v in zip(np.flatnonzero(peak_index), peak_value):
177             result.append("%s : %s" %(f, v) )
178         self.peak_list = "\n".join(result)
179

```

```
180         # fft
181         self.fft_parameters = fft_parameters
182         self.plot_sin_combine()
183
184     # 2
185     def plot_sin_combine(self):
186         index, data = fft_combine(self.fft_parameters, self)
187         self.plot_data.set_data("y2", data)
188
189     # ufunc
190     def triangle_func(self):
191         c = self.wave_width
192         c0 = self.length_c
193         hc = self.height_c
194
195         def trifunc(x):
196             x = x - int(x) # 1x
197             if x >= c: r = 0.0
198             elif x < c0: r = x / c0 * hc
199             else: r = (c-x) / (c-c0) * hc
200             return r
201
202     # trifuncufunc,
203     # Object
204     return np.frompyfunc(trifunc, 1, 1)
205
206 if __name__ == "__main__":
207     triangle = TriangleWave()
208     triangle.configure_traits()
```

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

FFT()IFFT()

FFT

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as pl

sampling_rate = 8000
fft_size = 512
t = np.arange(0, 1.0, 1.0/sampling_rate)
x = np.sin(2*np.pi*156.25*t) + 2*np.sin(2*np.pi*234.375*t)
xs = x[:fft_size]
xf = np.fft.rfft(xs)/fft_size
freqs = np.linspace(0, sampling_rate/2, fft_size/2+1)
xfp = 20*np.log10(np.clip(np.abs(xf), 1e-20, 1e100))
pl.figure(figsize=(8,4))
pl.subplot(211)
pl.plot(t[:fft_size], xs)
pl.xlabel(u"() ")
pl.title(u"156.25Hz234.375Hz")
pl.subplot(212)
pl.plot(freqs, xfp)
pl.xlabel(u"(Hz) ")
pl.subplots_adjust(hspace=0.4)
pl.show()
```

sampling_rate, fft_sizeFFT

np.arange1t1/sampline_rate

```
t = np.arange(0, 1.0, 1.0/sampling_rate)
```

t156.25Hz234.375Hz

```
x = np.sin(2*np.pi*156.25*t) + 2*np.sin(2*np.pi*234.375*t)
```

512FFT

NFFT

fs, NFFT
NFFT: $n * fs / N$
 $8000 / 512.0 = 15.625 \text{ Hz}$
 156.25 Hz
 234.375 Hz
 1015

xfft_size
fft(np.fft.rfft(FFTFFTrfftfft_size)

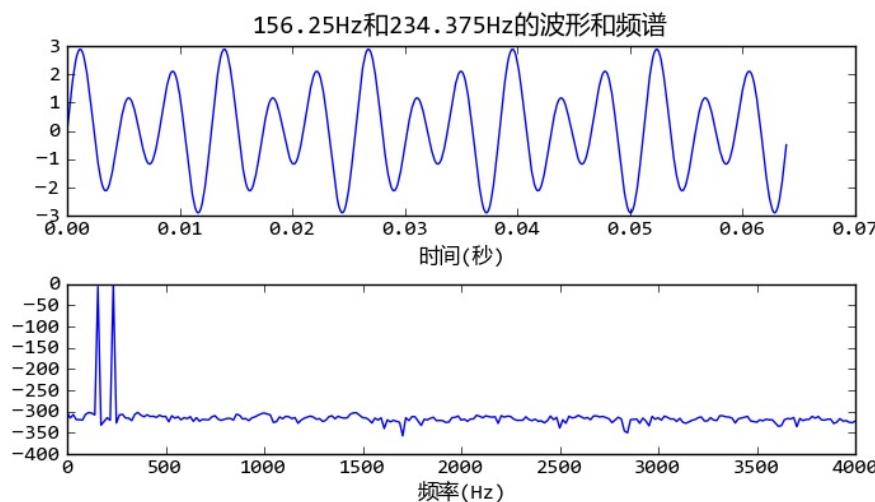
```
xs = x[:fft_size]
xf = np.fft.rfft(xs)/fft_size
```

rfftN/2+10(Hz)sampling_rate/2(Hz)N/2+1np.linspace

```
freqs = np.linspace(0, sampling_rate/2, fft_size/2+1)
```

20*np.log10() db0log10np.clipxf

```
xfp = 20*np.log10(np.clip(np.abs(xf), 1e-20, 1e100))
```



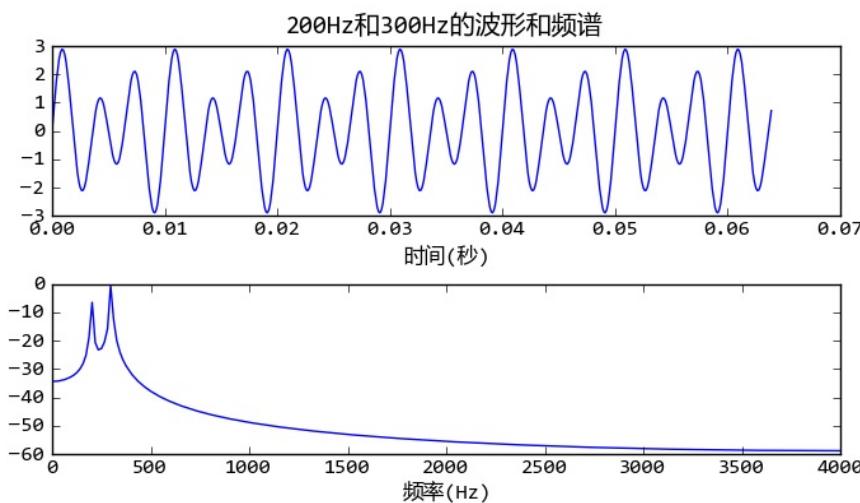
FFT

```
>>> xfp[10]
-6.0205999132796251
>>> xfp[15]
-9.6432746655328714e-16
```

156.25Hz-6dB 234.375Hz0dB(/2)

fft_size

```
x = np.sin(2*np.pi*200*t) + 2*np.sin(2*np.pi*300*t)
```



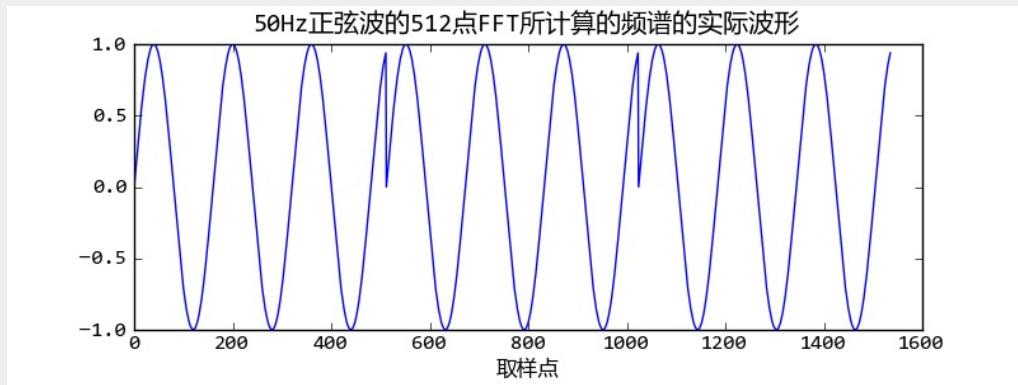
FFT

200Hz300Hzfft_size200Hz300Hz

512FFT512FFT51251250Hz512FFT

```
>>> t = np.arange(0, 1.0, 1.0/8000)
```

```
>>> x = np.sin(2*np.pi*50*t)[:512]
>>> pl.plot(np.hstack([x,x,x]))
>>> pl.show()
```



50Hz512FFT

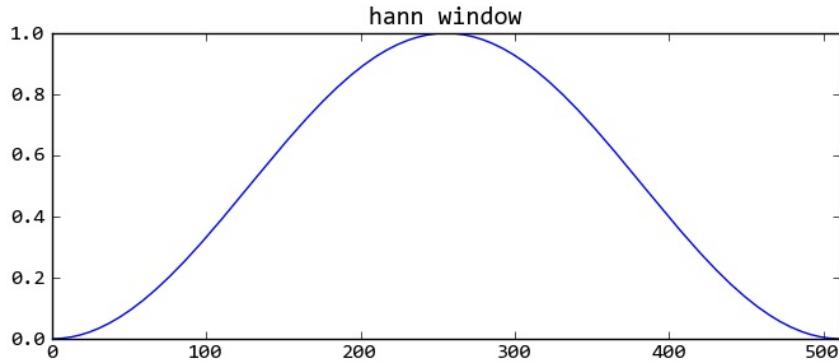
FFT

FFThann

$$w(n) = 0.5 \left(1 - \cos \left(\frac{2\pi n}{N-1} \right) \right)$$

N512hann

```
>>> import pylab as pl
>>> import scipy.signal as signal
>>> pl.figure(figsize=(8,3))
>>> pl.plot(signal.hann(512))
```



`hann`

`scipy.signal.Nhann051100FFT0`

100010

```
>>> np.sin(np.arange(0, 2*np.pi, 2*np.pi/10))
array([ 0.00000000e+00,   5.87785252e-01,   9.51056516e-01,
       9.51056516e-01,   5.87785252e-01,   1.22464680e-16,
      -5.87785252e-01,  -9.51056516e-01,  -9.51056516e-01,
      -5.87785252e-01])
>>> np.sin(np.linspace(0, 2*np.pi, 10))
array([ 0.00000000e+00,   6.42787610e-01,   9.84807753e-01,
       8.66025404e-01,   3.42020143e-01,  -3.42020143e-01,
      -8.66025404e-01,  -9.84807753e-01,  -6.42787610e-01,
      -2.44929360e-16])
```

0hannsym0N+1hannN

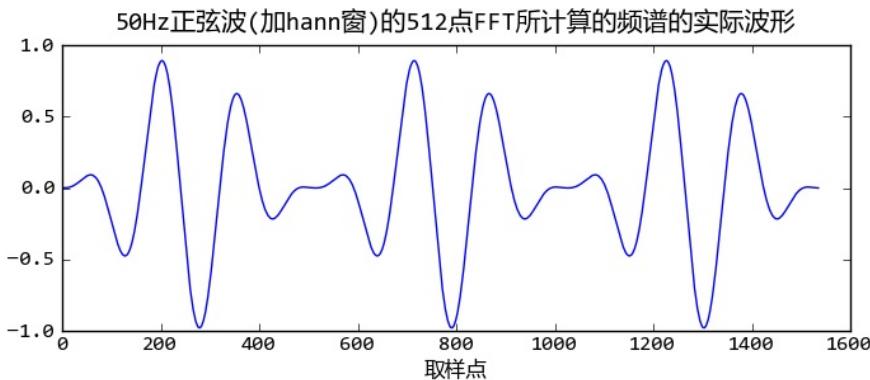
```
>>> signal.hann(8)
array([ 0.           ,  0.1882551 ,  0.61126047,  0.95048443,  0.95048443,
       0.61126047,  0.1882551 ,  0.           ])
>>> signal.hann(8, sym=0)
array([ 0.           ,  0.14644661,  0.5           ,  0.85355339,  1.
       0.85355339,  0.5           ,  0.14644661])
```

50Hz

```

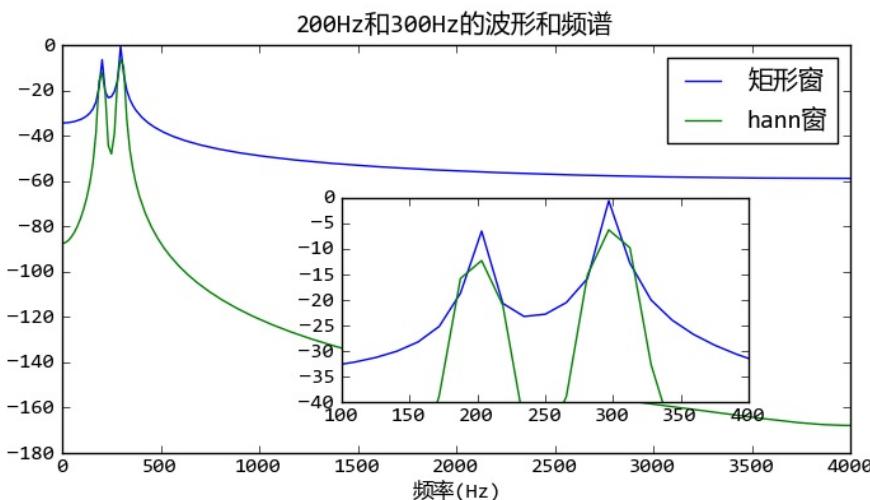
>>> t = np.arange(0, 1.0, 1.0/8000)
>>> x = np.sin(2*np.pi*50*t)[:512] * signal.hann(512, sym=0)
>>> pl.plot(np.hstack([x,x,x]))
>>> pl.show()

```



hann50Hz512FFT

200Hz, 300Hzhann



hannhann

hann200Hz300Hzhann

```

>>> np.sum(signal.hann(512, sym=0))/512
0.5

```

hann2

hann

FFT

```
import numpy as np
import scipy.signal as signal
import pylab as pl

def average_fft(x, fft_size):
    n = len(x) // fft_size * fft_size
    tmp = x[:n].reshape(-1, fft_size)
    tmp *= signal.hann(fft_size, sym=0)
    xf = np.abs(np.fft.rfft(tmp)/fft_size)
    avgf = np.average(xf, axis=0)
    return 20*np.log10(avgf)
```

average_fft(x, fft_size)
fft_sizeFFTdBxfft_size
reshapetmp1fft_size

```
n = len(x) // fft_size * fft_size
tmp = x[:n].reshape(-1, fft_size)
```

tmp1hann

```
tmp *= signal.hann(fft_size, sym=0)
```

rfffttmpFFT

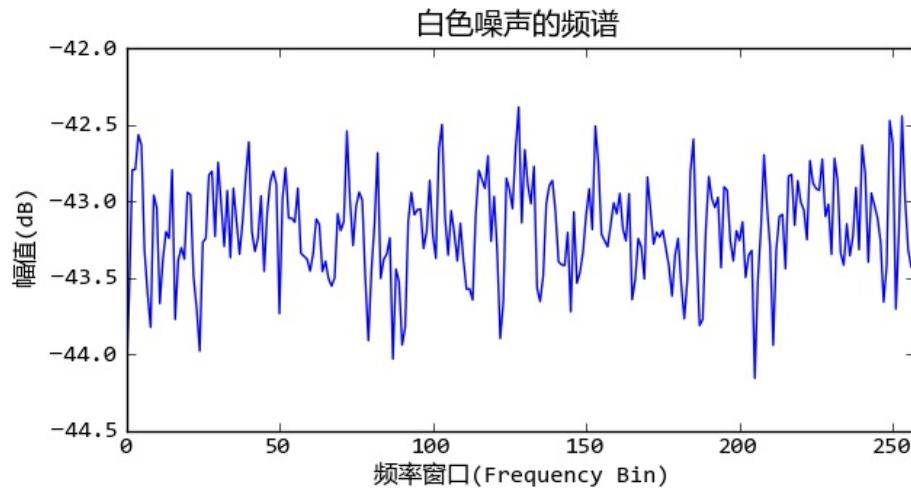
```
xf = np.abs(np.fft.rfft(tmp)/fft_size)
```

averagexf0

```
avgf = np.average(xf, axis=0)
```

averagge_fft

```
>>> x = np.random.rand(100000) - 0.5
>>> xf = average_fft(x, 512)
>>> pl.plot(xf)
>>> pl.show()
```



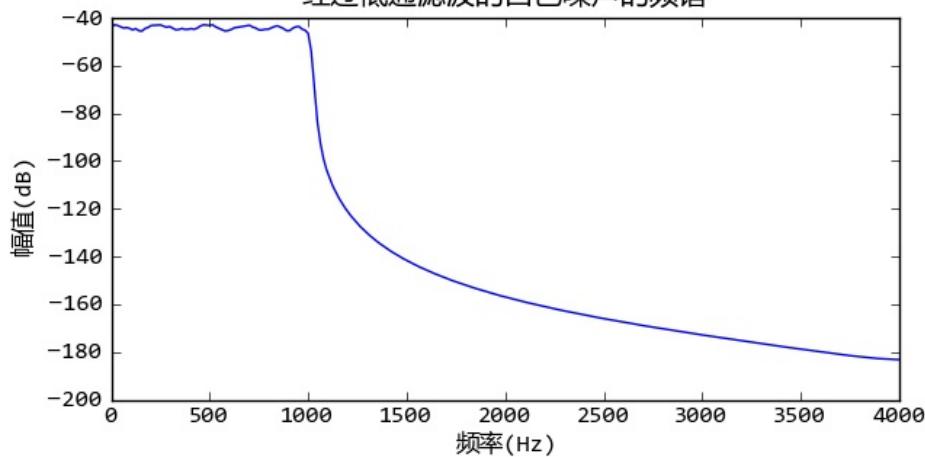
(Y)

scipy.signal.IIRII*iirdesign*8kHz1kHz Chebyshev I
iirdesign(0-1)*filtfilt*x

```
>>> b,a=signal.iirdesign(1000/4000.0, 1100/4000.0, 1, -40, 0, "ch"
>>> x = np.random.rand(100000) - 0.5
>>> y = signal.filtfilt(b, a, x)
```

average_ffty

经过低通滤波的白色噪声的频谱



xhyxhxhh

FFT O(N*log(N))FFT O(N*log(N)) O(N*N)FFT
N

FFTFFT

abab128 len(a) + len(b) - 1 = 257FFTab256

```
# -*- coding: utf-8 -*-
import numpy as np

def fft_convolve(a,b):
    n = len(a)+len(b)-1
    N = 2**int(np.log2(n))+1
    A = np.fft.fft(a, N)
    B = np.fft.fft(b, N)
    return np.fft.ifft(A*B)[:n]

if __name__ == "__main__":
    a = np.random.rand(128)
    b = np.random.rand(128)
    c = np.convolve(a,b)

    print np.sum(np.abs(c - fft_convolve(a,b)))
```

FFT5e-12

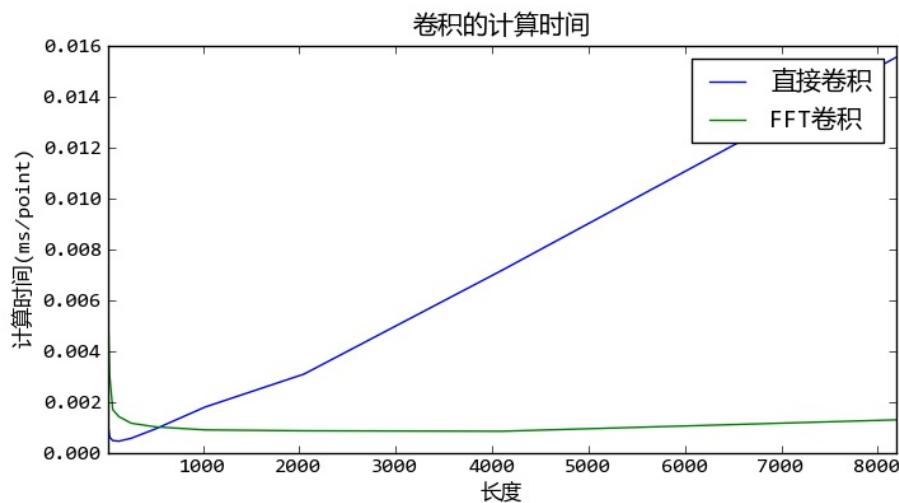
a,b128cn=255n2

```
N = 2**int(np.log2(n))+1
```

fftN(FFT)ffta,bifftc2Nc0

```
>>> import timeit  
>>> setup="""import numpy as np  
a=np.random.rand(10000)  
b=np.random.rand(10000)  
from spectrum_fft_convolve import fft_convolve"""  
>>> timeit.timeit("np.convolve(a,b)",setup, number=10)  
1.852900578146091  
>>> timeit.timeit("fft_convolve(a,b)",setup, number=10)  
0.19475575806416145
```

FFT



YO(N*N)/N1024

FFT

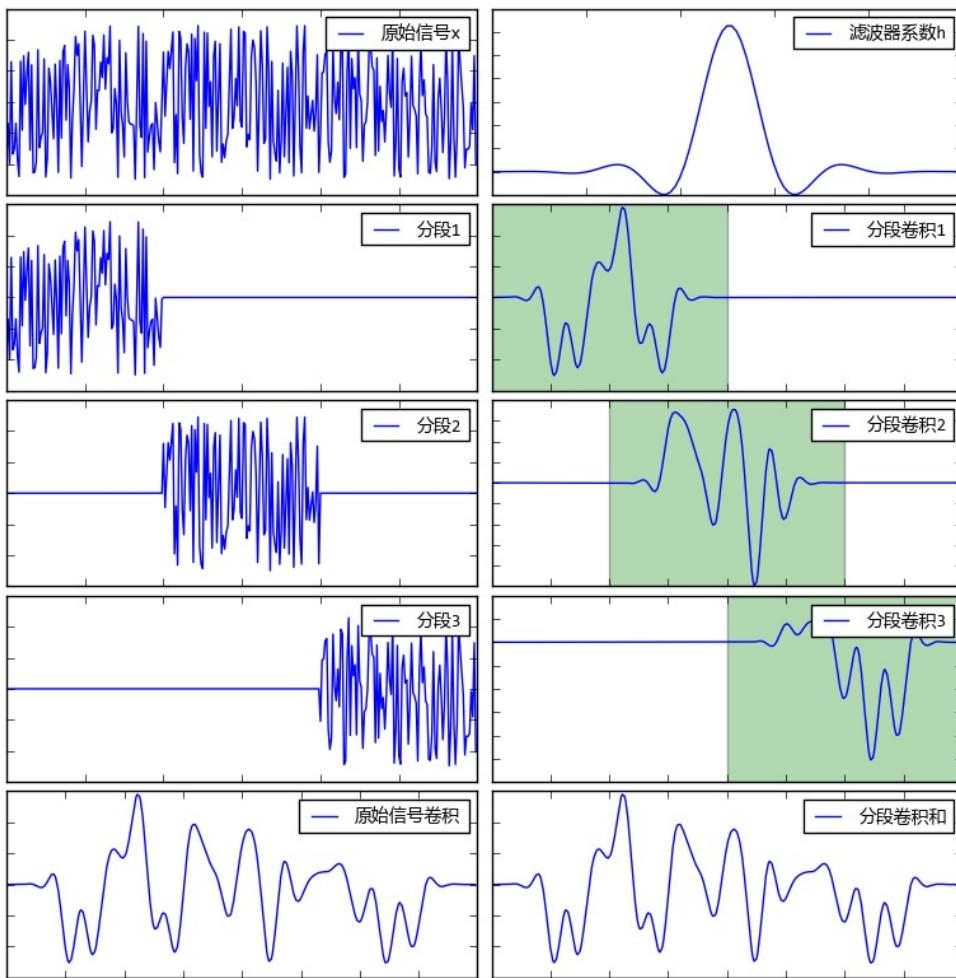
FFTscipy.signalfftconvolveFFT

xhx8kHz1x480000xhFIR

FFT x overlap-add

overlap-add

分段卷积演示



x300hh1012002003

xhhM

1. N+M-10
2. xNhN+M-1N
3. NN0N0
4. 2

```
# -*- coding: utf-8 -*-
import numpy as np
```

```

x = np.random.rand(1000)
h = np.random.rand(101)
y = np.convolve(x, h)

N = 50 #
M = len(h) #

output = []

#0
buffer = np.zeros(M+N-1, dtype=np.float64)

for i in xrange(len(x)/N):
    #N
    xslice = x[i*N:(i+1)*N]
    #
    yslice = np.convolve(xslice, h)
    #
    buffer += yslice
    #Ncopybuffer
    output.append( buffer[:N].copy() )
    #N
    buffer[0:M-1] = buffer[N:]
    #0
    buffer[M-1:] = 0

#
y2 = np.hstack(output)
#
print np.sum(np.abs( y2 - y[:len(x)] ) )

```

23Nbuffer

FFToverlap-addFFT20482048

Hilbert

Hilbert90

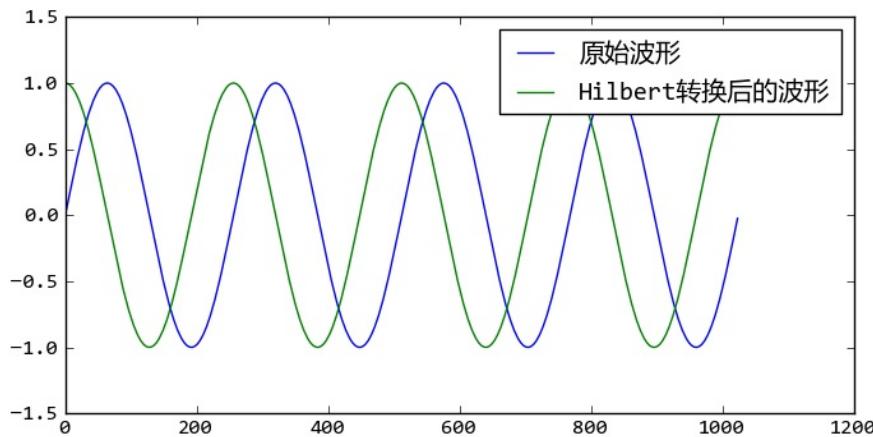
```
# -*- coding: utf-8 -*-
from scipy import fftpack
import numpy as np
import matplotlib.pyplot as pl

# 10244
t = np.linspace(0, 8*np.pi, 1024, endpoint=False)
x = np.sin(t)

# Hilbert
y = fftpack.hilbert(x)
pl.plot(x, label=u"")
pl.plot(y, label=u"Hilbert")
pl.legend()
pl.show()
```

- hilbertscipy.fftpack
- linspaceendpoint=False8*np.pi

hilbert





Hilbert

+90Hilbert+90Hilbert-90Hilbert

90 j Hilbert

$$H(\omega) = j \cdot sgn(\omega)$$

ω sgn

$$sgn(\omega) = \begin{cases} 1, & \text{for } \omega > 0, \\ 0, & \text{for } \omega = 0, \\ -1, & \text{for } \omega < 0, \end{cases}$$

- 0
- +90
- -90

HilbertHilbert

```
>>> x = np.random.rand(16)
>>> y = fftpack.hilbert(x)
>>> X = np.fft.fft(x)
>>> Y = np.fft.fft(y)
>>> np.imag(Y/X)
array([ 0.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
       0., -1., -1., -1., -1., -1., -1., -1.])
```

NFFT

- 0
- N/2/2
- 1N/2-1

- $N/2+1N$

$Y/XHilbertnp.real(Y/X)0$

Hilbert

$$envelope = \sqrt{H(x)^2 + x^2}$$

$xH(x)xHilbertenvelopexxH(x)$

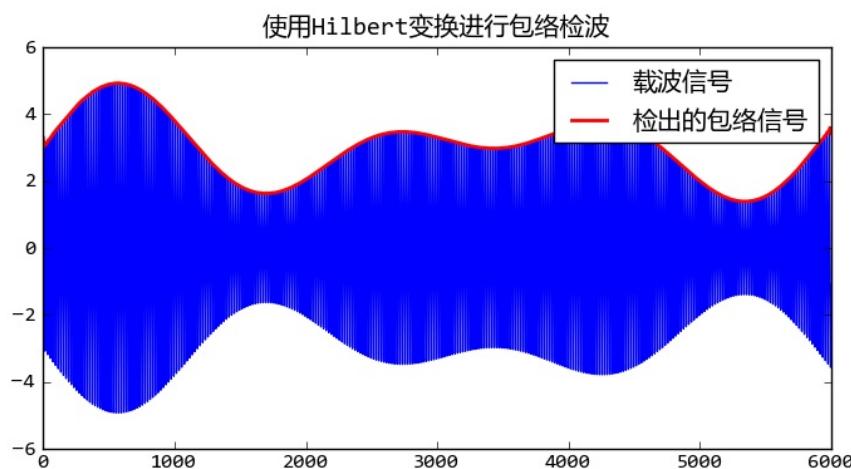
$$\sin^2(t) + \cos^2(t) = 1$$

envelope1sin(t)

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as pl
from scipy import fftpack

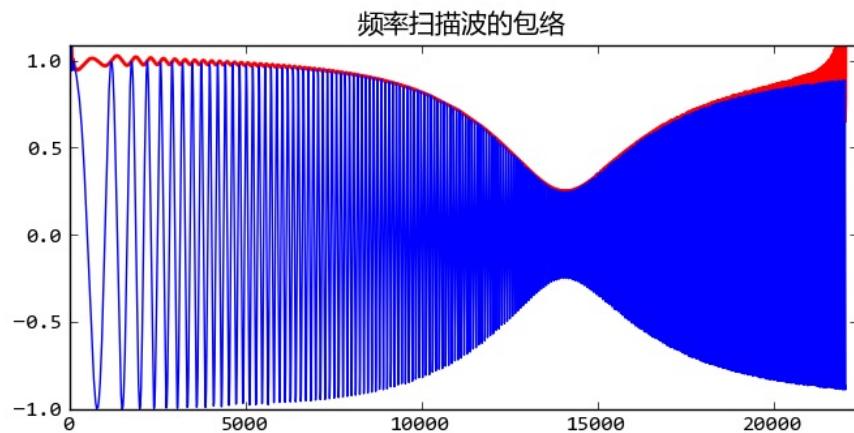
t = np.arange(0, 0.3, 1/20000.0)
x = np.sin(2*np.pi*1000*t) * (np.sin(2*np.pi*10*t) + np.sin(2*np.pi*100*t))
hx = fftpack.hilbert(x)

pl.plot(x, label=u"载波信号")
pl.plot(np.sqrt(x**2 + hx**2), "r", linewidth=2, label=u"检出的包络信号")
pl.title(u"Hilbert")
pl.legend()
pl.show()
```



 Hilbert

```
>>> run filter_lfilter_example01.py #
>>> hy = fftpack.hilbert(y)
>>> pl.plot( np.sqrt(y**2 + hy**2), "r", linewidth=2)
>>> pl.plot(y)
>>> pl.title(u"")
>>> pl.show()
```



 Hilbert

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

CtypesNumPy

ctypes

CtypesPythonWindowsCDLL

PythonCtypesCtypesPythonC

ctypesDLL

CTypes

-
-
- Pythonctypes
- ctypes

ctypes

numpyctypes

```
numpyctypeslib.load_libraryctypes
```

```
test_sum.dllmysum:
```

```
double mysum(double a[], long n)
{
    double sum = 0;
    int i;
    for(i=0;i<n;i++) sum += a[i];
    return sum;
}
```

```
>>> from ctypes import *
>>> sum_test = np.ctypeslib.load_library("sum_test", ".")
>>> print sum_test.mysum
<_FuncPtr object at 0x037D7210>
```

```
sumsum
```

```
>>> sum_test.mysum.argtypes = [POINTER(c_double), c_long]
>>> sum_test.mysum.restype = c_double
```

```
sum
```

```
>>> x = np.arange(1, 101, 1.0)
>>> sum_test.mysum(x.ctypes.data_as(POINTER(c_double)), len(x))
5050.0
```

```
sumPythonmysumCmysum:
```

```
def mysum(x):
    return sum_test.mysum(x.ctypes.data_as(POINTER(c_double)), len(x))
```

```
test_sum.mysumctypesPOINTER(c_double)mysumdouble
```

```
xx.ctypes.data_asxdouble
```

```
mysum
```

```
>>> x = np.arange(1,11,1.0)
>>> mysum(x[::2])
15.0
>>> sum(x[::2])
25.0
```

x[::2]xx[::2]16byptes(2double)mysumx5
1+2+3+4+5=151+3+5+7+9=25

numpyndpointerndpointerrestypeargtypes4

- **dtype** :
- **ndim** :
- **shape** :
- **flags** :

```
test_sum.mysum.argtypes = [
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=1, flags="C_CONTIGUOUS",
                           c_long
]
```

```
sumfuncdoubleC
```

```
mysumPython
```

```
>>> sum_test.mysum(x,len(x))
55.0
>>> sum_test.mysum(x[::2],len(x)/2)
ArgumentError: argument 1: <type 'exceptions.TypeError'>:
array must have flags ['C_CONTIGUOUS']
```

```
mysumC
```

shapestridesmysum2C

```
double mysum2(double a[], int strides[], int shapes[])
{
    double sum = 0;
    int i, j, M, N, S0, S1;
    M = shape[0]; N=shape[1];
    S0 = strides[0] / sizeof(double);
    S1 = strides[1] / sizeof(double);

    for(i=0;i<M;i++){
        for(j=0;j<N;j++){
            sum += a[i*S0 + j*S1];
        }
    }
    return sum;
}
```

mysum23a[]astrides(byte)dims

stridesbytessizeof(double)doubleS0S1aija[i*S0 + j*S1]
ctypesmysum2

```
sum_test.mysum2.restype = c_double
sum_test.mysum2.argtypes = [
    np.ctypeslib.ndpointer(dtype=np.float64, ndim=2),
    POINTER(c_int),
    POINTER(c_int)
]

def mysum2(x):
    return sum_test.mysum2(x, x.ctypes.strides, x.ctypes.shape)
```

mysum2xstridesshapeCx.ctypesstridesshapex.strides x.shapepythontuplex.ctypes.shapeatypes

```
>>> x = np.zeros((3,4), np.float)
>>> x.ctypes.shape
<numpy.core._internal.c_long_Array_2 object at 0x020B4DF0>
>>> s = x.ctypes.shape
>>> s[0]
3
>>> s[1]
```

x.ctypes.shapeCPythonC

Python »

← (Alt+X) | |

Python »

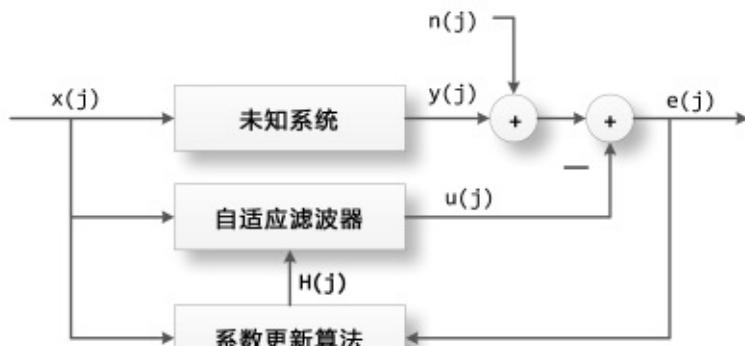
← (Alt+X) | |

NLMS

NLMSNLMSPythonctypesCNLMS

(adaptive signal process)

(system identification)



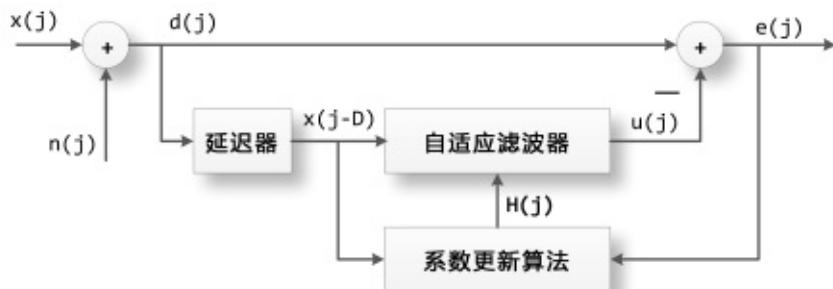
系统识别(system identification)框图

(System Identification)

$$x(j)H(j) u(j)d(j)=y(j)+n(j)n(j)d(j)u(j)e(j)=d(j)-u(j)H$$

H

jj



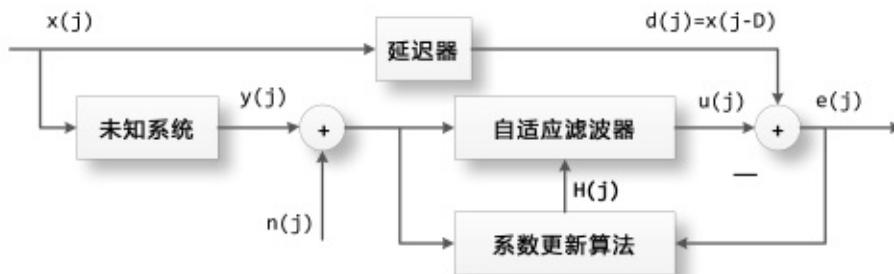
信号预测(prediction)框图

(Prediction)

$$x(j)n(j)d(j)=x(j)+n(j)$$

$$d(j)d(j-D)d(j-D)Hu(j)u(j)d(j)u(j)e(j)=d(j)-u(j)e(j)Hd(j)$$

$$x(j)x(j-D)Hu(j)e(j)$$



信号均衡(equalization)框图

(Equalization)

$$x(j)y(j)x(j)y(j)x(j)x(j)x(j-D)x(j)x(j-D)$$

$$y(j)+n(j)HHu(j)x(j-D)HH$$

NLMS

FIRNLMS()

$\mathbf{h} \leftarrow \mathbf{h}$

$$\mathbf{h}(0) = 0$$

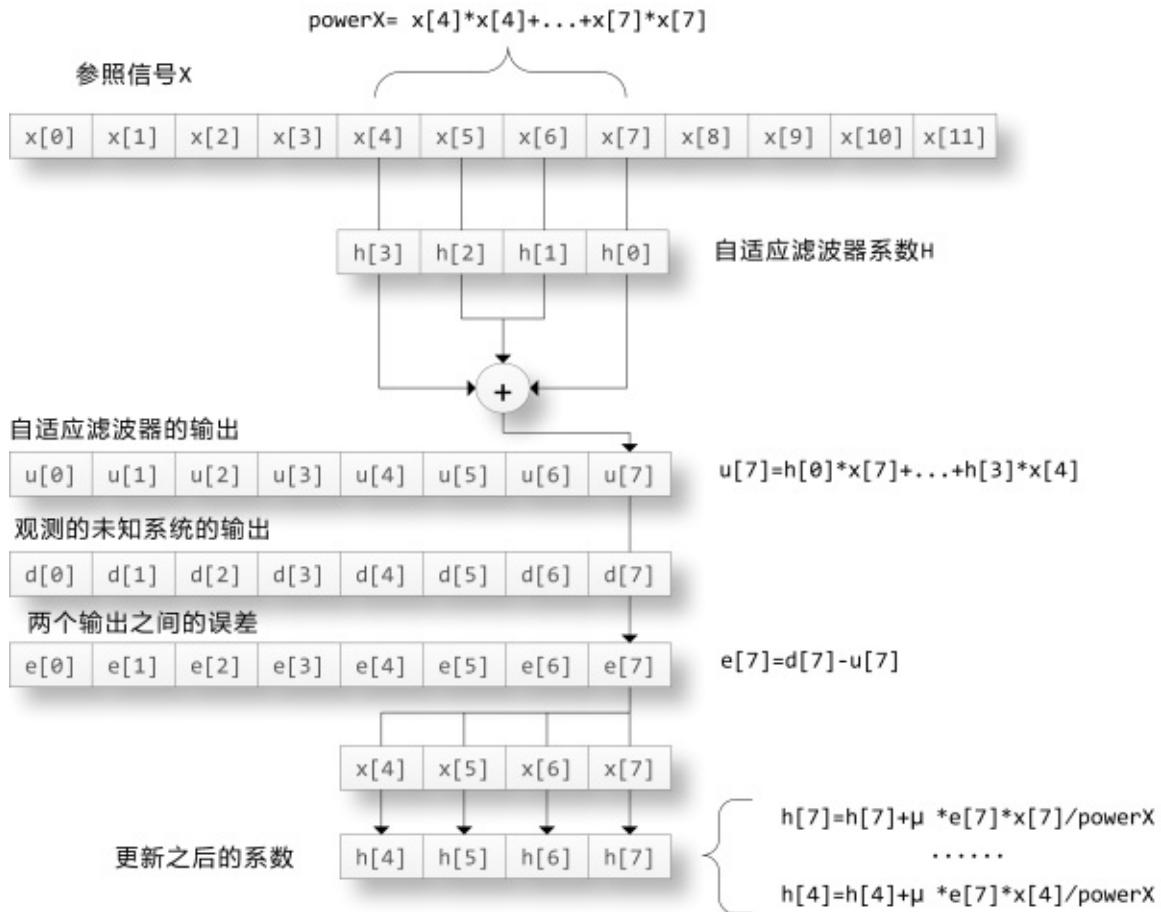
$n=0, 1, 2, \dots$

$$\mathbf{x}(n) = [x(n), x(n-1), \dots, x(n-I+1)]^T$$

$$e(n) = d(n) - \mathbf{h}^H(n)\mathbf{x}(n)$$

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \frac{\mu e(n)\mathbf{x}(n)}{\mathbf{x}^H(n)\mathbf{x}(n)}$$

$$\mathbf{h} \leftarrow \mathbf{h} + \mu \mathbf{x}(n) e(n)$$



NLMS

h47:

$$u[7] = h[0]*x[7] + h[1]*x[6] + h[2]*x[5] + h[3]*x[4]$$

powerX:

$$powerX = x[4]*x[4] + x[5]*x[5] + x[6]*x[6] + x[7]*x[7]$$

d[7]u[7]:

$$e[7] = d[7] - u[7]$$

u[7]x[4]..x[7]:

$$h[4] = h[4] + \mu * e[7]*x[4]/powerX$$

```
h[4] = h[5] + u * e[7]*x[5]/powerX  
h[4] = h[6] + u * e[7]*x[6]/powerX  
h[4] = h[7] + u * e[7]*x[7]/powerX
```

u01ix

NumPy

NLMSNumPyNLMS

```
# -*- coding: utf-8 -*-
# filename: nlms_numpy.py

import numpy as np

# NumpyNLMS
# xdh
# step_size
def nlms(x, d, h, step_size=0.5):
    i = len(h)
    size = len(x)
    # hhe
    power = np.sum( x[i:i-len(h):-1] * x[i:i-len(h):-1] )
    u = np.zeros(size, dtype=np.float64)

    while True:
        x_input = x[i:i-len(h):-1]
        u[i] = np.dot(x_input, h)
        e = d[i] - u[i]
        h += step_size * e / power * x_input

        power -= x_input[-1] * x_input[-1] #
        i+=1
        if i >= size: return u
        power += x[i] * x[i] #
```

powerhxxpowerx

nlmsxdhdxx

NLMS

```
def make_path(delay, length):
    path_length = length - delay
    h = np.zeros(length, np.float64)
    h[delay:] = np.random.standard_normal(path_length) * np.exp( n
    h /= np.sqrt(np.sum(h*h))
    return h
```

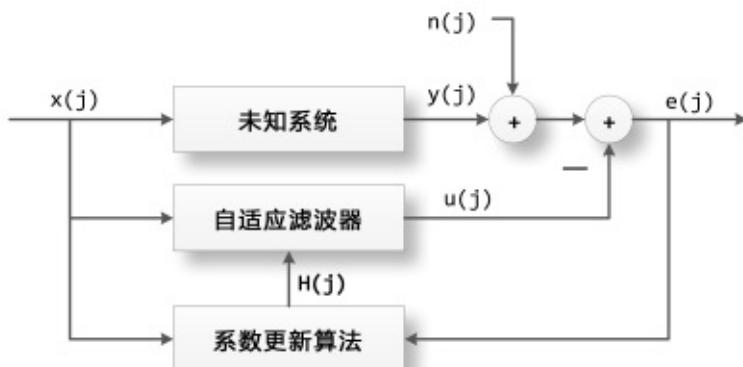
make_pathlengthdelay:

```
def plot_converge(y, u, label=""):  
    size = len(u)  
    avg_number = 200  
    e = np.power(y[:size] - u, 2)  
    tmp = e[:int(size/avg_number)*avg_number]  
    tmp.shape = -1, avg_number  
    avg = np.average( tmp, axis=1 )  
    pl.plot(np.linspace(0, size, len(avg)), 10*np.log10(avg), lin  
  
def diff_db(h0, h):  
    return 10*np.log10(np.sum((h0-h)*(h0-h)) / np.sum(h0*h0))
```

plot_convergeyuavg_numberplot_convergeyu
diff_dbh0h

nlms

```
def sim_system_identify(nlms, x, h0, step_size, noise_scale):  
    y = np.convolve(x, h0)  
    d = y + np.random.standard_normal(len(y)) * noise_scale #  
    h = np.zeros(len(h0), np.float64) # 0  
    u = nlms( x, d, h, step_size )  
    return y, u, h
```



系统识别(system identification)框图

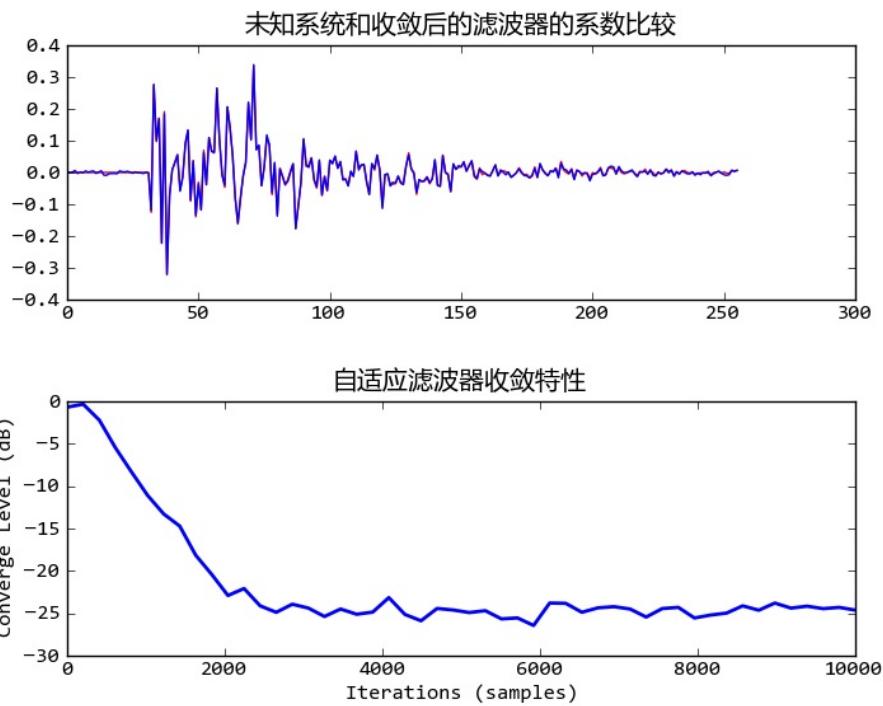
(System Identification)

- **nlms** : nlms
- **x** :
- **h0** :
- **step_size** : nlms
- **noise_scale** : 0

- **y** :
- **u** :
- **h** :

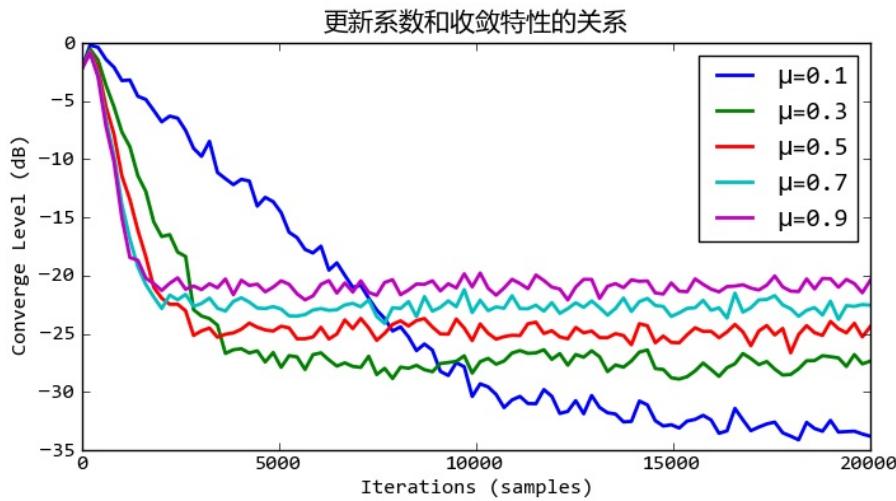
h0 xsim_system_identify

```
def system_identify_test1():
    h0 = make_path(32, 256) #
    x = np.random.standard_normal(10000) #
    y, u, h = sim_system_identify(nlms_numpy.nlms, x, h0, 0.5, 0.1)
    print diff_db(h0, h)
    pl.figure( figsize=(8, 6) )
    pl.subplot(211)
    pl.subplots_adjust(hspace=0.4)
    pl.plot(h0, c="r")
    pl.plot(h, c="b")
    pl.title(u"")
    pl.subplot(212)
    plot_converge(y, u)
    pl.title(u"")
    pl.xlabel("Iterations (samples)")
    pl.ylabel("Converge Level (dB)")
    pl.show()
```



$\text{diff_db}(h_0, h) - 25.35 \text{dB}$
 $y = \text{filter}(h, x)$

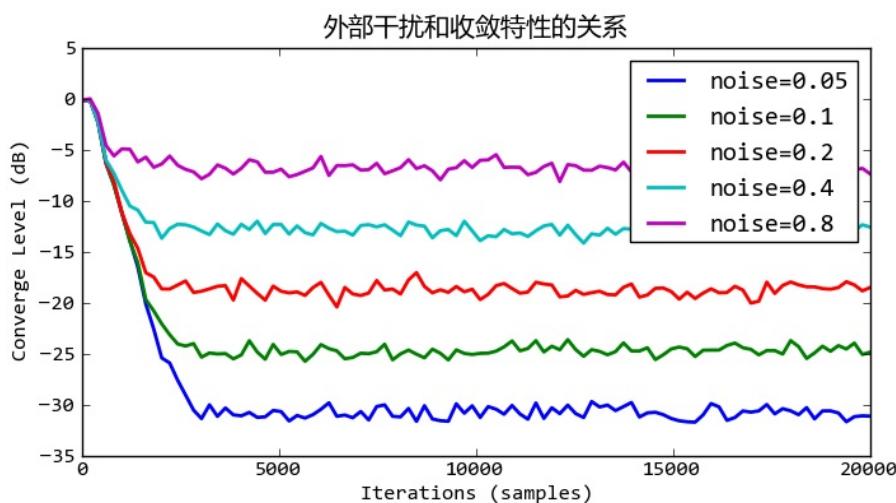
```
def system_identify_test2():
    h0 = make_path(32, 256) #
    x = np.random.standard_normal(20000) #
    pl.figure(figsize=(8,4))
    for step_size in np.arange(0.1, 1.0, 0.2):
        y, u, h = sim_system_identify(nlms_numpy.nlms, x, h0, step_size)
        plot_converge(y, u, label=u"\u03bc=%s" % step_size)
    pl.title(u"")
    pl.xlabel("Iterations (samples)")
    pl.ylabel("Converge Level (dB)")
    pl.legend()
    pl.show()
```



```

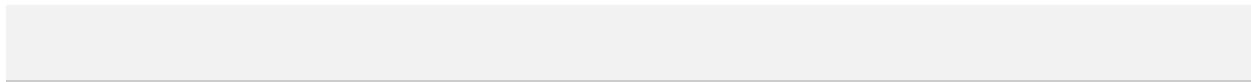
def system_identify_test3():
    h0 = make_path(32, 256) #
    x = np.random.standard_normal(20000) #
    pl.figure(figsize=(8,4))
    for noise_scale in [0.05, 0.1, 0.2, 0.4, 0.8]:
        y, u, h = sim_system_identify(nlms_numpy.nlms, x, h0, 0.5
            plot_converge(y, u, label=u"noise=%s" % noise_scale)
    pl.title(u"")
    pl.xlabel("Iterations (samples)")
    pl.ylabel("Converge Level (dB)")
    pl.legend()
    pl.show()

```

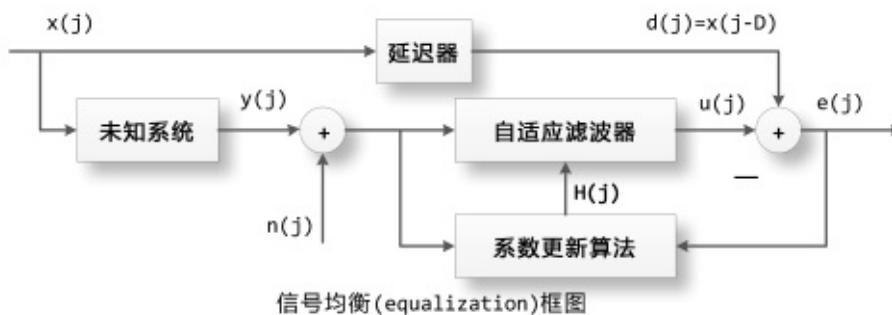




6dB 6dB



```
def sim_signal_equation(nlms, x, h0, D, step_size, noise_scale):
    d = x[:-D]
    x = x[D:]
    y = np.convolve(x, h0)[:len(x)]
    h = np.zeros(2*len(h0)+2*D, np.float64)
    y += np.random.standard_normal(len(y)) * noise_scale
    u = nlms(y, d, h, step_size)
    return h
```



sim_signal_equation

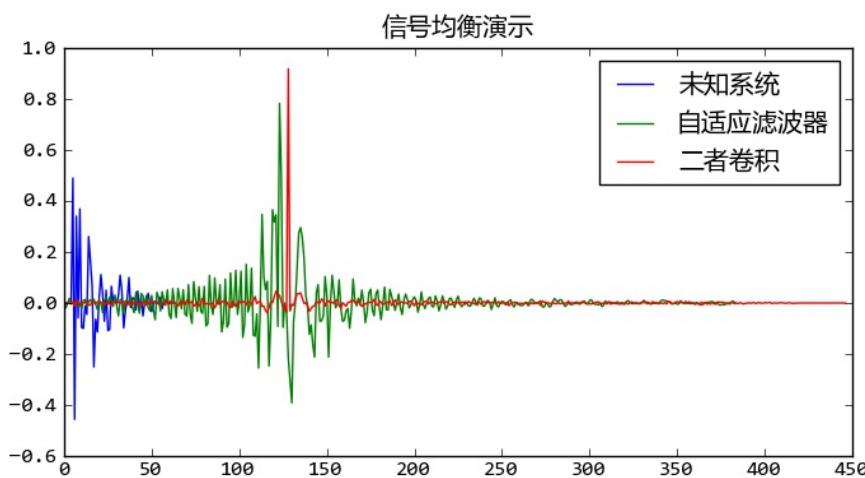
- **nlms** : nlms
 - **x** :
 - **h0** :
 - **D** :
 - **step_size** : nlms
 - **noise_scale** : 0
-
- **d** :

- y :
- h : $2 + 2$

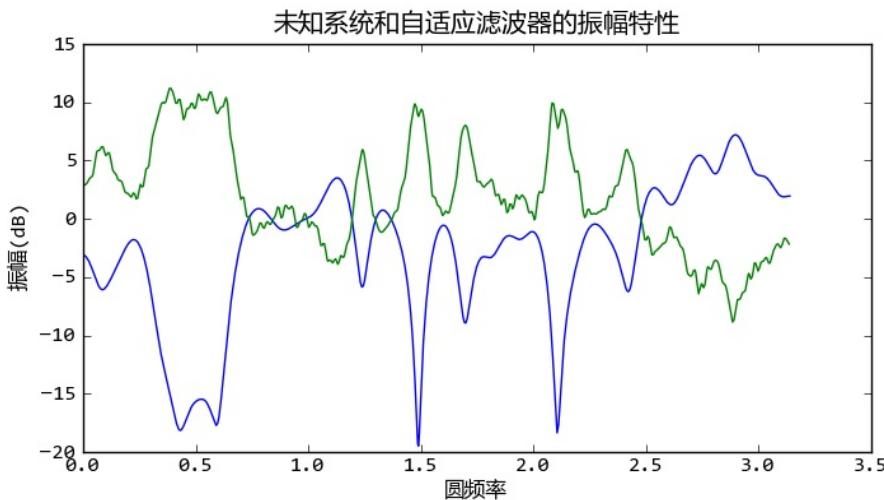
h_0

```
def signal_equation_test1():
    h0 = make_path(5, 64)
    D = 128
    length = 20000
    data = np.random.standard_normal(length+D)
    h = sim_signal_equation(nlms_numpy.nlms, data, h0, D, 0.5, 0.1)
    pl.figure(figsize=(8,4))
    pl.plot(h0, label=u"未知系统")
    pl.plot(h, label=u"自适应滤波器")
    pl.plot(np.convolve(h0, h), label=u"二者卷积")
    pl.title(u"信号均衡演示")
    pl.legend()
    w0, H0 = scipy.signal.freqz(h0, worN = 1000)
    w, H = scipy.signal.freqz(h, worN = 1000)
    pl.figure(figsize=(8,4))
    pl.plot(w0, 20*np.log10(np.abs(H0)), w, 20*np.log10(np.abs(H)))
    pl.title(u"幅频特性")
    pl.xlabel(u"频率")
    pl.ylabel(u"(dB)")
    pl.show()
```

DDh02h0, hh0hD(128)



00



h1h2h3h3h1h2h1h3

h1100h3199h2100h2100100199h3h1h3h2
h1h3

xh1uxh3dudNLMSh2

```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as pl
from nlms_numpy import nlms
import scipy.signal as signal

def inv_convolve(h1, h3, length):
    x = np.random.standard_normal(10000)
    u = signal.lfilter(h1, 1, x)
    d = signal.lfilter(h3, 1, x)
    h = np.zeros(length, np.float64)
    nlms(u, d, h, 0.1)
    return h
```

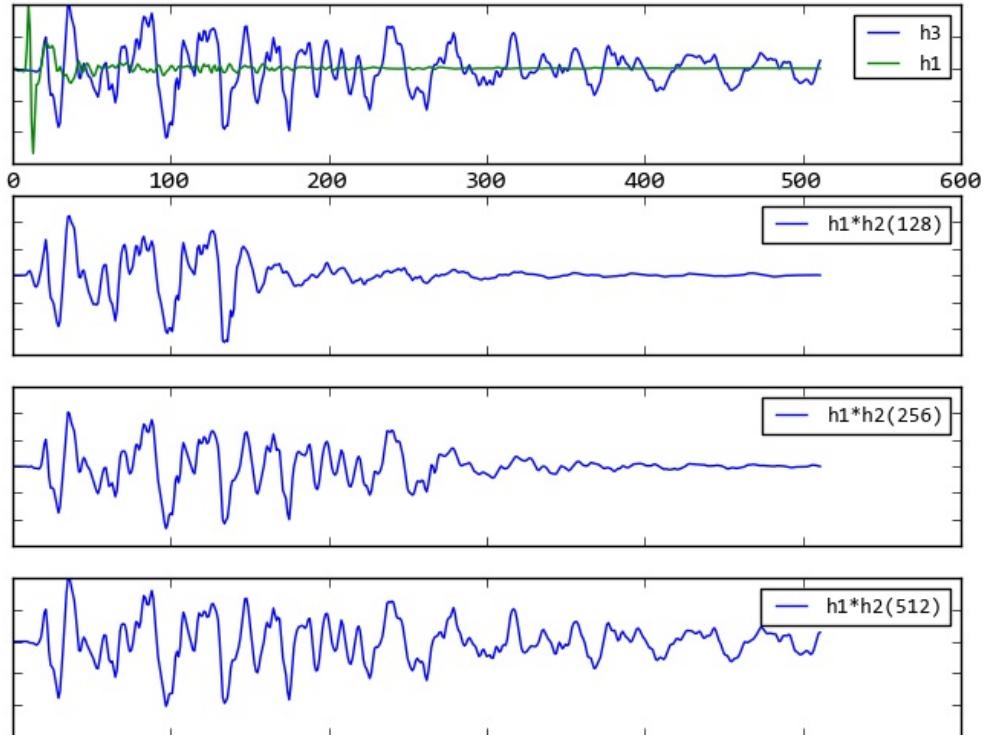
```

h1 = np.fromfile("h1.txt", sep="\n")
h1 /= np.max(h1)
h3 = np.fromfile("h3.txt", sep="\n")
h3 /= np.max(h3)

pl.rc('legend', fontsize=10)
pl.subplot(411)
pl.plot(h3, label="h3")
pl.plot(h1, label="h1")
pl.legend()
pl.gca().set_yticklabels([])
for idx, length in enumerate([128, 256, 512]):
    pl.subplot(412+idx)
    h2 = inv_convolve(h1, h3, length)
    pl.plot(np.convolve(h1, h2)[:len(h3)], label="h1*h2(%s)" % length)
    pl.legend()
    pl.gca().set_yticklabels([])
    pl.gca().set_xticklabels([])

pl.show()

```





h1h3ANC()h2

h2128, 256, 512h2

DLL

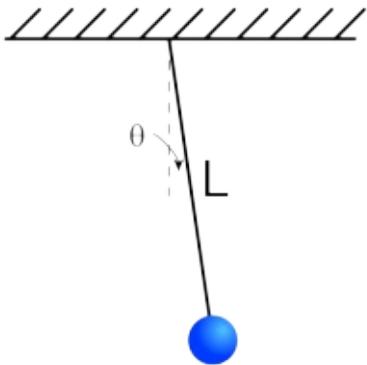
ctypespython

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |



$$\frac{d^2\theta}{dt^2} + \frac{g}{\ell} \sin \theta = 0$$

$$\theta \quad \ell \quad g$$

odeint

odeint

```
odeint(func, y0, t, ...)
```

```
funcPythony0tresult0t1 result[:, i] i
```

```
func func(y, t)ytfunct
```

odeint

$$\frac{d\theta(t)}{dt} = v(t)$$

$$\frac{dv(t)}{dt} = -\frac{g}{\ell} \sin \theta(t)$$

odeint

```
# -*- coding: utf-8 -*-

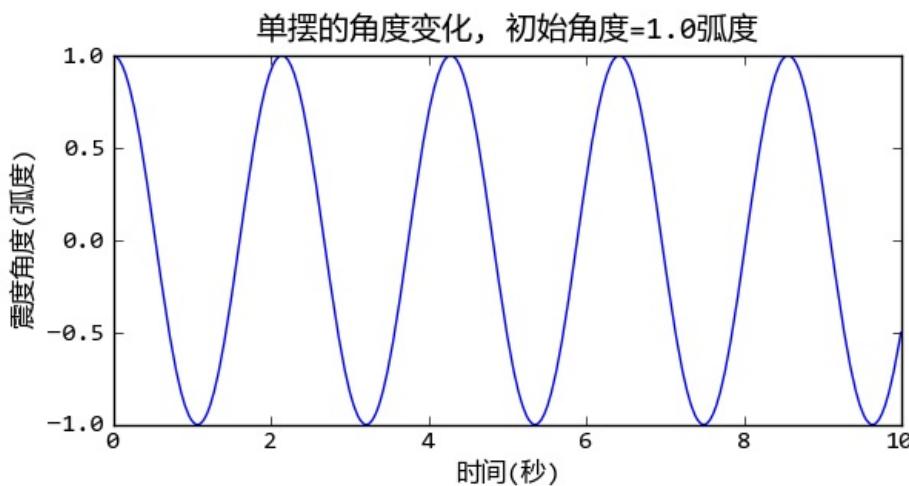
from math import sin
import numpy as np
from scipy.integrate import odeint

g = 9.8

def pendulum_equations(w, t, l):
    th, v = w
    dth = v
    dv = - g/l * sin(th)
    return dth, dv

if __name__ == "__main__":
    import pylab as pl
    t = np.arange(0, 10, 0.01)
    track = odeint(pendulum_equations, (1.0, 0), t, args=(1.0,))
    pl.plot(t, track[:, 0])
    pl.title(u"      =1.0")
    pl.xlabel(u"()")
    pl.ylabel(u"()")
    pl.show()
```

odeintargsfuncpendulum_equations



1

$$T_0 = 2\pi \sqrt{\frac{\ell}{g}}$$

$$\theta \ll 1 \quad \sin \theta \approx \theta$$

$$\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\theta = 0$$

04pendulum_th

```
def pendulum_th(t, l, th0):
    track = odeint(pendulum_equations, (th0, 0), [0, t], args=(l,
    return track[-1, 0]
```

pendulum_th l th0 toodeint toodeint[0, t] odeint

pendulum_th 0 pendulum_th scipy.optimize.fsolve

```
def pendulum_period(l, th0):
    t0 = 2*np.pi*sqrt( 1/g ) / 4
    t = fsolve( pendulum_th, t0, args = (l, th0) )
    return t**4
```

odeint fsolve args pendulum_th fsolve 1/4

pendulum_period 0 90

```
ths = np.arange(0, np.pi/2.0, 0.01)
periods = [pendulum_period(1, th) for th in ths]
```

fsolve

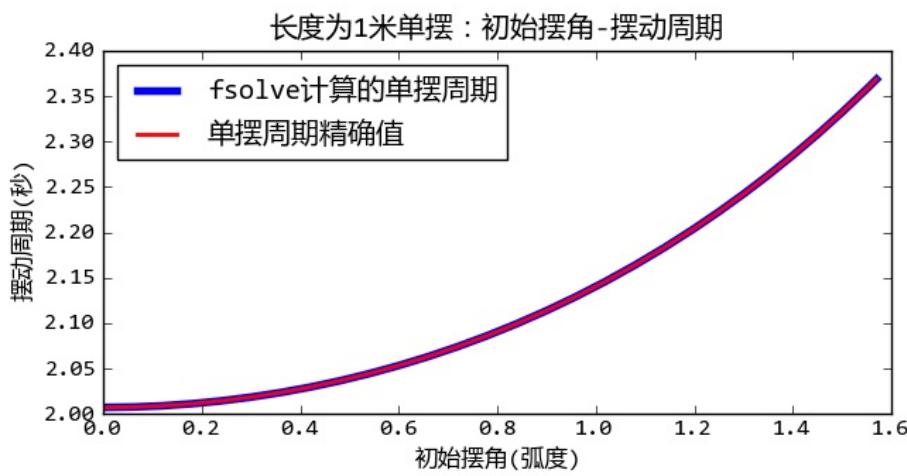
$$T = 4\sqrt{\frac{\ell}{g}} K \left(\sin \frac{\theta_0}{2} \right)$$

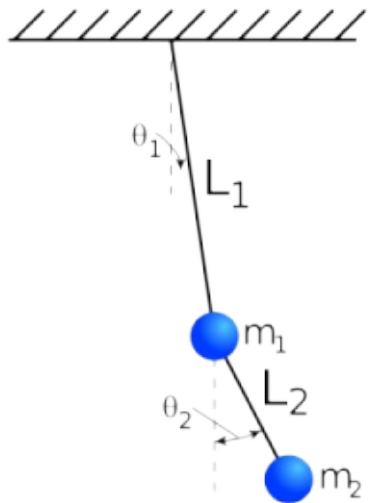
K

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

scipy.special.ellipk

```
periods2 = 4*sqrt(1.0/g)*ellipk(np.sin(ths/2)**2)
```





L1L2m1m2 θ_1 θ_2

0

1788

L1x1y1L2x2y2x1,y1,x2,y2

$$x_1 = L_1 \sin(\theta_1)$$

$$y_1 = -L_1 \cos(\theta_1)$$

$$x_2 = L_1 \sin(\theta_1) + L_2 \sin(\theta_2)$$

$$y_2 = -L_1 \cos(\theta_1) - L_2 \cos(\theta_2)$$

$$\mathcal{L}=T-V$$

$$\mathsf{T}\mathsf{V}$$

$$\mathcal{L}=\frac{m_1}{2}(\dot{x}_1^2+\dot{y}_1^2)+\frac{m_2}{2}(\dot{x}_2^2+\dot{y}_2^2)-m_1gy_1-m_2gy_2$$

$$\begin{aligned}\mathcal{L} = & \frac{m_1+m_2}{2}L_1^2\dot{\theta}_1^2+\frac{m_2}{2}L_2^2\dot{\theta}_2^2+m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2)+\\& (m_1+m_2)gL_1\cos(\theta_1)+m_2gL_2\cos(\theta_2)\end{aligned}$$

$$\theta _1$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}-\frac{\partial \mathcal{L}}{\partial \theta_1}=0$$

$$L_1[(m_1+m_2)L_1\ddot{\theta}_1+m_2L_2\cos(\theta_1-\theta_2)\ddot{\theta}_2+m_2L_2\sin(\theta_1-\theta_2)\dot{\theta}_2^2+(m_1+m_2)g\sin(\theta_1)]=0$$

$$\theta _2$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}-\frac{\partial \mathcal{L}}{\partial \theta_2}=0$$

$$m_2L_2[L_2\ddot{\theta}_2+L_1\cos(\theta_1-\theta_2)\ddot{\theta}_1-L_1\sin(\theta_1-\theta_2)\dot{\theta}_1^2+g\sin(\theta_2)]=0$$

sympy

```
# -*- coding: utf-8 -*-

```

```

from sympy import *
from sympy import Derivative as D

var("x1 x2 y1 y2 l1 l2 m1 m2 th1 th2 dth1 dth2 ddth1 ddth2 t g tmp v1 v2")

sublist = [
(D(th1(t), t, t), ddth1),
(D(th1(t), t), dth1),
(D(th2(t), t, t), ddth2),
(D(th2(t), t), dth2),
(th1(t), th1),
(th2(t), th2)
]

x1 = l1*sin(th1(t))
y1 = -l1*cos(th1(t))
x2 = l1*sin(th1(t)) + l2*sin(th2(t))
y2 = -l1*cos(th1(t)) - l2*cos(th2(t))

vx1 = diff(x1, t)
vx2 = diff(x2, t)
vy1 = diff(y1, t)
vy2 = diff(y2, t)

#
L = m1/2*(vx1**2 + vy1**2) + m2/2*(vx2**2 + vy2**2) - m1*g*y1 - m2*g*y2

#
def lagrange_equation(L, v):
    a = L.subs(D(v(t), t), tmp).diff(tmp).subs(tmp, D(v(t), t))
    b = L.subs(D(v(t), t), tmp).subs(v(t), v).diff(v).subs(v, v(t))
    c = a.diff(t) - b
    c = c.subs(sublist)
    c = trigsimp(simplify(c))
    c = collect(c, [th1, th2, dth1, dth2, ddth1, ddth2])
    return c

eq1 = lagrange_equation(L, th1)
eq2 = lagrange_equation(L, th2)

```

$$eq1 \quad \theta_1 \quad eq2 \quad \theta_2$$

$$\frac{\partial \mathcal{L}}{\partial v} = D(L, t), \quad D(f, v(t)) \quad \text{tmp} = D(v(t), t)$$

```
L.subs(D(v(t), t), tmp).diff(tmp).subs(tmp, D(v(t), t))
```

$$\partial \mathcal{L}/\partial v \ v(t) v' D(v(t), t) \ D(v(t), t) \ tmp$$

```
L.subs(D(v(t), t), tmp).subs(v(t), v).diff(v).subs(v, v(t)).subs(v, v(t))
```

eq1, eq2

```
>>> eq1
ddth1*(m1*l1**2 + m2*l1**2) +
ddth2*(l1*l2*m2*cos(th1)*cos(th2) + l1*l2*m2*sin(th1)*sin(th2)) +
dth2**2*(l1*l2*m2*cos(th2)*sin(th1) - l1*l2*m2*cos(th1)*sin(th2))
g*l1*m1*sin(th1) + g*l1*m2*sin(th1)
>>> eq2
ddth1*(l1*l2*m2*cos(th1)*cos(th2) + l1*l2*m2*sin(th1)*sin(th2)) +
dth1**2*(l1*l2*m2*cos(th1)*sin(th2) - l1*l2*m2*cos(th2)*sin(th1))
g*l2*m2*sin(th2) + ddth2*m2*l2**2
```

$$\sin(x+y) = \sin x \cos y + \cos x \sin y$$

$$\cos(x+y) = \cos x \cos y - \sin x \sin y$$

$$\sin(x-y) = \sin x \cos y - \cos x \sin y$$

$$\cos(x-y) = \cos x \cos y + \sin x \sin y$$

$$(m_1 + m_2)L_1\ddot{\theta}_1 + m_2L_2\cos(\theta_1 - \theta_2)\ddot{\theta}_2 + m_2L_2\sin(\theta_1 - \theta_2)\dot{\theta}_2^2 + (m_1 + m_2)g\sin(\theta_1) = 0$$

$$L_2\ddot{\theta}_2 + L_1\cos(\theta_1 - \theta_2)\ddot{\theta}_1 - L_1\sin(\theta_1 - \theta_2)\dot{\theta}_1^2 + g\sin(\theta_2) = 0$$

odeint44

$\theta_1, \theta_2, v_1, v_2$

$$\dot{\theta}_1 = v_1$$

$$\dot{\theta}_2 = v_2$$

$$(m_1 + m_2)L_1\dot{v}_1 + m_2L_2 \cos(\theta_1 - \theta_2)\dot{v}_2 + m_2L_2 \sin(\theta_1 - \theta_2)\dot{\theta}_2^2 + (m_1 + m_2)g \sin(\theta_1) = 0$$

$$L_2\dot{v}_2 + L_1 \cos(\theta_1 - \theta_2)\dot{v}_1 - L_1 \sin(\theta_1 - \theta_2)\dot{\theta}_1^2 + g \sin(\theta_2) = 0$$

scipy.integrate.odeint

```
# -*- coding: utf-8 -*-

from math import sin,cos
import numpy as np
from scipy.integrate import odeint

g = 9.8

class DoublePendulum(object):
    def __init__(self, m1, m2, l1, l2):
        self.m1, self.m2, self.l1, self.l2 = m1, m2, l1, l2
        self.init_status = np.array([0.0,0.0,0.0,0.0])

    def equations(self, w, t):
        """
        m1, m2, l1, l2 = self.m1, self.m2, self.l1, self.l2
        th1, th2, v1, v2 = w
        dth1 = v1
        dth2 = v2

        #eq of th1
        a = l1*l1*(m1+m2) # dv1 parameter
        b = l1*m2*l2*cos(th1-th2) # dv2 paramter
        c = l1*(m2*l2*sin(th1-th2)*dth2*dth2 + (m1+m2)*g*sin(th1))

        #eq of th2
        d = m2*l2*l1*cos(th1-th2) # dv1 parameter
        e = m2*l2*l2 # dv2 parameter
        f = m2*l2*(-l1*sin(th1-th2)*dth1*dth1 + g*sin(th2))

        dv1, dv2 = np.linalg.solve([[a,b],[d,e]], [-c,-f])

        return np.array([dth1, dth2, dv1, dv2])

def double_pendulum_odeint(pendulum, ts, te, tstep):
    """
    X-Y
    """
    pass
```

```

t = np.arange(ts, te, tstep)
track = odeint(pendulum.equations, pendulum.init_status, t)
th1_array, th2_array = track[:, 0], track[:, 1]
l1, l2 = pendulum.l1, pendulum.l2
x1 = l1*np.sin(th1_array)
y1 = -l1*np.cos(th1_array)
x2 = x1 + l2*np.sin(th2_array)
y2 = y1 - l2*np.cos(th2_array)
pendulum.init_status = track[-1, :].copy() #pendulum
return [x1, y1, x2, y2]

if __name__ == "__main__":
    import matplotlib.pyplot as pl
    pendulum = DoublePendulum(1.0, 2.0, 1.0, 2.0)
    th1, th2 = 1.0, 2.0
    pendulum.init_status[:2] = th1, th2
    x1, y1, x2, y2 = double_pendulum_odeint(pendulum, 0, 30, 0.02)
    pl.plot(x1,y1, label = u"" )
    pl.plot(x2,y2, label = u"" )
    pl.title(u", = %s,%s" % (th1, th2))
    pl.legend()
    pl.axis("equal")
    pl.show()

```

DoublePendulum.equations w

- th1:
- th2:
- v1:
- v2:

- dth1:
- dth2:
- dv1:
- dv2:

dth1dth2

```

dth1 = v1
dth2 = v2

```

$\text{dv1}\text{dv2}:$

$$\dot{v}_1 = \dots$$

$$\dot{v}_2 = \dots$$

$\text{dv1 } \text{dv2 } \text{linalg.solve}$

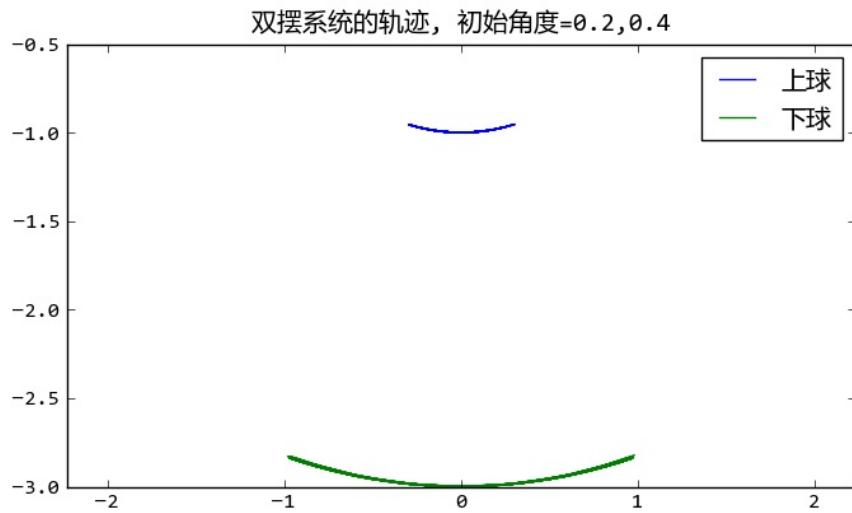
```
#eq of th1
a = l1*l1*(m1+m2) # dv1 parameter
b = l1*m2*l2*cos(th1-th2) # dv2 paramter
c = l1*(m2*l2*sin(th1-th2)*dth2*dth2 + (m1+m2)*g*sin(th1))

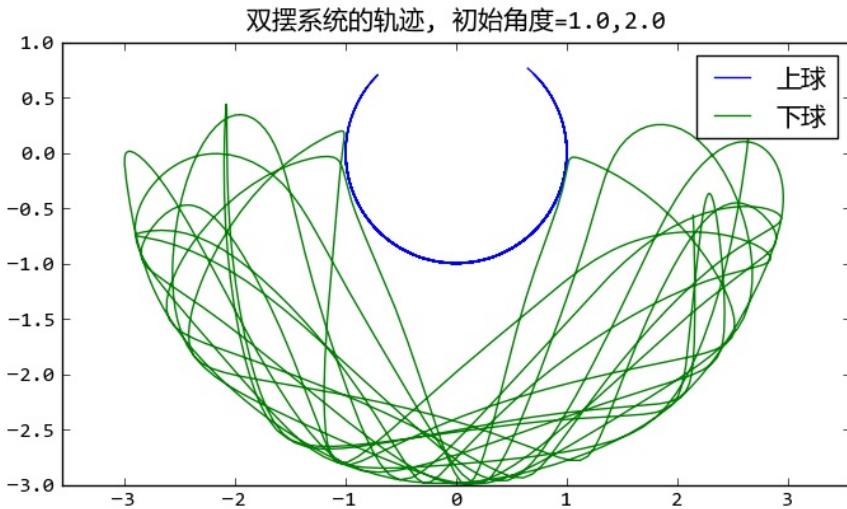
#eq of th2
d = m2*l2*l1*cos(th1-th2) # dv1 parameter
e = m2*l2*l2 # dv2 parameter
f = m2*l2*(-l1*sin(th1-th2)*dth1*dth1 + g*sin(th2))

dv1, dv2 = np.linalg.solve([[a,b],[d,e]], [-c,-f])
```

$$a\dot{v}_1 + b\dot{v}_2 + c = 0$$

$$d\dot{v}_1 + e\dot{v}_2 + f = 0$$





- visual3D
- pygame2D
- tkinterwxpython

matplotlib

```
# -*- coding: utf-8 -*-
import matplotlib
matplotlib.use('WXAgg') # do this before importing pylab
import matplotlib.pyplot as pl
from double_pendulum_odeint import double_pendulum_odeint, DoublePendulum

fig = pl.figure(figsize=(4,4))
line1, = pl.plot([0,0], [0,0], "-o")
line2, = pl.plot([0,0], [0,0], "-o")
pl.axis("equal")
pl.xlim(-4,4)
pl.ylim(-4,2)
```

```

pendulum = DoublePendulum(1.0, 2.0, 1.0, 2.0)
pendulum.init_status[:] = 1.0, 2.0, 0, 0

x1, y1, x2, y2 = [],[],[],[]
idx = 0

def update_line(event):
    global x1, x2, y1, y2, idx
    if idx == len(x1):
        x1, y1, x2, y2 = double_pendulum_odeint(pendulum, 0, 1, 0
        idx = 0
    line1.set_xdata([0, x1[idx]])
    line1.set_ydata([0, y1[idx]])
    line2.set_xdata([x1[idx], x2[idx]])
    line2.set_ydata([y1[idx], y2[idx]])
    fig.canvas.draw()
    idx += 1

import wx
id = wx.NewId()
actor = fig.canvas.manager.frame
timer = wx.Timer(actor, id=id)
timer.Start(1)
wx.EVT_TIMER(actor, id, update_line)
pl.show()

```

WXAgg

```
matplotlib.use('WXAgg')
```

wxupdate_line

```

import wx
id = wx.NewId()
actor = fig.canvas.manager.frame
timer = wx.Timer(actor, id=id)
timer.Start(1)
wx.EVT_TIMER(actor, id, update_line)

```

update_line

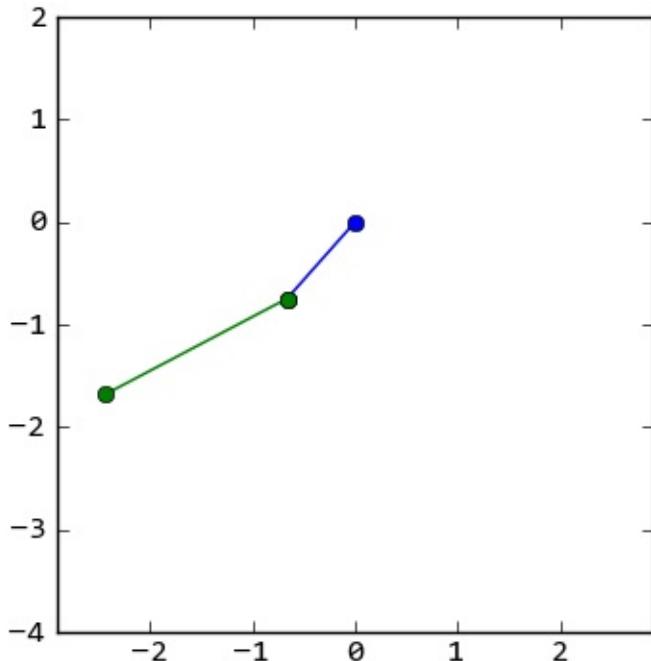
```

if idx == len(x1):
    x1, y1, x2, y2 = double_pendulum_odeint(pendulum, 0, 1, 0.05)
    idx = 0

```

```
double_pendulum_odeint odeint pendulum.init_status
double_pendulum_odeint
```

```
def double_pendulum_odeint(pendulum, ts, te, tstep):
    ...
    track = odeint(pendulum.equations, pendulum.init_status, t)
    ...
    pendulum.init_status = track[-1,:].copy()
    return [x1, y1, x2, y2]
```



Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

-
-

Python

Mandelbrot

Mandelbrot()

Mandelbrot()

Mandelbrot

$$f_c(z) = z^2 + c$$

$$c \in \mathbb{C}, z \in \mathbb{C}$$

$$(0, f_c(0), f_c(f_c(0)), f_c(f_c(f_c(0))), \dots)$$

Mandelbrot

Mandelbrotc Mandelbrot

Mandelbrot()

- $f_c(z) \in \mathbb{R}$
- \mathbb{R}
- \mathbb{N}
-

Mandelbrot

```
# -*- coding: utf-8 -*-

import numpy as np
import pylab as pl
import time
from matplotlib import cm

def iter_point(c):
    z = c
```

```

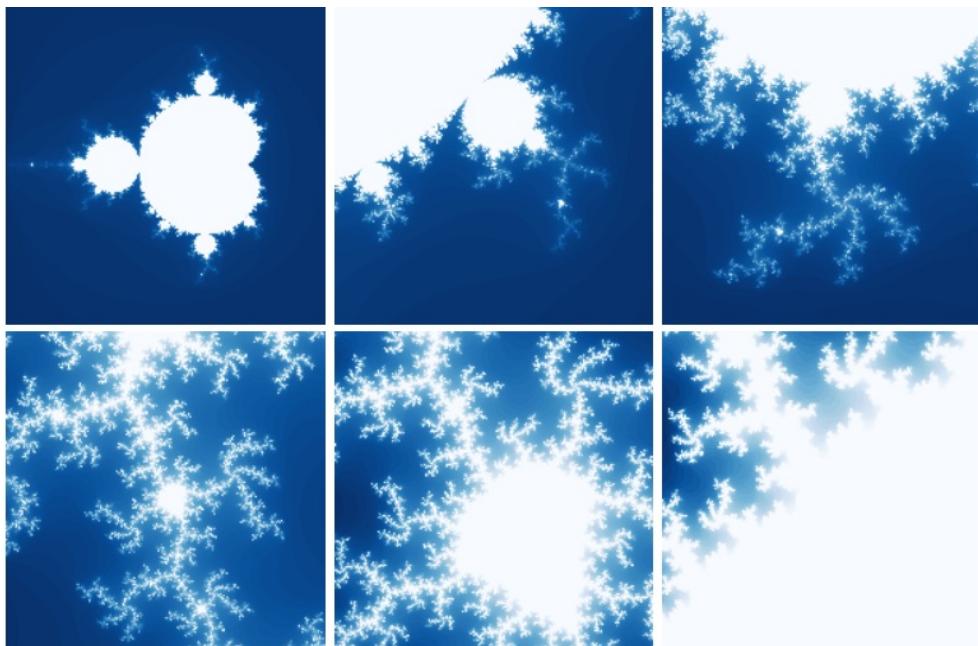
for i in xrange(1, 100): # 100
    if abs(z)>2: break # 2
    z = z*z+c
return i #

def draw_mandelbrot(cx, cy, d):
    """
    (cx, cy)dMandelbrot
    """
    x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d
    y, x = np.ogrid[y0:y1:200j, x0:x1:200j]
    c = x + y*1j
    start = time.clock()
    mandelbrot = np.frompyfunc(iter_point,1,1)(c).astype(np.float)
    print "time=",time.clock() - start
    pl.imshow(mandelbrot, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
    pl.gca().set_axis_off()

x,y = 0.27322626, 0.595153338

pl.subplot(231)
draw_mandelbrot(-0.5,0,1.5)
for i in range(2,7):
    pl.subplot(230+i)
    draw_mandelbrot(x, y, 0.2***(i-1))
pl.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0)
pl.show()

```



 Mandelbrot(0.273, 0.595)

```
iter_pointcR2.0100draw_mandelbrot(cx, cy)2*dMandelbrot
```

3cc200*200np.ogrid

```
x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d  
y, x = np.ogrid[y0:y1:200j, x0:x1:200j]  
c = x + y*1j
```

```
np.frompyfunciter_pointNumPyufunciter_pointobjectastype
```

```
mandelbrot = np.frompyfunc(iter_point,1,1)(c).astype(np.float)
```

matplotlibimshowcmap

```
pl.imshow(mandelbrot, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
```

PythonMandelbrot

```
time= 0.88162629608  
time= 1.53712748408  
time= 1.71502160191  
time= 1.8691174437  
time= 3.03812691278
```

iter_pointCscipy.weaveC++iter_pointPython

```
import scipy.weave as weave  
  
def weave_iter_point(c):  
    code = """  
    std::complex<double> z;  
    int i;  
    z = c;  
    for(i=1;i<100;i++)
```

```

{
    if(std::abs(z) > 2) break;
    z = z*z+c;
}
return_val=i;
"""

f = weave.inline(code, ["c"], compiler="gcc")
return f

```

weave_iter_pointMandelbrot

```

time= 0.285266982256
time= 0.271430028118
time= 0.293769180161
time= 0.308515188383
time= 0.411168179196

```

NumPyNumPy

```

# -*- coding: utf-8 -*-

import numpy as np
import pylab as pl
import time
from matplotlib import cm

def draw_mandelbrot(cx, cy, d, N=200):
    """
    (cx, cy)dMandelbrot
    """
    global mandelbrot

    x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d
    y, x = np.ogrid[y0:y1:N*1j, x0:x1:N*1j]
    c = x + y*1j

    # X, Y
    ix, iy = np.mgrid[0:N,0:N]

    # mandelbrot
    mandelbrot = np.ones(c.shape, dtype=np.int)*100

    #
    ix.shape = -1

```

```

iy.shape = -1
c.shape = -1
z = c.copy() # c1

start = time.clock()

for i in xrange(1,100):
    #
    z *= z
    z += c
    #
    tmp = np.abs(z) > 2.0
    # mandelbrot
    mandelbrot[ix[tmp], iy[tmp]] = i

    #
    np.logical_not(tmp, tmp)
    # ix, iy, c, z
    ix,iy,c,z = ix[tmp], iy[tmp], c[tmp],z[tmp]
    if len(z) == 0: break

print "time=",time.clock() - start
pl.imshow(mandelbrot, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
pl.gca().set_axis_off()

x,y = 0.27322626, 0.595153338

pl.subplot(231)
draw_mandelbrot(-0.5,0,1.5)
for i in range(2,7):
    pl.subplot(230+i)
    draw_mandelbrot(x, y, 0.2***(i-1))
pl.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0)
pl.show()

```

ixiyzc

```

#
z *= z
z += c

```

`*=`, `+=` NumPyz

tmpzzixiyix[tmp], iy[tmp]

```
#  
tmp = np.abs(z) > 2.0  
# mandelbrot  
mandelbrot[ix[tmp], iy[tmp]] = i
```

tmpix, iy, c, z

```
#  
np.logical_not(tmp, tmp)  
# ix, iy, c, z  
ix,iy,c,z = ix[tmp], iy[tmp], c[tmp], z[tmp]
```

```
time= 0.186070576008  
time= 0.327006365334  
time= 0.372756034636  
time= 0.410074464771  
time= 0.681048289658  
time= 0.878626752841
```

RN Mandelbrot N

$$n - \log_2 \log_2 |z_n|$$

$$z_n \in$$

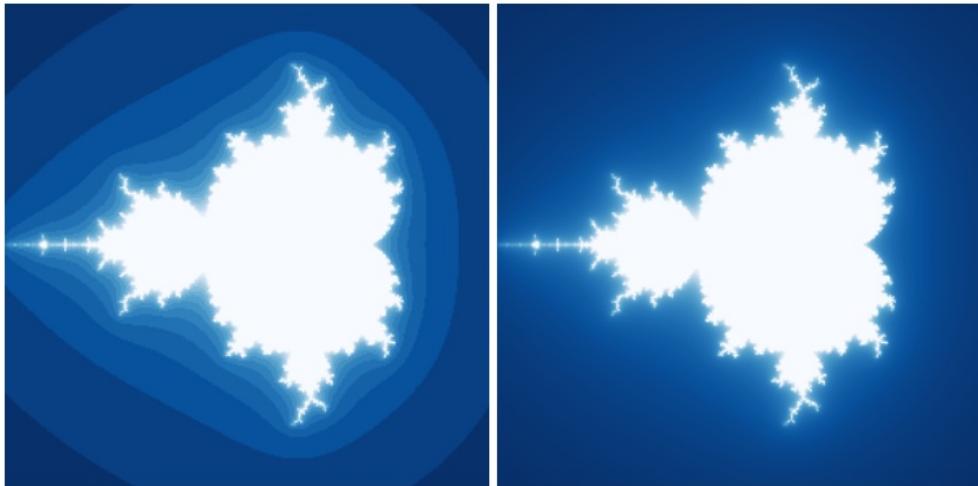
```
def smooth_iter_point(c):  
    z = c  
    for i in xrange(1, iter_num):  
        if abs(z)>escape_radius: break  
        z = z*z+c  
    absz = abs(z)  
    if absz > 2.0:  
        mu = i - log(log(abs(z),2),2)  
    else:  
        mu = i
```

```
return mu #
```

2.0z

```
z = z*z+c  
z = z*z+c  
i += 2
```

1020



 =10=20Mandelbrot

<http://linas.org/art-gallery/escape/ray.html>

Mandelbrot

4

```

1.
x(n+1) = 0
y(n+1) = 0.16 * y(n)

2.
x(n+1) = 0.2 * x(n) - 0.26 * y(n)
y(n+1) = 0.23 * x(n) + 0.22 * y(n) + 1.6

3.
x(n+1) = -0.15 * x(n) + 0.28 * y(n)
y(n+1) = 0.26 * x(n) + 0.24 * y(n) + 0.44

4.
x(n+1) = 0.85 * x(n) + 0.04 * y(n)
y(n+1) = -0.04 * x(n) + 0.85 * y(n) + 1.6

```

$x(n), y(n)$ $x(n+1), y(n+1)$ 41%, 7%, 7% 85%

(0,0)

```

# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as pl
import time

#
eq1 = np.array([[0,0,0],[0,0.16,0]])
p1 = 0.01

eq2 = np.array([[0.2,-0.26,0],[0.23,0.22,1.6]])
p2 = 0.07

eq3 = np.array([[-0.15, 0.28, 0],[0.26,0.24,0.44]])
p3 = 0.07

```

```

eq4 = np.array([[0.85, 0.04, 0], [-0.04, 0.85, 1.6]])
p4 = 0.85

def ifs(p, eq, init, n):
    """
    p:
    eq:
    init:
    n:

    X Y
    """

    #
    pos = np.ones(3, dtype=np.float)
    pos[:2] = init

    #
    p = np.add.accumulate(p)
    rands = np.random.rand(n)
    select = np.ones(n, dtype=np.int)*(n-1)
    for i, x in enumerate(p[::-1]):
        select[rands<x] = len(p)-i-1

    #
    result = np.zeros((n,2), dtype=np.float)
    c = np.zeros(n, dtype=np.float)

    for i in xrange(n):
        eqidx = select[i] #
        tmp = np.dot(eq[eqidx], pos) #
        pos[:2] = tmp #

        #
        result[i] = tmp
        c[i] = eqidx

    return result[:,0], result[:, 1], c

start = time.clock()
x, y, c = ifs([p1,p2,p3,p4],[eq1,eq2,eq3,eq4], [0,0], 100000)
print time.clock() - start
pl.figure(figsize=(6,6))
pl.subplot(121)
pl.scatter(x, y, s=1, c="g", marker="s", linewidths=0)

```

```
pl.axis("equal")
pl.axis("off")
pl.subplot(122)
pl.scatter(x, y, s=1, c = c, marker="s", linewidths=0)
pl.axis("equal")
pl.axis("off")
pl.subplots_adjust(left=0, right=1, bottom=0, top=1, wspace=0, hspace=0)
pl.gcf().patch.set_facecolor("white")
pl.show()
```

ifs(numpy.dot)

```
pos = np.ones(3, dtype=np.float)
pos[:2] = init
```

x(n), y(n), 1.0

selectaccumulate01

```
p = np.add.accumulate(p)
rands = np.random.rand(n)
select = np.ones(n, dtype=np.int)*(n-1)
for i, x in enumerate(p[::-1]):
    select[rands<x] = len(p)-i-1
```

scatter

```
pl.scatter(x, y, s=1, c="g", marker="s", linewidths=0)
```

- **s** : 101
- **c** :
- **marker** : "s"
- **linewidths** : 0

c

```
pl.scatter(x, y, s=1, c = c, marker="s", linewidths=0)
```



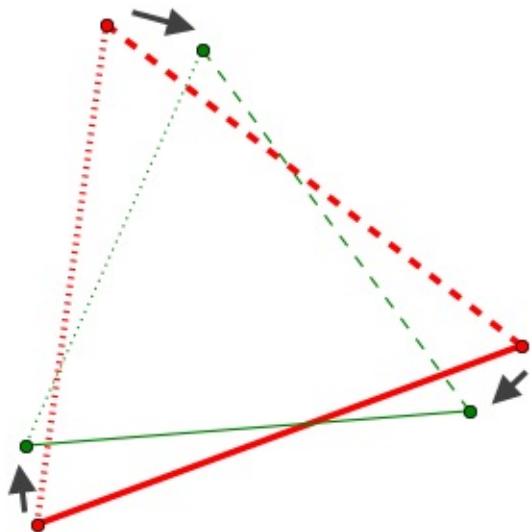
41% 1() 7% 85%

2D

$$\begin{aligned}x(n+1) &= A * x(n) + B * y(n) + C \\y(n+1) &= D * x(n) + E * y(n) + F\end{aligned}$$

2D2D2D

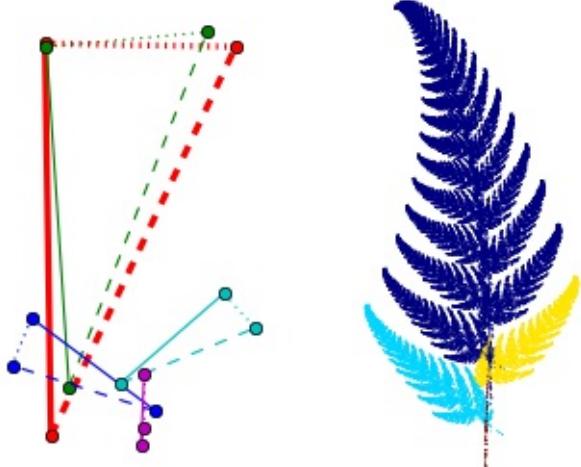
6A, B, C, D, E, F3



2D

N+1N00

5



5

solve_eqab, NumPy.linalg.solve $a^*X = b$

```

def solve_eq(triangle1, triangle2):
    """
        triangle1triangle2
            triangle1,2
            x0,y0
            x1,y1
            x2,y2
    """
    x0,y0 = triangle1[0]
    x1,y1 = triangle1[1]
    x2,y2 = triangle1[2]

    a = np.zeros((6,6), dtype=np.float)
    b = triangle2.reshape(-1)
    a[0, 0:3] = x0,y0,1
    a[1, 3:6] = x0,y0,1
    a[2, 0:3] = x1,y1,1
    a[3, 3:6] = x1,y1,1
    a[4, 0:3] = x2,y2,1
    a[5, 3:6] = x2,y2,1

    c = np.linalg.solve(a, b)
    c.shape = (2,3)
    return c

```

triangle_areaNumPycross

```

def triangle_area(triangle):
    """
    """
    A = triangle[0]
    B = triangle[1]
    C = triangle[2]
    AB = A-B
    AC = A-C
    return np.abs(np.cross(AB, AC))/2.0

```

TraitsUImatplotlibFigureMPLFigureEditor_MPLFigureEditor
 TraitsUI *Trait*

IFSDesigner._figure_defaultFigureaxax2axax2

```

def _figure_default(self):

```

```

"""
figureFigure
"""

figure = Figure()
self.ax = figure.add_subplot(121)
self.ax2 = figure.add_subplot(122)
self.ax2.set_axis_off()
self.ax.set_axis_off()
figure.subplots_adjust(left=0, right=1, bottom=0, top=1, wspace=0
figure.patch.set_facecolor("w")
return figure

```

IFSTrianglesFigurecanvas

```

canvas = ax.figure.canvas
# canvas
canvas.mpl_connect('button_press_event', self.button_press_callback)
canvas.mpl_connect('button_release_event', self.button_release_callback)
canvas.mpl_connect('motion_notify_event', self.motion_notify_callback)

```

canvasFigureFigureIFSTrianglesIFSDesignerview handlerIFSHandlerHandlerinit

```

class IFSHandler(Handler):
    """

    """
    def init(self, info):
        info.object.init_gui_component()
        return True

```

IFSDesignerinit_gui_componentcanvas

```

def init_gui_component(self):
    self.ifs_triangle = IFSTriangles(self.ax)
    self.figure.canvas.draw()
    thread.start_new_thread( self.ifs_calculate, ())
    ...

```

ifs_calculateITER_COUNTax2.scatterax2ax2 draw_pointsITER_TIMESscatterwx.CallAfterGUIDraw_points :

```

def ifs_calculate(self):
    """
    """
    def draw_points(x, y, c):
        if len(self.ax2.collections) < ITER_TIMES:
            try:
                self.ax2.scatter(x, y, s=1, c=c, marker="s", lineWidth=1)
                self.ax2.set_axis_off()
                self.ax2.axis("equal")
                self.figure.canvas.draw()
            except:
                pass

    def clear_points():
        self.ax2.clear()

    while 1:
        try:
            if self.exit == True:
                break
            if self.clear == True:
                self.clear = False
                self.initpos = [0, 0]
                x, y, c = ifs( self.ifs_triangle.get_areas(),
                                self.ifs_triangle.get_eqs(), self.initpos, 100 )
                self.initpos = [x[-1], y[-1]]
                self.ax2.clear()

                x, y, c = ifs( self.ifs_triangle.get_areas(),
                                self.ifs_triangle.get_eqs(), self.initpos, ITER_C )
                if np.max(np.abs(x)) < 1000000 and np.max(np.abs(y)) < 1000000:
                    self.initpos = [x[-1], y[-1]]
                    wx.CallAfter( draw_points, x, y, c )
                    time.sleep(0.05)
        except:
            pass

```

IFSTriangles.version IFSDesignerIFSTriangles

```
self.version += 1
```

IFSDesignerversion

```
@on_trait_change("ifs_triangle.version")
def on_ifs_version_changed(self):
    """
    """
    self.clear = True
```

IFSDesigner.clearTrueifs_calculate

L-System

L-System

- **F** :
- **+** :
- **-** :

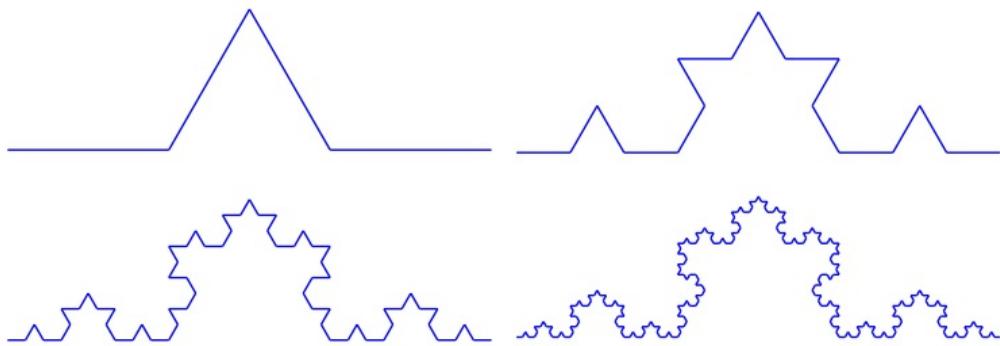
4

F+F--F+F

FF+F--F+F

F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

(+-60)



 F+F--F+F

F, +, -

- **f** :
- **[** :
- **]** :
- **S** :

)

```
S -> X
X -> F-[ [X]+X ]+F[ +FX ]-X
F -> FF
```

```
{
    "X":"F-[ [X]+X ]+F[ +FX ]-X", "F":"FF", "S":"X",
    "direct":-45,
    "angle":25,
    "iter":6,
    "title":"Plant"
}
```

- **direct** :
- **angle** : +,-
- **iter** :

```
class L_System(object):
    def __init__(self, rule):
        info = rule['S']
        for i in range(rule['iter']):
            ninfo = []
            for c in info:
                if c in rule:
                    ninfo.append(rule[c])
                else:
                    ninfo.append(c)
            info = ''.join(ninfo)
        self.rule = rule
        self.info = info

    def get_lines(self):
        d = self.rule['direct']
```

```

a = self.rule['angle']
p = (0.0, 0.0)
l = 1.0
lines = []
stack = []
for c in self.info:
    if c in "Ff":
        r = d * pi / 180
        t = p[0] + l*cos(r), p[1] + l*sin(r)
        lines.append(((p[0], p[1]), (t[0], t[1])))
        p = t
    elif c == "+":
        d += a
    elif c == "-":
        d -= a
    elif c == "[":
        stack.append((p, d))
    elif c == "]":
        p, d = stack[-1]
        del stack[-1]
return lines

```

matplotlibLineCollection

```

import matplotlib.pyplot as pl
from matplotlib import collections

# rule = {...}  rule

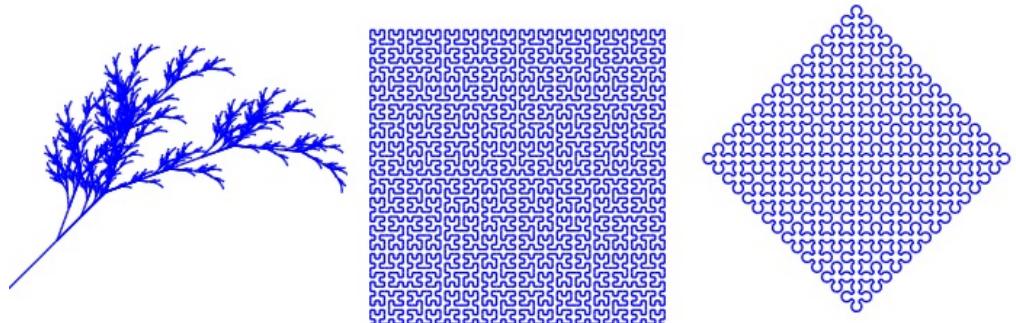
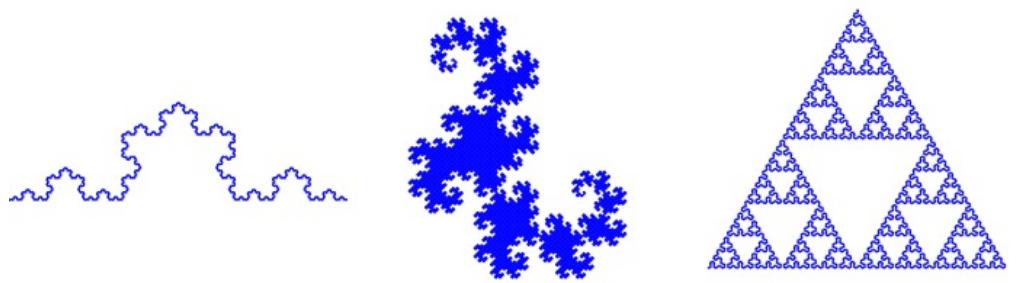
lines = L_System(rule).get_lines()

fig = pl.figure()
ax = fig.add_subplot(111)
linecollections = collections.LineCollection(lines)
ax.add_collection(linecollections, autolim=True)
pl.show()

```

L-System

L-System



 L-System

Python »

← (Alt+X) | |

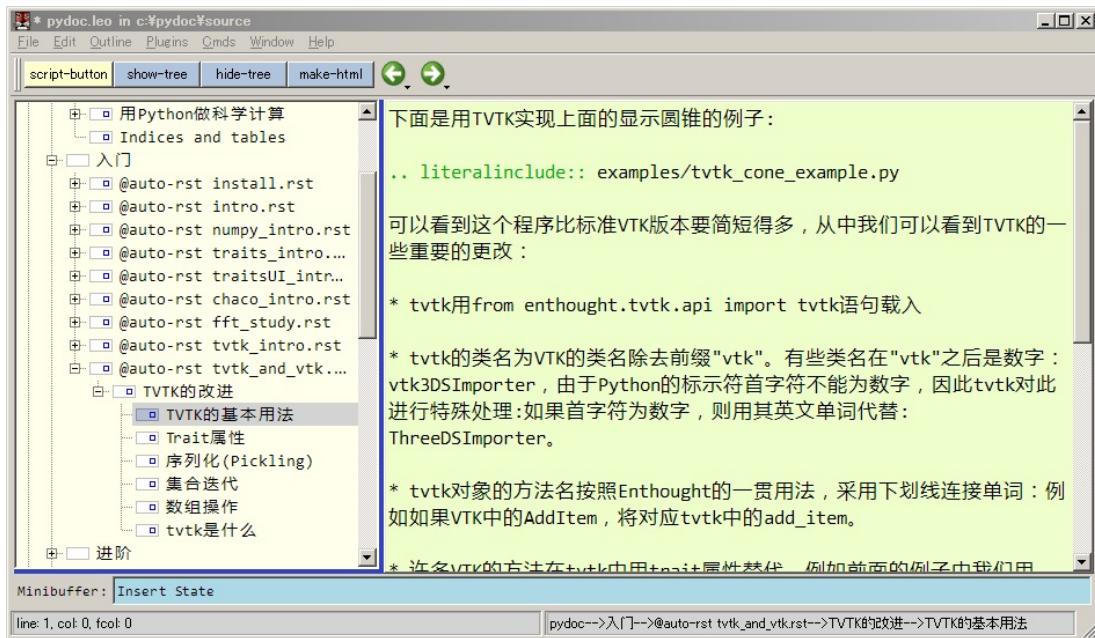
Python »

← (Alt+X) | |

SphinxreStructuredText

reStructuredText(rst) Sphinx reStructuredText html Leo
proText latex PNG

- **reStructuredText** :
- **Sphinx** : reStructuredText cross-references)
reStructuredText SphinxHTML
- **Leo** : rst python Leo



- **PicPick, Greenshot** :

SphinxPygmentsPygmentsconf.py

```
# The name of the Pygments (syntax highlighting) style to use.
pygments_style = 'sphinx'
```

pygmentssphinx

```
%Python%\Lib\site-packages\sphinx\highlighting.py
```

Commentnoitalic

```
...
styles.update({
    Generic.Output: '#333',
    Comment: 'noitalic #408090',
    Number: '#208050',
})
...
```

Sphinx

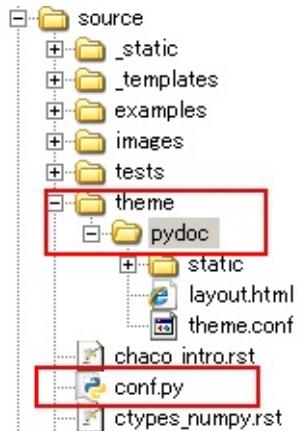
javascriptShpinx

- conf.py

```
# The theme to use for HTML and HTML Help pages. Major theme
# Sphinx are currently 'default' and 'sphinxdoc'.
html_theme = 'pydoc'

# Add any paths that contain custom themes here, relative to
html_theme_path = ["./theme"]
```

- conf.pythemesphinxthemes\phinxdocthemepydoc



theme

- layout.htmlSphinxrsthtmlcssjs:

```

<link type="text/css" href="_static/jquery-ui-1.7.2.custom.css" rel="stylesheet">
<link type="text/css" href="_static/comments.css" rel="stylesheet">
<script type="text/javascript" src="_static/jquery-ui-1.7.2.custom.js"></script>
<script type="text/javascript" src="_static/pydoc.js"></script>

```

- theme\pydoc\staticcssjshtml\theme\pydoc\theme.conf
stickysidebar=True|IE7.0css|IE6.0(Firefox, IE7, Chrome)

```

div.sphinxsidebar{
    position : fixed;
    left : 0px;
    top : 30px;
    margin-left : 0px !important;
}

```

htmlSphinxunicode conf.py

```

html_use_smartytags = False

```

latex

Sphinx latex png windows latex
latexmake.bat

proTeXt tex700M1.3G Sphinx

```
%SPHINXBUILD% -D pngmath_latex="..\latex.exe" -b html %ALLSPHINXTOPICS%
```

latex:

```
X_k = \sum_{n=0}^{N-1} x_n e^{-\{i 2\pi k \frac{n}{N}\}} \qquad k = 0, \dots, N-1.
```

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \dots, N-1.$$

Leo

Leo Leo

- LeotkqttkLeoqtlaunchLeo.py --gui=qt
- Consolas

YaHei Mono http://hyry.dip.jp/files/yahei_mono.7z

- leo\config\leoSettings.leo myLeoSettings.leoLeo
qtGui plugin-->@data qt-gui-plugin-style-sheet Yahei Mono

```
QTextEdit#richTextEdit {  
    ...  
    font-family: Yahei Mono;  
    font-size: 17px;  
    ...  
}
```

- @settings-->Window-->@string initial_split_orientation
 @settings-->Window-->Options for new windows-->@strings[vertical,horizontal] initial_splitter_orientation
 horizontal
- Leo@auto-rstrstrstrrstLeounicode1rst2
Leo rst9Leo6leo\core\leoRst.pyunderline
 len(s)len(ss)

```
def underline (self,s,p):
    ...

    try:
        ss = s.encode("gbk")
    except:
        try:
            ss = s.encode("shiftjis")
        except:
            ss = s

    trace = False and not g.unitTesting
    ...
```

Matplotlib

%PythonPath%\Lib\site-packages\matplotlib\mpl-data\fonts\ttf\

Yahei Mono

Matplotlibmatplotlibrc

%PythonPath%\Lib\site-packages\matplotlib\mpl-data\matplotlibrc

c:\Documents and Settings\%UserName%\matplotlib\matplotlibrc

- font.family: font.family : monospace#
- font.monospace : Yahei Mono

matplotlibunicodeu""

Matplotlib

matplotlibSphinx..plotHTML

```
c:\Python26\Lib\site-packages\matplotlib\sphinxext\plot_directive  
c:\Python26\Lib\site-packages\matplotlib\sphinxext\only_directive
```

sourceextsplot_directive.pyconf.py

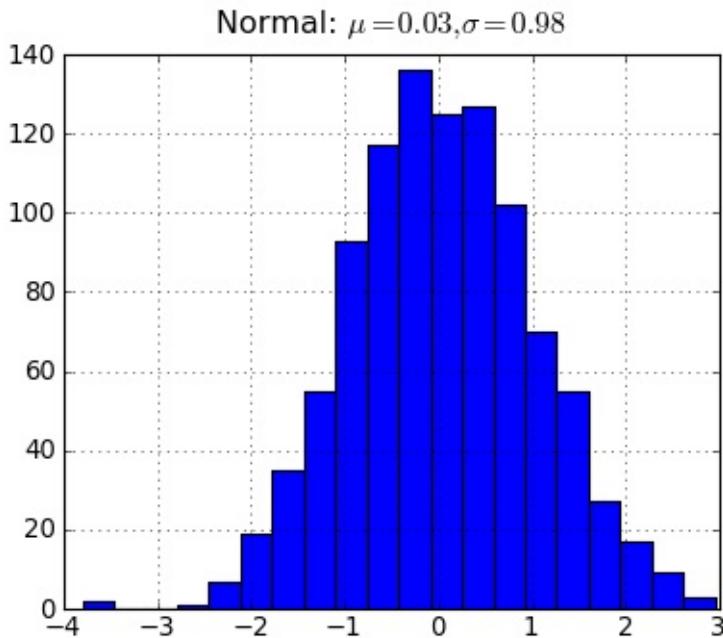
```
sys.path.append(os.path.abspath('exts'))  
extensions = ['sphinx.ext.autodoc', 'sphinx.ext.doctest',  
             'sphinx.ext.pngmath', 'plot_directive']
```

extsplot_directive.pyplot_directive.pyPDF

rst

```
.. plot:: examples/matplotlib_hist.py  
:include-source:
```

```
import matplotlib.pyplot as plt  
import numpy as np  
x = np.random.randn(1000)  
plt.hist(x, 20)  
plt.grid()  
plt.title(r'Normal: $\mu=%.2f, \sigma=%.2f$'%(x.mean(), x.std()))  
plt.show()
```



Graphviz

SphinxGraphvizGraphvizWindowsc:\graphviz

Graphviz <http://www.graphviz.org>

conf.py extensions 'sphinx.ext.graphviz'

make.batdot.exe

```
set SPHINXBUILD=sphinx-build -D graphviz_dot=c:\graphviz\bin\dot..
```

rstGraphviz

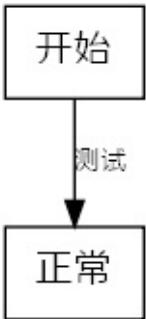
```
.. graphviz::

    digraph GraphvizDemo{
        node [fontname="Yahei Mono" shape="rect"];
        edge [fontname="Yahei Mono" fontsize=10];

        node1[label:@""];
        node2[label:@""];
    }
```

```
    node1->node2[label=""];
}
```

:



CHM

SphinxCHM
make htmlhelp(.hhc)

- sphinxbuildershtmlhelp.pyhtmlhelpcn.pyCHM
- builders__init__.pyBUILTIN_BUILDERS

```
'htmlhelpcn': ('htmlhelpcn', 'HTMLHelpBuilder')
```

- make.bat

```
if "%1" == "htmlhelpcn" (
    %SPHINXBUILD% -b htmlhelpcn %ALLSPHINXOPTS% build/htmlh
    echo.
    echo.Build finished; now you can run HTML Help Workshop
    .hhp project file in build/htmlhelpcn.
    goto end
)
```

- htmlhelpcn.pyproject_templateLanguage
Language=0x804
- make.bat htmlhelpcnhtmlhelpcn.py.encode("gb2312")

```
f.write(item.encode('ascii', 'xmlcharrefreplace'))  
# -->  
f.write(item.encode('gb2312'))
```

- rstCHM
- make.bat htmlhelpcnCHM

```
"C:\Program Files\HTML Help Workshop\hhc.exe" htmlhelpcn\sc
```

CHMFlash

reStructuredText raw htmlFlash

```
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" WIDT  
CODEBASE="http://active.macromedia.com/flash5/cabs/swflash.cab#"  
<PARAM NAME="movie" VALUE="_images/fft_study_04.swf">  
<PARAM NAME="play" VALUE="true">  
<PARAM NAME="loop" VALUE="false">  
<PARAM NAME="wmode" VALUE="transparent">  
<PARAM NAME="quality" VALUE="high">  
<EMBED SRC="_images/fft_study_04.swf" width="589" HEIGHT="447" qu  
loop="false" wmode="transparent" TYPE="application/x-shockwave-f.  
PLUGINSPAGE=  
"http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_</EMBED>  
</OBJECT>
```

Html Help WorkshopswfCHMCHMflashflashhtml

```

```

Html Help Workshopfft_study_04.swfCSS

PDF

make latexlatex xelatex scipydoc.tex PDFxelatexproTeXt

PDF

- conf.py Options for LaTeX output texYahei Mono

```
latex_preamble = r"""
    \usepackage{float}
    \textwidth 6.5in
    \oddsidemargin -0.2in
    \evensidemargin -0.2in
    \usepackage{ccaption}
    \usepackage{fontspec,xunicode,xltextra}
    \setsansfont{Microsoft YaHei}
    \setromanfont{Microsoft YaHei}
    \setmainfont{Microsoft YaHei}
    \setmonofont{Yahei Mono}
    \XeTeXlinebreaklocale "zh"
    \XeTeXlinebreakskip = 0pt plus 1pt
    \renewcommand{\baselinestretch}{1.3}
    \setcounter{tocdepth}{3}
    \captiontitlefont{\small\sffamily}
    \captiondelim{ - }
    \renewcommand\today{\number\year\,\number\month\,\number\day}
    \makeatletter
    \renewcommand*\l@subsection{\@dottedtocline{2}{2.0em}{4.0em}}
    \renewcommand*\l@subsubsection{\@dottedtocline{3}{3em}{5em}}
    \makeatother
    \titleformat{\chapter}[display]
    {\bfseries\Huge}
    {\filleft \Huge \hspace{2 mm} \thechapter \hspace{4 mm} }
    {4ex}
    {\titlerule
    \vspace{2ex}%
    \filright
    [\vspace{2ex}%
    \titlerule]
    \%definecolor{VerbatimBorderColor}{rgb}{0.2,0.2,0.2}
    \%definecolor{VerbatimColor}{rgb}{0.95,0.95,0.95}
"""
.decode("utf-8")
```

renewcommandPDF

latex_preamble\nnewcommand

```
.. raw:: latex

    \renewcommand{\partname}{}
    \renewcommand{\chaptermark}[1]{\markboth{ \thechapter\ \hspace{1em}}{\fancyhead[LE,RO]{Python}}}
    \renewcommand{\figurename}{\textsc{}}
```

- conf.py

```
latex_paper_size = 'a4'
latex_font_size = '11pt'
latex_use_modindex = False
```

- PDFnonstopmode

```
xelatex -interaction=nonstopmode scipydoc.tex
```

latexreStructuredTextPythontexlatex

- begin{figure}[htbp]begin{figure}[H]tex
- \tableofcontents\renewcommand\contentsname{ }

PDF

PDFPDFcover.pdf

PDF <http://www.softinterface.com>

PDFcover.pdfPDF

PDFPDFsam <http://www.pdfsam.org>

PDFsamPDFPDF

```
set MERGE=java -jar "c:\Program Files\pdfsam\lib\pdfsam-console-2
%MERGE% -f cover.pdf -f scipydoc.pdf -o %CD%\scipydoc2.pdf concat
```

- -fPDF
- -oPDF%CD%

zip, chm, pdf

```
rename html scipydoc
"c:\Program Files\7-Zip\7z.exe" a scipydoc.zip scipydoc
rename scipydoc html

"C:\Program Files\HTML Help Workshop\hhc.exe" htmlhelpcn\scipydoc
copy htmlhelpcn\scipydoc.chm . /y

cd latex
xelatex -interaction=nonstopmode scipydoc.tex
cd ..
copy latex\scipydoc.pdf . /y
```

HTML

Sphinxsearchindex.jsSphinxchinese_search.pysmallseg

smallseg: <http://code.google.com/p/smallseg>

```
from os import path

import re
import cPickle as pickle

from docutils.nodes import comment, Text, NodeVisitor, SkipNode

from sphinx.util.stemmer import PorterStemmer
from sphinx.util import jsdump, rpartition

from smallseg import SEG

DEBUG = False
```

```

word_re = re.compile(r'\w+(?u)')

stopwords = set("""
a and are as at
be but by
for
if in into is it
near no not
of on or
such
that the their then there these they this to
was will with
""".split())

if DEBUG:
    testfile = file("testfile.txt", "wb")

class _JavaScriptIndex(object):
    """
    The search index as javascript file that calls a function
    on the documentation search object to register the index.
    """

    PREFIX = 'Search.setIndex('
    SUFFIX = ')'

    def dumps(self, data):
        return self.PREFIX + jsdump.dumps(data) + self.SUFFIX

    def loads(self, s):
        data = s[len(self.PREFIX):-len(self.SUFFIX)]
        if not data or not s.startswith(self.PREFIX) or not \
           s.endswith(self.SUFFIX):
            raise ValueError('invalid data')
        return jsdump.loads(data)

    def dump(self, data, f):
        f.write(self.dumps(data))

    def load(self, f):
        return self.loads(f.read())

js_index = _JavaScriptIndex()

```

```

class Stemmer(PorterStemmer):
    """
    All those porter stemmer implementations look hideous.
    make at least the stem method nicer.
    """

    def stem(self, word):
        word = word.lower()
        return word
        #return PorterStemmer.stem(self, word, 0, len(word) - 1)

class WordCollector(NodeVisitor):
    """
    A special visitor that collects words for the `IndexBuilder`.
    """

    def __init__ (self, document):
        NodeVisitor.__init__ (self, document)
        self.found_words = []

    def dispatch_visit (self, node):
        if node.__class__ is comment:
            raise SkipNode
        if node.__class__ is Text:
            words = seg.cut(node.astext()).encode("utf8"))
            words.reverse()
            self.found_words.extend(words)

class IndexBuilder(object):
    """
    Helper class that creates a searchindex based on the doctrees
    passed to the `feed` method.
    """

    formats = {
        'jsdump': jsdump,
        'pickle': pickle
    }

    def __init__ (self, env):
        self.env = env
        self.stemmer = Stemmer()
        # filename -> title
        self.titles = {}
        # stemmed word -> set(filenames)
        self.mapping = {}
        # desctypes -> index

```

```

self._desctypes = {}

def load(self, stream, format):
    """Reconstruct from frozen data."""
    if isinstance(format, basestring):
        format = self.formats[format]
    frozen = format.load(stream)
    # if an old index is present, we treat it as not existing
    if not isinstance(frozen, dict):
        raise ValueError('old format')
    index2fn = frozen['filenames']
    self._titles = dict(zip(index2fn, frozen['titles']))
    self._mapping = {}
    for k, v in frozen['terms'].iteritems():
        if isinstance(v, int):
            self._mapping[k] = set([index2fn[v]])
        else:
            self._mapping[k] = set(index2fn[i] for i in v)
    # no need to load keywords/desctypes

def dump(self, stream, format):
    """Dump the frozen index to a stream."""
    if isinstance(format, basestring):
        format = self.formats[format]
    format.dump(self.freeze(), stream)

def get_modules(self, fn2index):
    rv = {}
    for name, (doc, _, _, _) in self.env.modules.iteritems():
        if doc in fn2index:
            rv[name] = fn2index[doc]
    return rv

def get_descrefs(self, fn2index):
    rv = {}
    dt = self._desctypes
    for fullname, (doc, desctype) in self.env.descrefs.iteritems():
        if doc not in fn2index:
            continue
        prefix, name = rpartition(fullname, '.')
        pdict = rv.setdefault(prefix, {})
        try:
            i = dt[desctype]
        except KeyError:
            i = len(dt)
            dt[desctype] = i
        pdict[name] = (fn2index[doc], i)

```

```

    return rv

def get_terms(self, fn2index):
    rv = []
    for k, v in self._mapping.iteritems():
        if len(v) == 1:
            fn, = v
            if fn in fn2index:
                rv[k] = fn2index[fn]
        else:
            rv[k] = [fn2index[fn] for fn in v if fn in fn2index]
    return rv

def freeze(self):
    """Create a usable data structure for serializing."""
    filenames = self._titles.keys()
    titles = self._titles.values()
    fn2index = dict((f, i) for (i, f) in enumerate(filenames))
    return dict(
        filenames=filenames,
        titles=titles,
        terms=self.get_terms(fn2index),
        desrefs=self.get_desrefs(fn2index),
        modules=self.get_modules(fn2index),
        desctypes=dict((v, k) for (k, v) in self._desctypes.items())
    )

def prune(self, filenames):
    """Remove data for all filenames not in the list."""
    new_titles = {}
    for filename in filenames:
        if filename in self._titles:
            new_titles[filename] = self._titles[filename]
    self._titles = new_titles
    for wordnames in self._mapping.itervalues():
        wordnames.intersection_update(filenames)

def feed(self, filename, title, doctree):
    """Feed a doctree to the index."""
    self._titles[filename] = title

    visitor = WordCollector(doctree)
    doctree.walk(visitor)

    def add_term(word, prefix='', stem=self._stemmer.stem()):
        word = stem(word)
        word = word.strip(u"!@#$%^&*()_+-*/\\\";,.[]{}<>")

```

```

        if len(word) <= 1: return
        if word.encode("utf8").isalpha() and len(word) < 3: return
        if word.isdigit(): return
        if word in stopwords: return

    try:
        float(word)
        return
    except:
        pass

    if DEBUG:
        testfile.write("%s\n" % word.encode("utf8"))
    self._mapping.setdefault(prefix + word, set()).add(word)

words = seg.cut(title.encode("utf8"))
for word in words:
    add_term(word)

for word in visitor.found_words:
    add_term(word)

def load_indexer(self):
    def func(docnames):
        print "##### CHINESE INDEXER #####"
        self.indexer = IndexBuilder(self.env)
        keep = set(self.env.all_docs) - set(docnames)
        try:
            f = open(path.join(self.outdir, self.searchindex_file))
            try:
                self.indexer.load(f, self.indexer_format)
            finally:
                f.close()
        except (IOError, OSError, ValueError):
            if keep:
                self.warn('search index couldn\'t be loaded, but '
                          'documents will be built: the index will be '
                          'incomplete.')
        # delete all entries for files that will be rebuilt
        self.indexer.prune(keep)
    return func

def builder_initiated(app):
    if app.builder.name == 'html':
        print "*****"
        global seg

```

```

        seg = SEG()
        app.builder.load_indexer = load_indexer(app.builder)

def setup(app):
    app.connect('builder-inited', builder_inited)

```

PDF

Sphinx\text{\label}\ref{label_tex_ref}.pyPDFrole\label, tref,
tpageref\label, \ref, \pageref

```

# -*- coding: utf-8 -*-
from docutils import nodes, utils

class tref(nodes.Inline, nodes.TextElement):
    pass

class tlabel(nodes.Inline, nodes.TextElement):
    pass

class tpageref(nodes.Inline, nodes.TextElement):
    pass

def tref_role(role, rawtext, text, lineno, inliner, options={}, content=[]):
    data = text.split(',')
    if u"" in data[0]:
        name = u""
        pos = data[0][0]
        ref = data[1]
        return [tref(name=name, ref=ref, pos=pos)], []
    return [], []
def tlabel_role(role, rawtext, text, lineno, inliner, options={}, content=[]):
    return [tlabel(latex=text)], []

def tpageref_role(role, rawtext, text, lineno, inliner, options={}, content=[]):
    return [tpageref(latex=text)], []

def latex_visit_ref(self, node):
    self.body.append(r"%s\ref{%s}" % (node['name'], node['ref']))
    raise nodes.SkipNode

def html_visit_ref(self, node):

```

```
    self.body.append(r'<a href="#%s">%s%s</a>' % (node['ref'], no
raise nodes.SkipNode

def latex_visit_label(self, node):
    self.body.append(r"\label{\%s}" % node['latex'])
    raise nodes.SkipNode

def latex_visit_pageref(self, node):
    self.body.append(r"\pageref{\%s}" % node['latex'])
    raise nodes.SkipNode

def empty_visit(self, node):
    raise nodes.SkipNode

def setup(app):
    app.add_node(tref, latex=(latex_visit_ref, None), text=(empty_v
    app.add_node(tlabel, latex=(latex_visit_label, None), text=(emp
    app.add_node(tpageref, latex=(latex_visit_pageref, None), text=

    app.add_role('tref', tref_role)
    app.add_role('tlabel', tlabel_role)
    app.add_role('tpageref', tpageref_role)
```

ReST

figure

```
.. _pythonxyhome:  
.. figure:: images/pythonxy_home.png  
    Python(x,y)
```

PDF

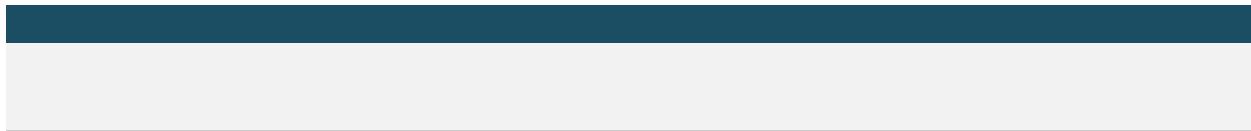
figurefigwidthalignlatexwrapfigure

```
\  
.. _fig-traitsUIintro03:  
.. figure:: images/traitsUI_intro_03.png  
    :figwidth: 4cm  
    :align: left  
  
    labeltooltip
```

rstrstLeohtml

Sphinx

```
.. literalinclude:: examples/tvtk_cone.example.py  
.. literalinclude:: example.c  
    :language: c
```



Leo

LeomySettings.leoshow-treehide-treeshow-tree
1:3hide-tree

```
@button show tree
    g.es(c.frame.resizePanesToRatio(0.25,0.7))

@button hide tree
    g.es(c.frame.resizePanesToRatio(0.0,0.7))
```

MatplotlibPDF

plot_directive.py latex

PDFBookmarks

PDFBookmarkstex

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

- 2010/01/15: [Mayavi](#)

- 2010/01/14: [IIR](#)

- 2010/01/12: Sphinx

smallseg <http://code.google.com/p/smallseg>

- 2010/01/07:

- 2010/01/05: [Sympy](#) [NumPy](#)-

- 2010/01/04: [L-System](#)

- 2010/01/03:

- 2010/01/02: [\(IFS\)](#)

- 2009/12/30 : [MatplotlibAxis](#)

- 2009/12/29 : [Mandelbrot](#)

Python »

← (Alt+X) | |

Python »

← (Alt+X) | |

- FFT
- traitsUImatplotlib
- CSV
- NLMS
-
- hann
- FFT
-
-
-
- Mandelbrot
-
- L-System

Python »

← (Alt+X) | |

Python >> >>

← (Alt+X) | |

FFT

FFT

FFTFFT

```
# -*- coding: utf-8 -*-
from enthought.traits.api import \
    Str, Float, HasTraits, Property, cached_property, Range, Instance

from enthought.chaco.api import Plot, AbstractPlotData, ArrayPlot

from enthought.traits.ui.api import \
    Item, View, VGroup, HSplit, ScrubberEditor, VSplit

from enthought.enable.api import Component, ComponentEditor
from enthought.chaco.tools.api import PanTool, ZoomTool
```

```

import numpy as np

#
scrubber = ScrubberEditor(
    hover_color = 0xFFFFFFFF,
    active_color = 0xA0CD9E,
    border_color = 0x808080
)

# FFTfreqsnloops
def fft_combine(freqs, n, loops=1):
    length = len(freqs) * loops
    data = np.zeros(length)
    index = loops * np.arange(0, length, 1.0) / length * (2 * np.pi)
    for k, p in enumerate(freqs[:n]):
        if k != 0: p *= 2 # *2
        data += np.real(p) * np.cos(k*index) #
        data -= np.imag(p) * np.sin(k*index) #
    return index, data

class TriangleWave(HasTraits):
    # Rangetraits
    # trait
    low = Float(0.02)
    hi = Float(1.0)

    #
    wave_width = Range("low", "hi", 0.5)

    # Cx
    length_c = Range("low", "wave_width", 0.5)

    # y
    height_c = Float(1.0)

    # FFTenum
    fftsize = Enum([(2**x) for x in range(6, 12)])

    # FFTx
    fft_graph_up_limit = Range(0, 400, 20)

    # FFT
    peak_list = Str

    #
    N = Range(1, 40, 4)

```

```

#
plot_data = Instance(AbstractPlotData)

#
plot_wave = Instance(Component)

# FFT
plot_fft = Instance(Component)

#
container = Instance(Component)

#
view = View(
    HSplit(
        VSplit(
            VGroup(
                Item("wave_width", editor = scrubber, label=u"Wave Width"),
                Item("length_c", editor = scrubber, label=u"X Length"),
                Item("height_c", editor = scrubber, label=u"Y Height"),
                Item("fft_graph_up_limit", editor = scrubber),
                Item("fftsize", label=u"FFT Size"),
                Item("N", label=u"Number of Points")
            ),
            Item("peak_list", style="custom", show_label=False)
        ),
        VGroup(
            Item("container", editor=ComponentEditor(size=(600, 400)),
                 orientation = "vertical")
        )
    ),
    resizable = True,
    width = 800,
    height = 600,
    title = u"FFT"
)

#
#
def _create_plot(self, data, name, type="line"):
    p = Plot(self.plot_data)
    p.plot(data, name=name, title=name, type=type)
    p.tools.append(PanTool(p))
    zoom = ZoomTool(component=p, tool_mode="box", always_on=True)
    p.overlays.append(zoom)
    p.title = name
    return p

```

```

def __init__(self):
    #
    super(TriangleWave, self).__init__()

    # Plot
    self.plot_data = ArrayPlotData(x=[], y=[], f=[], p=[], x2:)

    #
    self.container = VPlotContainer()

    # (x,y)(x2,y2)
    self.plot_wave = self._create_plot(("x","y"), "Triangle Wave")
    self.plot_wave.plot(("x2","y2"), color="red")

    # fp
    self.plot_fft = self._create_plot(("f","p"), "FFT", type="line")

    #
    self.container.add( self.plot_wave )
    self.container.add( self.plot_fft )

    #
    self.plot_wave.x_axis.title = "Samples"
    self.plot_fft.x_axis.title = "Frequency pins"
    self.plot_fft.y_axis.title = "(dB)"

    # fftsize1024Enum
    self.fftsize = 1024

# FFTX
def _fft_graph_up_limit_changed(self):
    self.plot_fft.x_axis.mapper.range.high = self.fft_graph_up_limit

def _N_changed(self):
    self.plot_sin_combine()

# trait@on_trait_change
@on_trait_change("wave_width, length_c, height_c, fftsize")
def update_plot(self):
    #
    global y_data
    x_data = np.arange(0, 1.0, 1.0/self.fftsize)
    func = self.triangle_func()
    # funcfloat64
    y_data = np.cast["float64"](func(x_data))

```

```

#
fft_parameters = np.fft.fft(y_data) / len(y_data)

#
fft_data = np.clip(20*np.log10(np.abs(fft_parameters))[:self.fftsize]

#
self.plot_data.set_data("x", np.arange(0, self.fftsize))
self.plot_data.set_data("y", y_data)
self.plot_data.set_data("f", np.arange(0, len(fft_data)))
self.plot_data.set_data("p", fft_data)

# x2
self.plot_data.set_data("x2", np.arange(0, 2*self.fftsize

# x
self._fft_graph_up_limit_changed()

# -80dB
peak_index = (fft_data > -80)
peak_value = fft_data[peak_index][:20]
result = []
for f, v in zip(np.flatnonzero(peak_index), peak_value):
    result.append("%s : %s" %(f, v))
self.peak_list = "\n".join(result)

# fft
self.fft_parameters = fft_parameters
self.plot_sin_combine()

# 2
def plot_sin_combine(self):
    index, data = fft_combine(self.fft_parameters, self.N, 2)
    self.plot_data.set_data("y2", data)

# ufunc
def triangle_func(self):
    c = self.wave_width
    c0 = self.length_c
    hc = self.height_c

    def trifunc(x):
        x = x - int(x) # 1x
        if x >= c: r = 0.0
        elif x < c0: r = x / c0 * hc
        else: r = (c-x) / (c-c0) * hc
        return r

```

```
# trifuncufunc,
# Object
return np.frompyfunc(trifunc, 1, 1)

if __name__ == "__main__":
    triangle = TriangleWave()
    triangle.configure_traits()
```

Python » »

← (Alt+X) | |

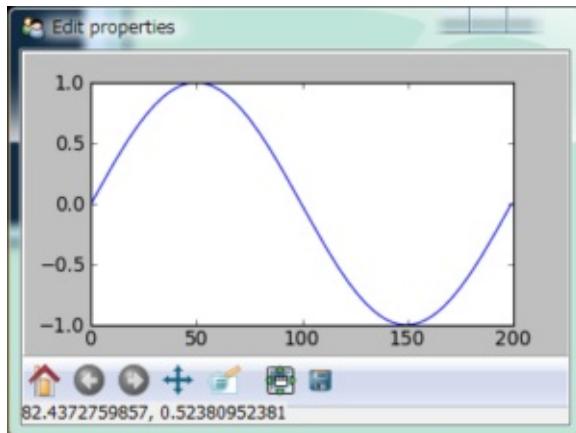
Python >> >>

← (Alt+X) | |

traitsUImatplotlib

Trait

traitsUImatplotlib



```
# -*- coding: utf-8 -*-
# file name: mpl_figure_editor.py
import wx
import matplotlib
# matplotlibWXAggwxtraitsUI
matplotlib.use("WXAgg")
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigureCanvas
from matplotlib.backends.backend_wx import NavigationToolbar2Wx
from enthought.traits.ui.wx.editor import Editor
from enthought.traits.ui.basic_editor_factory import BasicEditorFactory

class _MPLFigureEditor(Editor):
    """
    WX
    """
    scrollable = True

    def init(self, parent):
        self.control = self._create_canvas(parent)
        self.set_tooltip()
        print dir(self.item)

    def update_editor(self):
        pass
```

```

def _create_canvas(self, parent):
    """
    Panel, BoxSizer, panel
    FigureCanvas, NavigationToolbar2Wx, StaticText
    FigureCanvasmousemovedStaticText

    """
    panel = wx.Panel(parent, -1, style=wx.CLIP_CHILDREN)
    def mousemoved(event):
        panel.info.SetLabel("%s, %s" % (event.xdata, event.yd))
    panel.mousemoved = mousemoved
    sizer = wx.BoxSizer(wx.VERTICAL)
    panel.SetSizer(sizer)
    mpl_control = FigureCanvas(panel, -1, self.value)
    mpl_control.mpl_connect("motion_notify_event", mousemoved)
    toolbar = NavigationToolbar2Wx(mpl_control)
    sizer.Add(mpl_control, 1, wx.LEFT | wx.TOP | wx.GROW)
    sizer.Add(toolbar, 0, wx.EXPAND|wx.RIGHT)
    panel.info = wx.StaticText(parent, -1)
    sizer.Add(panel.info)

    self.value.canvas.SetMinSize((10,10))
    return panel

class MPLFigureEditor(BasicEditorFactory):
    """
    traits.uiEditorFactory
    """
    klass = _MPLFigureEditor

    if __name__ == "__main__":
        from matplotlib.figure import Figure
        from enthought.traits.api import HasTraits, Instance
        from enthought.traits.ui.api import View, Item
        from numpy import sin, cos, linspace, pi

        class Test(HasTraits):
            figure = Instance(Figure, ())
            view = View(
                Item("figure", editor=MPLFigureEditor(), show_label=False),
                width = 400,
                height = 300,
                resizable = True)
            def __init__(self):
                super(Test, self).__init__()
                axes = self.figure.add_subplot(111)
                t = linspace(0, 2*pi, 200)

```

```
axes.plot(sin(t))  
Test().configure_traits()
```

Python >> »

← (Alt+X) | |

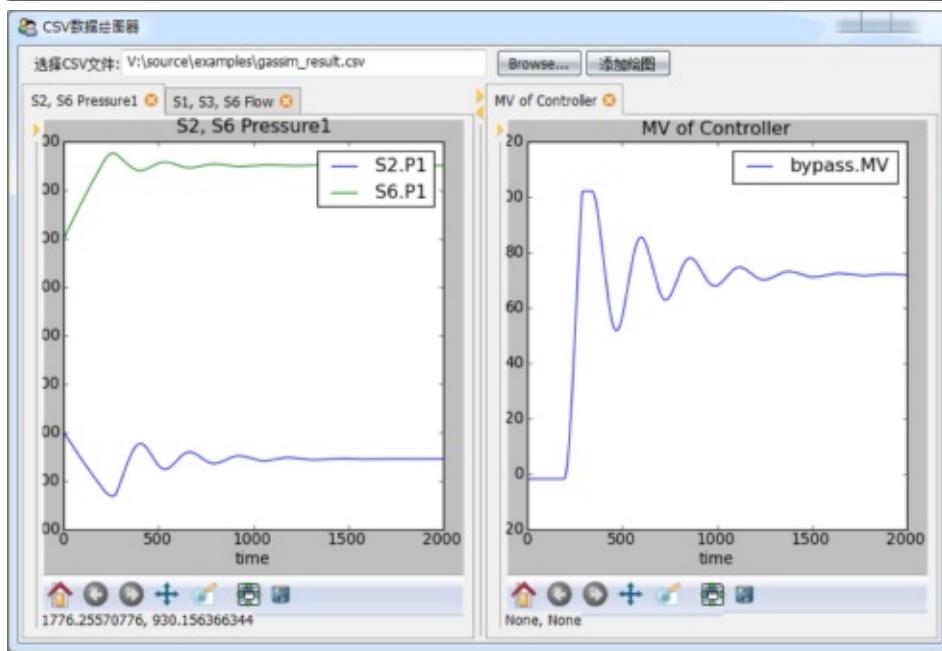
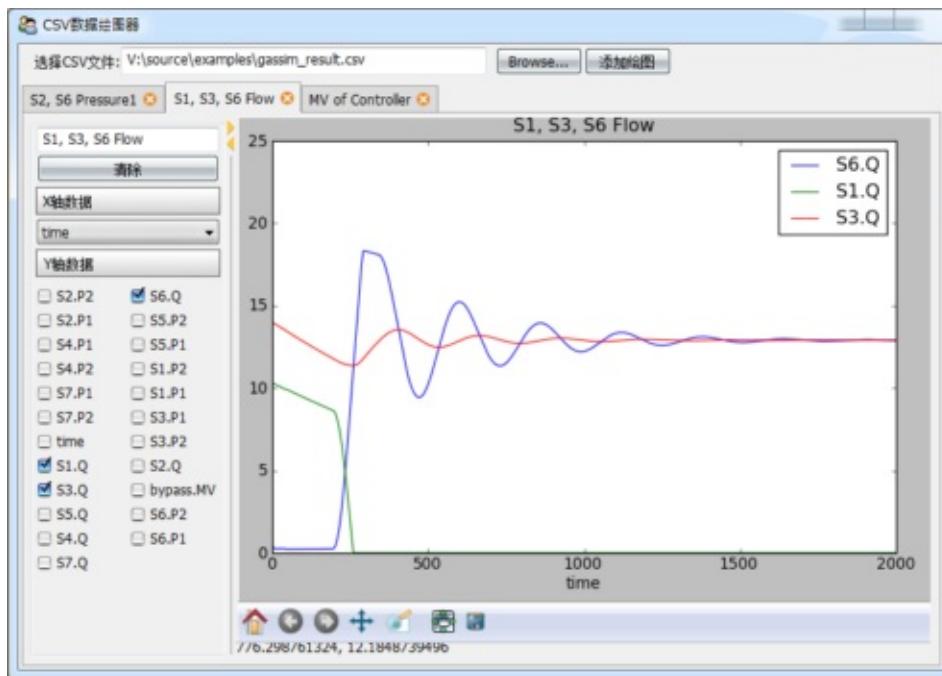
Python >> >>

← (Alt+X) | |

CSV

Trait

traitsUImatplotlib CSV



```
# -*- coding: utf-8 -*-
```

```
from matplotlib.figure import Figure
from mpl_figure_editor import MPLFigureEditor
from enthought.traits.ui.api import *
from enthought.traits.api import *
import csv

class DataSource(HasTraits):
    """
    data
    namesdata
    """
    data = DictStrAny
    names = List(Str)

    def load_csv(self, filename):
        """
        CSVdata[names]
        """
        f = file(filename)
        reader = csv.DictReader(f)
        self.names = reader.fieldnames
        for field in reader.fieldnames:
            self.data[field] = []
        for line in reader:
            for k, v in line.iteritems():
                self.data[k].append(float(v))
        f.close()

class Graph(HasTraits):
    """
    """
    name = Str # 
    data_source = Instance(DataSource) #
    figure = Instance(Figure) # Figure
    selected_xaxis = Str # X
    selected_items = List # Y

    clear_button = Button(u"\u2605") # Y

    view = View(
        HSplit( # HSplit
            #
            VGroup(
                Item("name"), #
                Item("clear_button"), #
                Heading(u"\u2605"), #

```

```

# XEnumEditorComboBox
# data_sourcenames
Item("selected_xaxis", editor=
    EnumEditor(name="object.data_source.names", f
Heading(u"Y" ), #
# YYCheckBox
Item("selected_items", style="custom",
    editor=CheckListEditor(name="object.data_sou
        cols=2, format_str=u"%s")),
show_border = True, #
scrollable = True, #
show_labels = False #
),
#
Item("figure", editor=MPLFigureEditor(), show_label=F
)
)

def _name_changed(self):
"""
"""

axe = self.figure.axes[0]
axe.set_title(self.name)
self.figure.canvas.draw()

def _clear_button_fired(self):
"""
"""

self.selected_items = []
self.update()

def _figure_default(self):
"""
figureFigure
"""

figure = Figure()
figure.add_axes([0.05, 0.1, 0.9, 0.85]) #
return figure

def _selected_items_changed(self):
"""
Y
"""

self.update()

```

```

def _selected_xaxis_changed(self):
    """
    X
    """
    self.update()

def update(self):
    """
    """
    axe = self.figure.axes[0]
    axe.clear()
    try:
        xdata = self.data_source.data[self.selected_xaxis]
    except:
        return
    for field in self.selected_items:
        axe.plot(xdata, self.data_source.data[field], label=field)
    axe.set_xlabel(self.selected_xaxis)
    axe.set_title(self.name)
    axe.legend()
    self.figure.canvas.draw()

class CSVGrapher(HasTraits):
    """
    """

    graph_list = List(Instance(Graph)) #
    data_source = Instance(DataSource) #
    csv_file_name = File(filter=[u"*.csv"]) #
    add_graph_button = Button(u"" ) #

    view = View(
        #
        VGroup(
            # HGroup
            HGroup(
                #
                Item("csv_file_name", label=u"CSV" , width=400
                #
                Item("add_graph_button", show_label=False)
            ),
            # ListEditor
            Item("graph_list", style="custom", show_label=False,
                  editor=ListEditor(
                      use_notebook=True, #

```

```
        deletable=True, #
        dock_style="tab", # dock
        page_name=".name") # Graphname
    )
),
resizable = True,
height = 0.8,
width = 0.8,
title = u"CSV"
)

def _csv_file_name_changed(self):
"""
DataSource
"""
self.data_source = DataSource()
self.data_source.load_csv(self.csv_file_name)
del self.graph_list[:]

def _add_graph_button_changed(self):
"""

"""
if self.data_source != None:
    self.graph_list.append( Graph(data_source = self.data_source))

if __name__ == "__main__":
    csv_grapher = CSVGrapher()
    csv_grapher.configure_traits()
```

Python » »

←(Alt+X) | |

Python >> >>

← (Alt+X) | |

NLMS

NLMS

NLMS

```
# -*- coding: utf-8 -*-
# filename: nlms_test.py

import numpy as np
import pylab as pl
import nlms_numpy
import scipy.signal

# FIRlength delay
def make_path(delay, length):
    path_length = length - delay
    h = np.zeros(length, np.float64)
    h[delay:] = np.random.standard_normal(path_length) * np.exp( -np.pi*delay*(path_length-delay)/length )
    h /= np.sqrt(np.sum(h*h))
    return h

def plot_converge(y, u, label=""):
    size = len(u)
    avg_number = 200
    e = np.power(y[:size] - u, 2)
    tmp = e[:int(size/avg_number)*avg_number]
    tmp.shape = -1, avg_number
    avg = np.average( tmp, axis=1 )
    pl.plot(np.linspace(0, size, len(avg)), 10*np.log10(avg), label=label)

def diff_db(h0, h):
    return 10*np.log10(np.sum((h0-h)*(h0-h)) / np.sum(h0*h0))

# NLMS, h0, x
def sim_system_identify(nlms, x, h0, step_size, noise_scale):
    y = np.convolve(x, h0)
    d = y + np.random.standard_normal(len(y)) * noise_scale #
    h = np.zeros(len(h0), np.float64) # 0
    u = nlms( x, d, h, step_size )
    return y, u, h

def system_identify_test1():
```

```

h0 = make_path(32, 256) #
x = np.random.standard_normal(10000) #
y, u, h = sim_system_identify(nlms_numpy.nlms, x, h0, 0.5, 0.1)
print diff_db(h0, h)
pl.figure(figsize=(8, 6))
pl.subplot(211)
pl.subplots_adjust(hspace=0.4)
pl.plot(h0, c="r")
pl.plot(h, c="b")
pl.title(u"")
pl.subplot(212)
plot_converge(y, u)
pl.title(u"")
pl.xlabel("Iterations (samples)")
pl.ylabel("Converge Level (dB)")
pl.show()

def system_identify_test2():
    h0 = make_path(32, 256) #
    x = np.random.standard_normal(20000) #
    pl.figure(figsize=(8,4))
    for step_size in np.arange(0.1, 1.0, 0.2):
        y, u, h = sim_system_identify(nlms_numpy.nlms, x, h0, step_size)
        plot_converge(y, u, label=u" $\mu=%s$ " % step_size)
    pl.title(u"")
    pl.xlabel("Iterations (samples)")
    pl.ylabel("Converge Level (dB)")
    pl.legend()
    pl.show()

def system_identify_test3():
    h0 = make_path(32, 256) #
    x = np.random.standard_normal(20000) #
    pl.figure(figsize=(8,4))
    for noise_scale in [0.05, 0.1, 0.2, 0.4, 0.8]:
        y, u, h = sim_system_identify(nlms_numpy.nlms, x, h0, 0.5, noise_scale)
        plot_converge(y, u, label=u"noise=%s" % noise_scale)
    pl.title(u"")
    pl.xlabel("Iterations (samples)")
    pl.ylabel("Converge Level (dB)")
    pl.legend()
    pl.show()

def sim_signal_equation(nlms, x, h0, D, step_size, noise_scale):
    d = x[:-D]
    x = x[D:]
    y = np.convolve(x, h0)[:len(x)]

```

```

h = np.zeros(2*len(h0)+2*D, np.float64)
y += np.random.standard_normal(len(y)) * noise_scale
u = nlms(y, d, h, step_size)
return h

def signal_equation_test1():
    h0 = make_path(5, 64)
    D = 128
    length = 20000
    data = np.random.standard_normal(length+D)
    h = sim_signal_equation(nlms_numpy.nlms, data, h0, D, 0.5, 0.1)
    pl.figure(figsize=(8,4))
    pl.plot(h0, label=u"")
    pl.plot(h, label=u"")
    pl.plot(np.convolve(h0, h), label=u"")
    pl.title(u"")
    pl.legend()
    w0, H0 = scipy.signal.freqz(h0, worN = 1000)
    w, H = scipy.signal.freqz(h, worN = 1000)
    pl.figure(figsize=(8,4))
    pl.plot(w0, 20*np.log10(np.abs(H0)), w, 20*np.log10(np.abs(H)))
    pl.title(u"")
    pl.xlabel(u"")
    pl.ylabel(u"(dB)")
    pl.show()

signal_equation_test1()

```

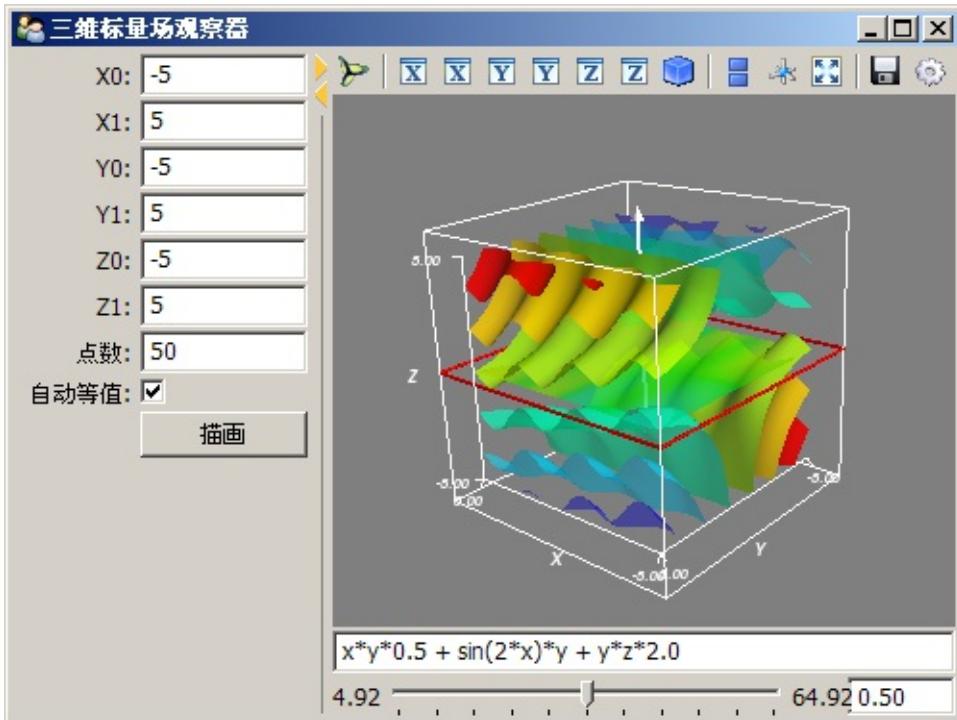
Python » »

← (Alt+X) | |

Python >> >>

← (Alt+X) | |

Mayavi



```
# -*- coding: utf-8 -*-
import numpy as np
from numpy import *

from enthought.traits.api import *
from enthought.traits.ui.api import *
from enthought.tvtk.pyface.scene_editor import SceneEditor
from enthought.mayavi.tools.mlab_scene_model import MlabSceneMode
from enthought.mayavi.core.ui.mayavi_scene import MayaviScene

class FieldViewer(HasTraits):
    """
    #
    x0, x1 = Float(-5), Float(5)
    y0, y1 = Float(-5), Float(5)
    z0, z1 = Float(-5), Float(5)
    points = Int(50) #
    autocontour = Bool(True) #
    v0, v1 = Float(0.0), Float(1.0) #
```

```

contour = Range("v0", "v1", 0.5) #
function = Str("x*x*0.5 + y*y + z*z*2.0") #
plotbutton = Button(u"" )
scene = Instance(MlabSceneModel, ()) # mayavi

view = View(
    HSplit(
        VGroup(
            "x0", "x1", "y0", "y1", "z0", "z1",
            Item('points', label=u"" ),
            Item('autocontour', label=u"" ),
            Item('plotbutton', show_label=False),
        ),
        VGroup(
            Item(name='scene',
                editor=SceneEditor(scene_class=MayaviScene),
                resizable=True,
                height=300,
                width=350
            ), 'function',
            Item('contour',
                editor=RangeEditor(format="%1.2f",
                    low_name="v0", high_name="v1")
            ), show_labels=False
        )
    ),
    width = 500, resizable=True, title=u""
)

def _plotbutton_fired(self):
    self.plot()

def _autocontour_changed(self):
    """
    if hasattr(self, "g"):
        self.g.contour.auto_contours = self.autocontour
    if not self.autocontour:
        self._contour_changed()

def _contour_changed(self):
    """
    if hasattr(self, "g"):
        if not self.g.contour.auto_contours:
            self.g.contour.contours = [self.contour]

def plot(self):
    """

```

```
#  
x, y, z = mgrid[  
    self.x0:self.x1:1j*self.points,  
    self.y0:self.y1:1j*self.points,  
    self.z0:self.z1:1j*self.points]  
scalars = eval(self.function) #  
self.scene.mlab.clf() #  
  
#  
g = self.scene.mlab.contour3d(x, y, z, scalars, contours=10)  
g.contour.auto_contours = self.autocontour  
self.scene.mlab.axes() #  
  
# X-Y  
s = self.scene.mlab.pipeline.scalar_cut_plane(g)  
cutpoint = (self.x0+self.x1)/2, (self.y0+self.y1)/2, (self.z0+self.z1)/2  
s.implicit_plane.normal = (0,0,1) # x cut  
s.implicit_plane.origin = cutpoint  
  
self.g = g  
self.scalars = scalars  
#  
self.v0 = np.min(scalars)  
self.v1 = np.max(scalars)  
  
app = FieldViewer()  
app.configure_traits()
```

Python » »

←(Alt+X) | |

Python >> >>

← (Alt+X) | |

hann

8kHz200Hz 300Hz512FFThann

```
# -*- coding: utf-8 -*-
#hann
#
import numpy as np
import pylab as pl
import scipy.signal as signal

sampling_rate = 8000
fft_size = 512
t = np.arange(0, 1.0, 1.0/sampling_rate)
x = np.sin(2*np.pi*200*t) + 2*np.sin(2*np.pi*300*t)

xs = x[:fft_size]
ys = xs * signal.hann(fft_size, sym=0)

xf = np.fft.rfft(xs)/fft_size
yf = np.fft.rfft(ys)/fft_size
freqs = np.linspace(0, sampling_rate/2, fft_size/2+1)
xfp = 20*np.log10(np.clip(np.abs(xf), 1e-20, 1e100))
yfp = 20*np.log10(np.clip(np.abs(yf), 1e-20, 1e100))
pl.figure(figsize=(8,4))
pl.title(u"200Hz300Hz")
pl.plot(freqs, xfp, label=u"")
pl.plot(freqs, yfp, label=u"hann")
pl.legend()
pl.xlabel(u"(Hz)")

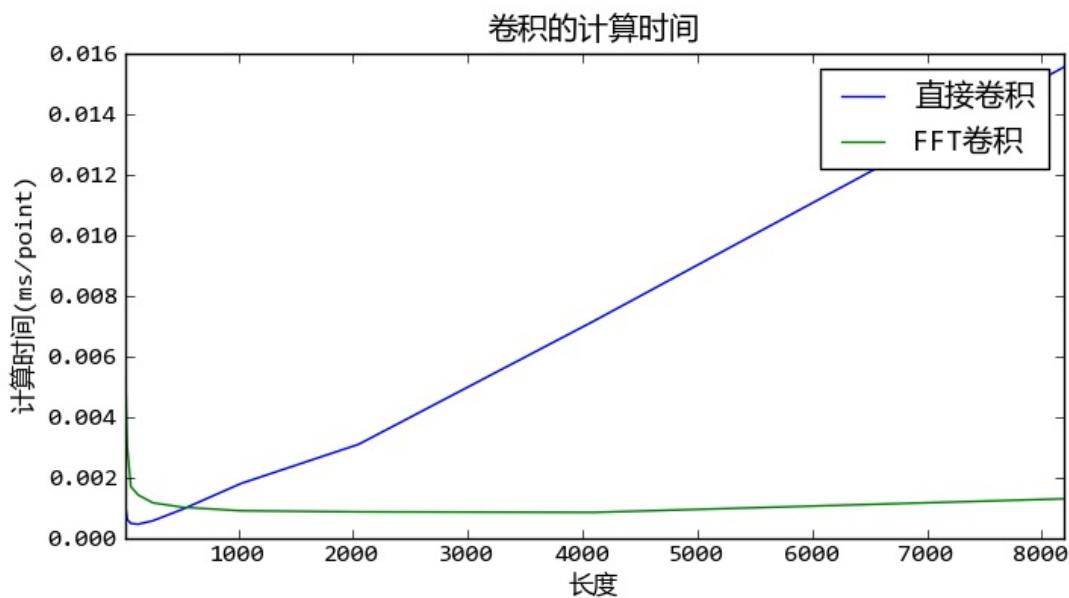
a = pl.axes([.4, .2, .4, .4])
a.plot(freqs, xfp, label=u"")
a.plot(freqs, yfp, label=u"hann")
a.set_xlim(100, 400)
a.set_ylim(-40, 0)
pl.show()
```


Python >> >>

← (Alt+X) | |

FFT

$O(N^2)$ FFT $O(N \log N)$ Y



```
# -*- coding: utf-8 -*-
import numpy as np
import timeit
def fft_convolve(a,b):
    n = len(a)+len(b)-1
    N = 2**int(np.log2(n))+1
    A = np.fft.fft(a, N)
    B = np.fft.fft(b, N)
    return np.fft.ifft(A*B)[:n]

if __name__ == "__main__":
    from pylab import *
    n_list = []
    t1_list = []
    t2_list = []
    for n in xrange(4, 14):
        N = 2**n
        count = 10000**2 / N**2
        if count > 10000: count = 10000
        setup = """
```

```
import numpy as np
from __main__ import fft_convolve
a = np.random.rand(%s)
b = np.random.rand(%s)
    """ % (N, N)
t1 = timeit.timeit("np.convolve(a,b)", setup, number=count)
t2 = timeit.timeit("fft_convolve(a,b)", setup, number=count)
t1_list.append(t1*1000/count/N)
t2_list.append(t2*1000/count/N)
n_list.append(N)
figure(figsize=(8,4))
plot(n_list, t1_list, label=u""      )
plot(n_list, t2_list, label=u"FFT"   )
legend()
title(u""          )
ylabel(u"(ms/point)"      )
xlabel(u""          )
xlim(min(n_list),max(n_list))
show()
```

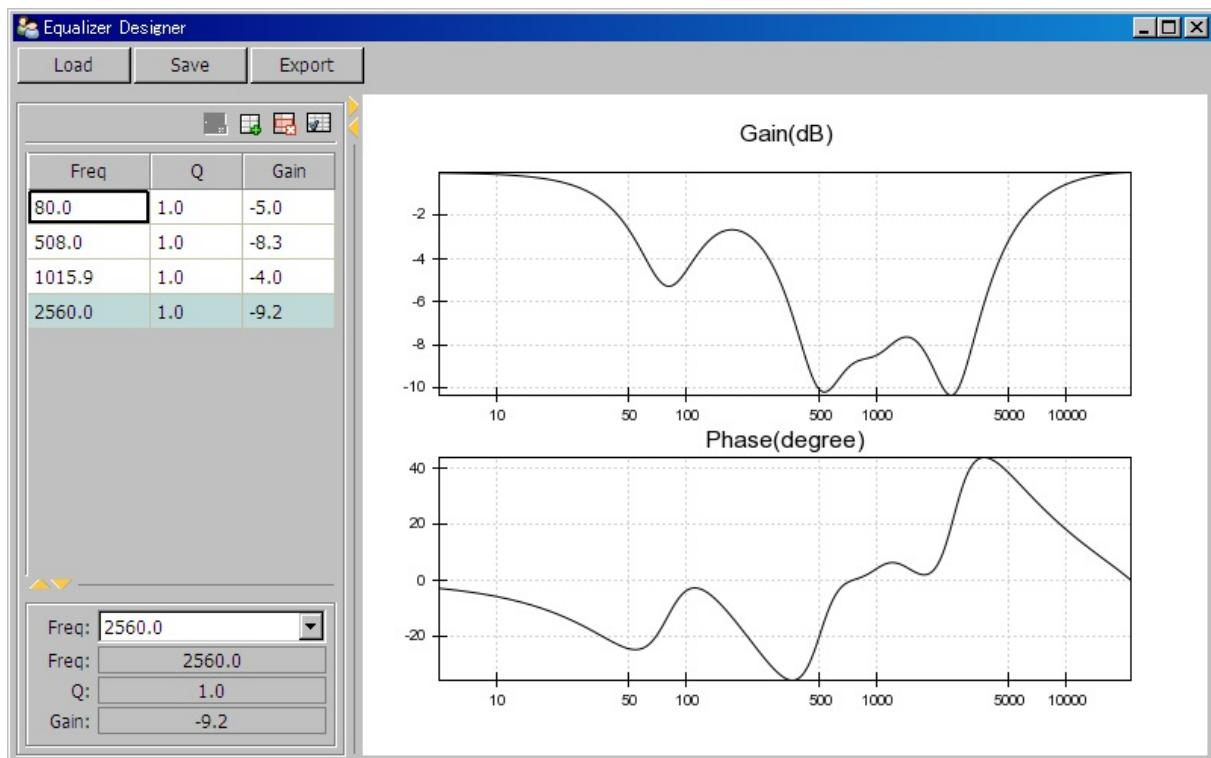
Python » »

← (Alt+X) | |

Python >> >>

← (Alt+X) | |

Traits.UIChacoQ



```
# -*- coding: utf-8 -*-
import math
from enthought.traits.api import Float, HasTraits, List, Array, on_trait_change
from enthought.traits.ui.api import View, TableEditor, Item, GroupEditor, \
    VGroup, HGroup, TabularEditor, TabularEditor
from enthought.traits.ui.table_column import ObjectColumn
from enthought.chaco.api import Plot, AbstractPlotData, ArrayPlotData
from enthought.chaco.tools.api import PanTool, ZoomTool
from enthought.enable.api import Component, ComponentEditor
from enthought.pyface.api import FileDialog, OK
import pickle
import numpy as np

SAMPLING_RATE = 44100.0 #
WORN = 1000 #

#
```

```

W = np.logspace(np.log10(10/SAMPLING_RATE*np.pi), np.log10(np.pi))

#
FREQS = W / 2 / np.pi * SAMPLING_RATE

#
EQ_FREQS = [20.0, 25.2, 31.7, 40.0, 50.4, 63.5, 80.0, 100.8,
             127.0, 160.0, 201.6, 254.0, 320.0, 403.2, 508.0, 640.0,
             806.3, 1015.9, 1280.0, 1612.7, 2031.9, 2560.0, 3225.4,
             4063.7, 5120.0, 6450.8, 8127.5, 10240.0, 12901.6,
             16255.0, 20480.0,]

def scrubber(inc):
    '''ScrubberEditor'''
    return ScrubberEditor(
        hover_color = 0xFFFFFFFF,
        active_color = 0xA0CD9E,
        border_color = 0x808080,
        increment = inc
    )

def myfreqz(b, a, w):
    '''W'''
    zm1 = np.exp(-1j*w)
    h = np.polyval(b[:-1], zm1) / np.polyval(a[:-1], zm1)
    return h

def design_equalizer(freq, Q, gain, Fs):
    '''
    A = 10**(gain/40.0)
    w0 = 2*math.pi*freq/Fs
    alpha = math.sin(w0) / 2 / Q

    b0 = 1 + alpha * A
    b1 = -2*math.cos(w0)
    b2 = 1 - alpha * A
    a0 = 1 + alpha / A
    a1 = -2*math.cos(w0)
    a2 = 1 - alpha / A

    return [b0/a0, b1/a0, b2/a0], [1.0, a1/a0, a2/a0]
    '''

    class Equalizer(HasTraits):
        freq = Range(10.0, SAMPLING_RATE/2, 1000)
        Q = Range(0.1, 10.0, 1.0)
        gain = Range(-24.0, 24.0, 0)


```

```

a = List(Float, [1.0,0.0,0.0])
b = List(Float, [1.0,0.0,0.0])

h = Array(dtype=np.complex, transient = True)

def __init__(self):
    super(Equalizer, self).__init__()
    self.design_parameter()

@on_trait_change("freq,Q,gain")
def design_parameter(self):
    .....
    try:
        self.b, self.a = design_equalizer(self.freq, self.Q,
    except:
        self.b, self.a = [1.0,0.0,0.0], [1.0,0.0,0.0]
    self.h = myfreqz(self.b, self.a, w)

def export_parameters(self, f):
    '''
    tmp = self.b[0], self.b[1], self.b[2], self.a[1], self.a[2]
    f.write("%s,%s,%s,%s,%s", // %s,%s,%s\n" % tmp)

class Equalizers(HasTraits):
    eqs = List(Equalizer, [Equalizer()])
    h = Array(dtype=np.complex, transient = True)

    # Equalizer eqs
    table_editor = TableEditor(
        columns = [
            ObjectColumn(name="freq", width=0.4, style="readonly"),
            ObjectColumn(name="Q", width=0.3, style="readonly"),
            ObjectColumn(name="gain", width=0.3, style="readonly")
        ],
        deletable = True,
        sortable = True,
        auto_size = False,
        show_toolbar = True,
        edit_on_first_click = False,
        orientation = 'vertical',
        edit_view = View(
            Group(
                Item("freq", editor=EnumEditor(values=EQ_FREQS)),
                Item("freq", editor=scrubber(1.0)),
                Item("Q", editor=scrubber(0.01)),
                Item("gain", editor=scrubber(0.1)),
                show_border=True,

```

```

        ),
        resizable = True
    ),
    row_factory = Equalizer
)

view = View(
    Item("eqs", show_label=False, editor=table_editor),
    width     = 0.25,
    height    = 0.5,
    resizable = True
)

@on_trait_change("eqs.h")
def recalculate_h(self):
    .....
    try:
        tmp = np.array([eq.h for eq in self.eqs if eq.h != None])
        self.h = np.prod(tmp, axis=0)
    except:
        pass

def export(self, path):
    '''C'''
    f = file(path, "w")
    f.write("double EQ_PARS[][][5] = {\n")
    f.write("//b0,b1,b2,a0,a1 // frequency, Q, gain\n")
    for eq in self.eqs:
        eq.export_parameters(f)
    f.write("};\n")
    f.close()

class EqualizerDesigner(HasTraits):
    .....

    equalizers = Instance(Equalizers)

    #
    plot_data = Instance(AbstractPlotData)

    #
    container = Instance(Component)

    plot_gain = Instance(Component)
    plot_phase = Instance(Component)
    save_button = Button("Save")
    load_button = Button("Load")

```

```

export_button = Button("Export")

view = View(
    VGroup(
        HGroup(
            Item("load_button"),
            Item("save_button"),
            Item("export_button"),
            show_labels = False
        ),
        HSplit(
            VGroup(
                Item("equalizers", style="custom", show_label=False,
                     show_border=True,
                ),
                Item("container", editor=ComponentEditor(size=(800, 400)))
            ),
            resizable = True,
            width = 800,
            height = 500,
            title = u"Equalizer Designer"
        )
    )
)

def _create_plot(self, data, name, type="line"):
    p = Plot(self.plot_data)
    p.plot(data, name=name, title=name, type=type)
    p.tools.append(PanTool(p))
    zoom = ZoomTool(component=p, tool_mode="box", always_on=False)
    p.overlays.append(zoom)
    p.title = name
    p.index_scale = "log"
    return p

def __init__(self):
    super(EqualizerDesigner, self).__init__()
    self.plot_data = ArrayPlotData(f=FREQS, gain=[], phase[])
    self.plot_gain = self._create_plot(("f", "gain"), "Gain")
    self.plot_phase = self._create_plot(("f", "phase"), "Phase")
    self.container = VPlotContainer()
    self.container.add( self.plot_phase )
    self.container.add( self.plot_gain )
    self.plot_gain.padding_bottom = 20
    self.plot_phase.padding_top = 20

def _equalizers_default(self):
    return Equalizers()

```

```

@on_trait_change("equalizers.h")
def redraw(self):
    gain = 20*np.log10(np.abs(self.equalizers.h))
    phase = np.angle(self.equalizers.h, deg=1)
    self.plot_data.set_data("gain", gain)
    self.plot_data.set_data("phase", phase)

def _save_button_fired(self):
    dialog = FileDialog(action="save as", wildcard='EQ files')
    result = dialog.open()
    if result == OK:
        f = file(dialog.path, "wb")
        pickle.dump( self.equalizers , f)
        f.close()

def _load_button_fired(self):
    dialog = FileDialog(action="open", wildcard='EQ files (*.')
    result = dialog.open()
    if result == OK:
        f = file(dialog.path, "rb")
        self.equalizers = pickle.load(f)
        f.close()

def _export_button_fired(self):
    dialog = FileDialog(action="save as", wildcard='c files (')
    result = dialog.open()
    if result == OK:
        self.equalizers.export(dialog.path)

win = EqualizerDesigner()
win.configure_traits()

```

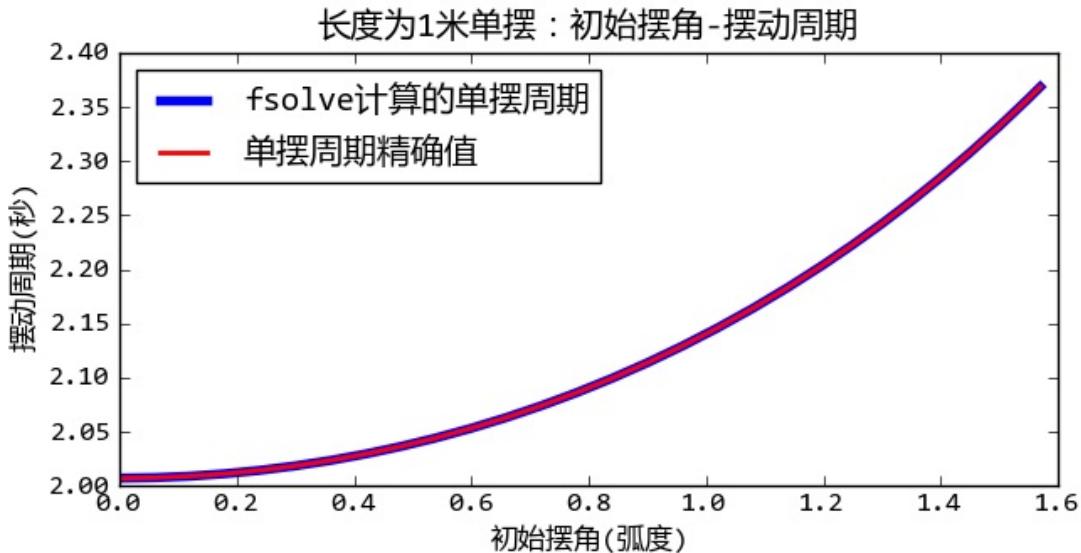
Python » »

←(Alt+X) | |

Python >> >>

← (Alt+X) | |

odeintfsolve



```
# -*- coding: utf-8 -*-
from math import sin, sqrt
import numpy as np
from scipy.integrate import odeint
from scipy.optimize import fsolve
import pylab as pl
from scipy.special import ellipk

g = 9.8

def pendulum_equations(w, t, l):
    th, v = w
    dth = v
    dv = - g/l * sin(th)
    return dth, dv

def pendulum_th(t, l, th0):
    track = odeint(pendulum_equations, (th0, 0), [0, t], args=(l,
    return track[-1, 0]

def pendulum_period(l, th0):
```

```
t0 = 2*np.pi*sqrt( l/g ) / 4
t = fsolve( pendulum_th, t0, args = (l, th0) )
return t**4

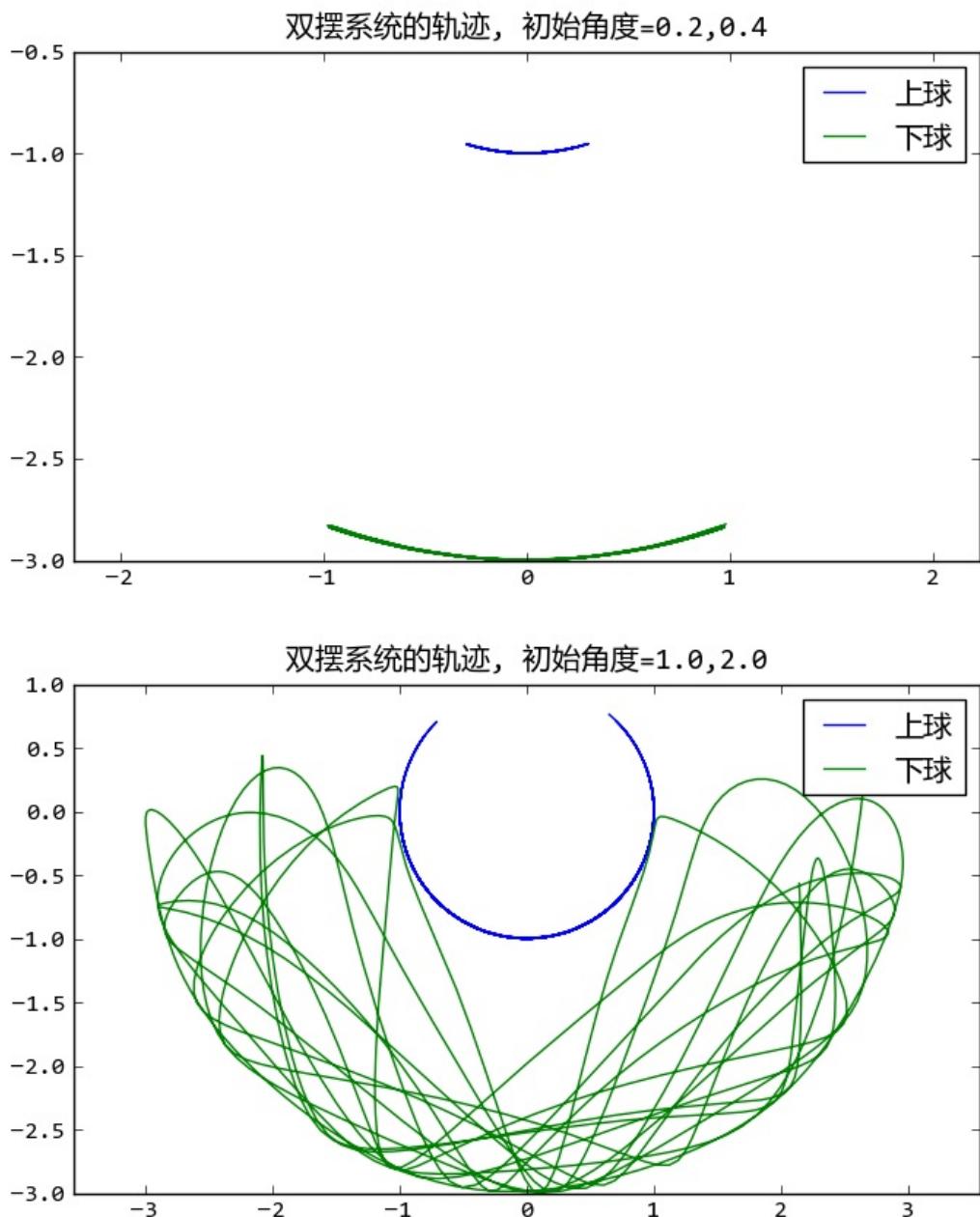
ths = np.arange(0, np.pi/2.0, 0.01)
periods = [pendulum_period(1, th) for th in ths]
periods2 = 4*sqrt(1.0/g)*ellipk(np.sin(ths/2)**2) #
pl.plot(ths, periods, label = u"fsolve" , linewidth=4.0)
pl.plot(ths, periods2, "r", label = u"" , linewidth=2.0)
pl.legend(loc='upper left')
pl.title(u"1-")
pl.xlabel(u"()")
pl.ylabel(u"()")
pl.show()
```

Python » »

← (Alt+X) | |

Python >> >>

← (Alt+X) | |



odeint

: double_pendulum_odeint.py

```
# -*- coding: utf-8 -*-

from math import sin,cos
import numpy as np
from scipy.integrate import odeint

g = 9.8

class DoublePendulum(object):
    def __init__(self, m1, m2, l1, l2):
        self.m1, self.m2, self.l1, self.l2 = m1, m2, l1, l2
        self.init_status = np.array([0.0,0.0,0.0,0.0])

    def equations(self, w, t):
        """
        """
        m1, m2, l1, l2 = self.m1, self.m2, self.l1, self.l2
        th1, th2, v1, v2 = w
        dth1 = v1
        dth2 = v2

        #eq of th1
        a = l1*l1*(m1+m2) # dv1 parameter
        b = l1*m2*l2*cos(th1-th2) # dv2 paramter
        c = l1*(m2*l2*sin(th1-th2)*dth2*dth2 + (m1+m2)*g*sin(th1))

        #eq of th2
        d = m2*l2*l1*cos(th1-th2) # dv1 parameter
        e = m2*l2*l2 # dv2 parameter
        f = m2*l2*(-l1*sin(th1-th2)*dth1*dth1 + g*sin(th2))

        dv1, dv2 = np.linalg.solve([[a,b],[d,e]], [-c,-f])

        return np.array([dth1, dth2, dv1, dv2])

def double_pendulum_odeint(pendulum, ts, te, tstep):
    """
    X-Y
    """
```

```
t = np.arange(ts, te, tstep)
track = odeint(pendulum.equations, pendulum.init_status, t)
th1_array, th2_array = track[:,0], track[:, 1]
l1, l2 = pendulum.l1, pendulum.l2
x1 = l1*np.sin(th1_array)
y1 = -l1*np.cos(th1_array)
x2 = x1 + l2*np.sin(th2_array)
y2 = y1 - l2*np.cos(th2_array)
pendulum.init_status = track[-1,:].copy() #pendulum
return [x1, y1, x2, y2]

if __name__ == "__main__":
    import matplotlib.pyplot as pl
    pendulum = DoublePendulum(1.0, 2.0, 1.0, 2.0)
    th1, th2 = 1.0, 2.0
    pendulum.init_status[:2] = th1, th2
    x1, y1, x2, y2 = double_pendulum_odeint(pendulum, 0, 30, 0.02
    pl.plot(x1,y1, label = u"" )
    pl.plot(x2,y2, label = u"" )
    pl.title(u", = %s,%s" % (th1, th2))
    pl.legend()
    pl.axis("equal")
    pl.show()
```

: double_pendulum_animation.py

```
# -*- coding: utf-8 -*-
import matplotlib
matplotlib.use('WXAgg') # do this before importing pylab
import matplotlib.pyplot as pl
from double_pendulum_odeint import double_pendulum_odeint, DoublePendulum

fig = pl.figure(figsize=(4,4))
line1, = pl.plot([0,0], [0,0], "-o")
line2, = pl.plot([0,0], [0,0], "-o")
pl.axis("equal")
pl.xlim(-4,4)
pl.ylim(-4,2)

pendulum = DoublePendulum(1.0, 2.0, 1.0, 2.0)
pendulum.init_status[:] = 1.0, 2.0, 0, 0

x1, y1, x2, y2 = [],[],[],[]
idx = 0

def update_line(event):
    global x1, x2, y1, y2, idx
    if idx == len(x1):
        x1, y1, x2, y2 = double_pendulum_odeint(pendulum, 0, 1, 0)
        idx = 0
    line1.set_xdata([0, x1[idx]])
    line1.set_ydata([0, y1[idx]])
    line2.set_xdata([x1[idx], x2[idx]])
    line2.set_ydata([y1[idx], y2[idx]])
    fig.canvas.draw()
    idx += 1

import wx
id = wx.NewId()
actor = fig.canvas.manager.frame
timer = wx.Timer(actor, id=id)
timer.Start(1)
wx.EVT_TIMER(actor, id, update_line)
pl.show()
```

Python » »

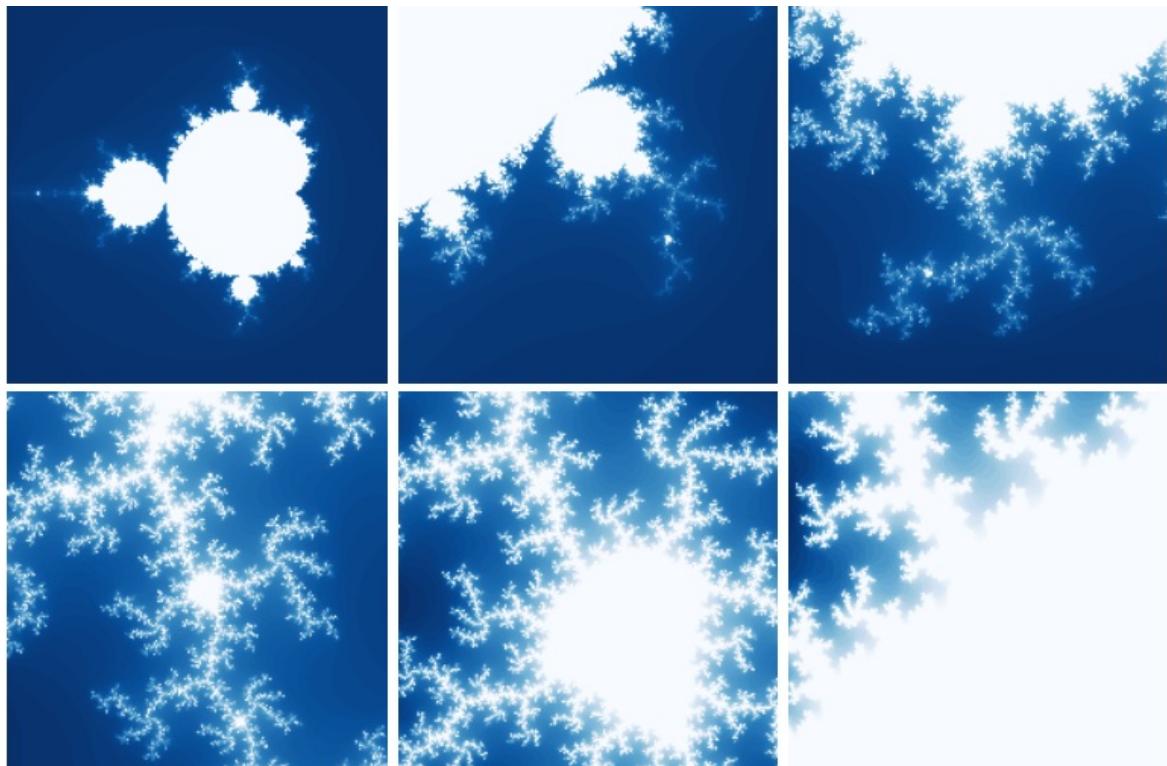
← (Alt+X) | |

Python >> >>

← (Alt+X) | |

Mandelbrot

Mandelbrot



Python

```
# -*- coding: utf-8 -*-

import numpy as np
import pylab as pl
import time
from matplotlib import cm

def iter_point(c):
    z = c
    for i in xrange(1, 100): # 100
        if abs(z)>2: break # 2
        z = z*z+c
    return i #

def draw_mandelbrot(cx, cy, d):
    """
    (cx, cy)dMandelbrot
    """
    x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d
    y, x = np.ogrid[y0:y1:200j, x0:x1:200j]
    c = x + y*1j
    start = time.clock()
    mandelbrot = np.frompyfunc(iter_point,1,1)(c).astype(np.float)
    print "time=",time.clock() - start
    pl.imshow(mandelbrot, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
    pl.gca().set_axis_off()

x,y = 0.27322626, 0.595153338

pl.subplot(231)
draw_mandelbrot(-0.5,0,1.5)
for i in range(2,7):
    pl.subplot(230+i)
    draw_mandelbrot(x, y, 0.2***(i-1))
pl.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0)
pl.show()
```

Weave

```
# -*- coding: utf-8 -*-

import numpy as np
import pylab as pl
import time
import scipy.weave as weave
from matplotlib import cm

def weave_iter_point(c):
    code = """
    std::complex<double> z;
    int i;
    z = c;
    for(i=1;i<100;i++)
    {
        if(std::abs(z) > 2) break;
        z = z*z+c;
    }
    return_val=i;
    """

    f = weave.inline(code, ["c"], compiler="gcc")
    return f

def draw_mandelbrot(cx, cy, d, N=200):
    """
    (cx, cy)dMandelbrot
    """
    x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d
    y, x = np.ogrid[y0:y1:N*1j, x0:x1:N*1j]
    c = x + y*1j
    start = time.clock()
    mandelbrot = np.frompyfunc(weave_iter_point, 1, 1)(c).astype(np
    print "time=", time.clock() - start
    pl.imshow(mandelbrot, cmap=cm.Blues_r, extent=[x0, x1, y0, y1])
    pl.gca().set_axis_off()

x, y = 0.27322626, 0.595153338

pl.subplot(231)
draw_mandelbrot(-0.5, 0, 1.5)
for i in range(2, 7):
```

```
pl.subplot(230+i)
draw_mandelbrot(x, y, 0.2** (i-1))
pl.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0.02)

pl.show()
```

NumPy

```
# -*- coding: utf-8 -*-

import numpy as np
import pylab as pl
import time
from matplotlib import cm

def draw_mandelbrot(cx, cy, d, N=200):
    """
    (cx, cy)dMandelbrot
    """
    global mandelbrot

    x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d
    y, x = np.ogrid[y0:y1:N*_1j, x0:x1:N*_1j]
    c = x + y*_1j

    # X, Y
    ix, iy = np.mgrid[0:N, 0:N]

    # mandelbrot
    mandelbrot = np.ones(c.shape, dtype=np.int)*100

    #
    ix.shape = -1
    iy.shape = -1
    c.shape = -1
    z = c.copy() # c1

    start = time.clock()

    for i in xrange(1,100):
        #
        z *= z
        z += c
        #
        tmp = np.abs(z) > 2.0
        # mandelbrot
        mandelbrot[ix[tmp], iy[tmp]] = i

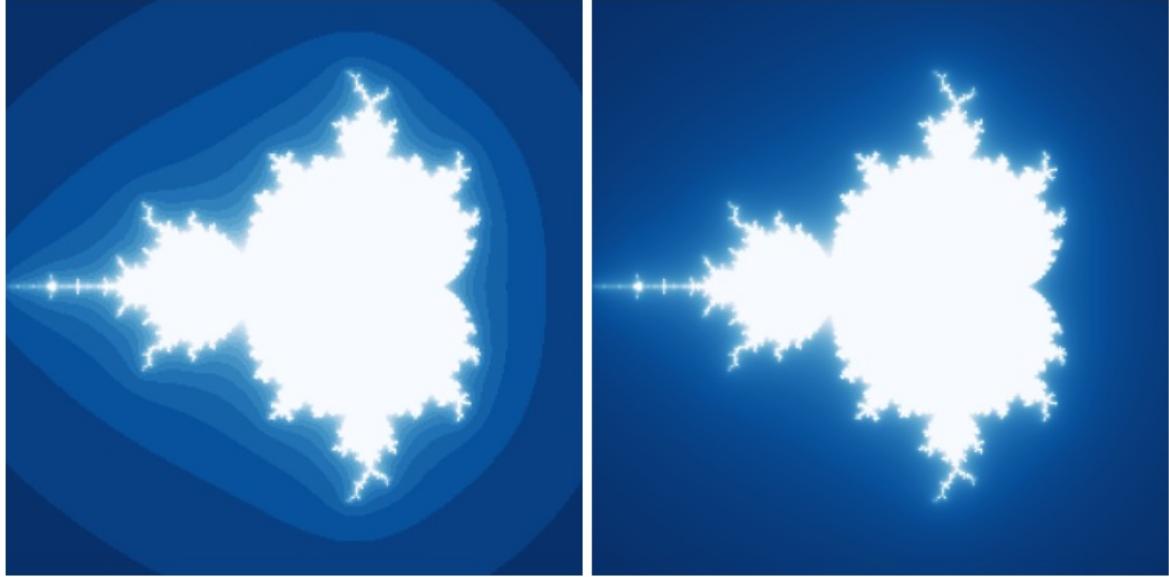
        #
        np.logical_not(tmp, tmp)
```

```
# ix, iy, c, z
ix,iy,c,z = ix[tmp], iy[tmp], c[tmp],z[tmp]
if len(z) == 0: break

print "time=",time.clock() - start
pl.imshow(mandelbrot, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
pl.gca().set_axis_off()

x,y = 0.27322626, 0.595153338

pl.subplot(231)
draw_mandelbrot(-0.5,0,1.5)
for i in range(2,7):
    pl.subplot(230+i)
    draw_mandelbrot(x, y, 0.2** (i-1))
pl.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0)
pl.show()
```



```
# -*- coding: utf-8 -*-
import numpy as np
import pylab as pl
from math import log
from matplotlib import cm

escape_radius = 10
iter_num = 20

def smooth_iter_point(c):
    z = c
    for i in xrange(1, iter_num):
        if abs(z)>escape_radius: break
        z = z*z+c
    absz = abs(z)
    if absz > 2.0:
        mu = i - log(log(abs(z),2),2)
    else:
        mu = i
    return mu #

def iter_point(c):
    z = c
    for i in xrange(1, iter_num):
```

```
    if abs(z)>escape_radius: break
    z = z*z+c
return i

def draw_mandelbrot(cx, cy, d, N=200):
    global mandelbrot
    """
    (cx, cy)dMandelbrot
    """
    x0, x1, y0, y1 = cx-d, cx+d, cy-d, cy+d
    y, x = np.ogrid[y0:y1:N*1j, x0:x1:N*1j]
    c = x + y*1j
    mand = np.frompyfunc(iter_point,1,1)(c).astype(np.float)
    smooth_mand = np.frompyfunc(smooth_iter_point,1,1)(c).astype()
    pl.subplot(121)
    pl.gca().set_axis_off()
    pl.imshow(mand, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
    pl.subplot(122)
    pl.imshow(smooth_mand, cmap=cm.Blues_r, extent=[x0,x1,y0,y1])
    pl.gca().set_axis_off()

draw_mandelbrot(-0.5,0,1.5,300)
pl.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0)
pl.show()
```

Python » »

← (Alt+X) | |

Python >> >>

← (Alt+X) | |

(IFS)



```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as pl
import time

#
eq1 = np.array([[0,0,0],[0,0.16,0]])
p1 = 0.01
```

```

eq2 = np.array([[0.2, -0.26, 0], [0.23, 0.22, 1.6]])
p2 = 0.07

eq3 = np.array([[-0.15, 0.28, 0], [0.26, 0.24, 0.44]])
p3 = 0.07

eq4 = np.array([[0.85, 0.04, 0], [-0.04, 0.85, 1.6]])
p4 = 0.85

def ifs(p, eq, init, n):
    """
    p:
    eq:
    init:
    n:

    X Y
    """

    #
    pos = np.ones(3, dtype=np.float)
    pos[:2] = init

    #
    p = np.add.accumulate(p)
    rands = np.random.rand(n)
    select = np.ones(n, dtype=np.int)*(n-1)
    for i, x in enumerate(p[::-1]):
        select[rands<x] = len(p)-i-1

    #
    result = np.zeros((n,2), dtype=np.float)
    c = np.zeros(n, dtype=np.float)

    for i in xrange(n):
        eqidx = select[i] #
        tmp = np.dot(eq[eqidx], pos) #
        pos[:2] = tmp #

        #
        result[i] = tmp
        c[i] = eqidx

    return result[:,0], result[:, 1], c

```

```
start = time.clock()
x, y, c = ifs([p1,p2,p3,p4],[eq1,eq2,eq3,eq4], [0,0], 100000)
print time.clock() - start
pl.figure(figsize=(6,6))
pl.subplot(121)
pl.scatter(x, y, s=1, c="g", marker="s", linewidths=0)
pl.axis("equal")
pl.axis("off")
pl.subplot(122)
pl.scatter(x, y, s=1,c = c, marker="s", linewidths=0)
pl.axis("equal")
pl.axis("off")
pl.subplots_adjust(left=0,right=1,bottom=0,top=1,wspace=0,hspace=0)
pl.gcf().patch.set_facecolor("white")
pl.show()
```

```
# -*- coding: utf-8 -*-
from enthought.traits.ui.api import *
from enthought.traits.ui.menu import OKCancelButtons
from enthought.traits.api import *
from enthought.traits.ui.wx.editor import Editor

import matplotlib
# matplotlibWXAggwxtraitsUI
matplotlib.use("WXAgg")
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigureCanvas
from matplotlib.figure import Figure

import numpy as np
import thread
import time
import wx
import pickle

ITER_COUNT = 4000 # ifs
ITER_TIMES = 10   # ifs
```

```

def triangle_area(triangle):
    """
    """
    A = triangle[0]
    B = triangle[1]
    C = triangle[2]
    AB = A-B
    AC = A-C
    return np.abs(np.cross(AB,AC))/2.0

def solve_eq(triangle1, triangle2):
    """
    triangle1triangle2
        triangle1,2
        x0,y0
        x1,y1
        x2,y2
    """
    x0,y0 = triangle1[0]
    x1,y1 = triangle1[1]
    x2,y2 = triangle1[2]

    a = np.zeros((6,6), dtype=np.float)
    b = triangle2.reshape(-1)
    a[0, 0:3] = x0,y0,1
    a[1, 3:6] = x0,y0,1
    a[2, 0:3] = x1,y1,1
    a[3, 3:6] = x1,y1,1
    a[4, 0:3] = x2,y2,1
    a[5, 3:6] = x2,y2,1

    c = np.linalg.solve(a, b)
    c.shape = (2,3)
    return c

def ifs(p, eq, init, n):
    """
    p:
    eq:
    init:
    n:
    X Y
    """

```

```

#
pos = np.ones(3, dtype=np.float)
pos[:2] = init

#
p = np.add.accumulate(p)
rands = np.random.rand(n)
select = np.ones(n, dtype=np.int)*(n-1)
for i, x in enumerate(p[::-1]):
    select[rands<x] = len(p)-i-1

#
result = np.zeros((n,2), dtype=np.float)
c = np.zeros(n, dtype=np.float)

for i in xrange(n):
    eqidx = select[i] #
    tmp = np.dot(eq[eqidx], pos) #
    pos[:2] = tmp #

    #
    result[i] = tmp
    c[i] = eqidx

return result[:,0], result[:, 1], c

class _MPLFigureEditor(Editor):
    """
    matplotlib figuretraits
    """
    scrollable = True

    def __init__(self, parent):
        self.control = self._create_canvas(parent)

    def update_editor(self):
        pass

    def _create_canvas(self, parent):
        panel = wx.Panel(parent, -1, style=wx.CLIP_CHILDREN)
        sizer = wx.BoxSizer(wx.VERTICAL)
        panel.SetSizer(sizer)
        mpl_control = FigureCanvas(panel, -1, self.value)
        sizer.Add(mpl_control, 1, wx.LEFT | wx.TOP | wx.GROW)
        self.value.canvas.SetMinSize((10,10))
        return panel

```

```

class MPLFigureEditor(BasicEditorFactory):
    """
    traits.uiEditorFactory
    """
    klass = _MPLFigureEditor

class IFSTriangles(HasTraits):
    """

    """
    version = Int(0) #

    def __init__(self, ax):
        super(IFSTriangles, self).__init__()
        self.colors = ["r", "g", "b", "c", "m", "y", "k"]
        self.points = np.array([(0,0),(2,0),(2,4),(0,1),(1,1),(1,3)])
        self.equations = self.get_eqs()
        self.ax = ax
        self.ax.set_xlim(-10,10)
        self.ax.set_ylim(-10,10)
        canvas = ax.figure.canvas
        # canvas
        canvas.mpl_connect('button_press_event', self.button_press)
        canvas.mpl_connect('button_release_event', self.button_release)
        canvas.mpl_connect('motion_notify_event', self.motion_notify)
        self.canvas = canvas
        self._ind = None
        self.background = None
        self.update_lines()

    def refresh(self):
        """

        """
        self.update_lines()
        self.canvas.draw()
        self.version += 1

    def del_triangle(self):
        """

        """
        self.points = self.points[:-3].copy()
        self.refresh()

    def add_triangle(self):
        """

```

```

    """
    self.points = np.vstack((self.points, np.array([(0,0),(1,0),(2,0)])))
    self.refresh()

def set_points(self, points):
    """
    """

    self.points = points.copy()
    self.refresh()

def get_eqs(self):
    """
    """

    eqs = []
    for i in range(1,len(self.points)/3):
        eqs.append( solve_eq( self.points[:3,:], self.points[i*3:i*3+3,:] ) )
    return eqs

def get_areas(self):
    """
    """

    areas = []
    for i in range(1, len(self.points)/3):
        areas.append( triangle_area(self.points[i*3:i*3+3,:]) )
    s = sum(areas)
    return [x/s for x in areas]

def update_lines(self):
    """
    """

    del self.ax.lines[:]
    for i in xrange(0,len(self.points),3):
        color = self.colors[i/3%len(self.colors)]
        x0, x1, x2 = self.points[i:i+3, 0]
        y0, y1, y2 = self.points[i:i+3, 1]
        type = color+"%so"
        if i==0:
            linewidth = 3
        else:
            linewidth = 1
        self.ax.plot([x0,x1],[y0,y1], type % "--", linewidth=linewidth)
        self.ax.plot([x1,x2],[y1,y2], type % "-.", linewidth=linewidth)

```

```

        self.ax.plot([x0,x2],[y0,y2], type % ":" , linewidth=1)

        self.ax.set_ylim(-10,10)
        self.ax.set_xlim(-10,10)

    def button_release_callback(self, event):
        """
        """
        self._ind = None

    def button_press_callback(self, event):
        """
        """

        if event.inaxes!=self.ax: return
        if event.button != 1: return
        self._ind = self.get_ind_under_point(event.xdata, event.ydata)

    def get_ind_under_point(self, mx, my):
        """
        mx, my
        """
        for i, p in enumerate(self.points):
            if abs(mx-p[0]) < 0.5 and abs(my-p[1])< 0.5:
                return i
        return None

    def motion_notify_callback(self, event):
        """
        """

        self.event = event
        if self._ind is None: return
        if event.inaxes != self.ax: return
        if event.button != 1: return
        x,y = event.xdata, event.ydata

        #
        self.points[self._ind,:] = [x, y]

        i = self._ind / 3 * 3
        #
        x0, x1, x2 = self.points[i:i+3, 0]
        y0, y1, y2 = self.points[i:i+3, 1]
        self.ax.lines[i].set_data([x0,x1],[y0,y1])
        self.ax.lines[i+1].set_data([x1,x2],[y1,y2])

```

```

        self.ax.lines[i+2].set_data([x0,x2],[y0,y2])

#
if self.background == None:
    self.ax.clear()
    self.ax.set_axis_off()
    self.canvas.draw()
    self.background = self.canvas.copy_from_bbox(self.ax.bbox)
    self.update_lines()

#
self.canvas.restore_region(self.background) #
#
for line in self.ax.lines:
    self.ax.draw_artist(line)
self.canvas.blit(self.ax.bbox)

self.version += 1

class AskName(HasTraits):
    name = Str("")
    view = View(
        Item("name", label = u"" ),
        kind = "modal",
        buttons = OKCancelButtons
    )

class IFSHandler(Handler):
    """
    """

    def init(self, info):
        info.object.init_gui_component()
        return True

class IFSDesigner(HasTraits):
    figure = Instance(Figure) # Figure
    ifs_triangle = Instance(IFSTriangles)
    add_button = Button(u"")
    del_button = Button(u"")
    save_button = Button(u"IFS")
    unsave_button = Button(u"IFS")
    clear = Bool(True)
    exit = Bool(False)
    ifs_names = List()
    ifs_points = List()
    current_name = Str

```

```

view = View(
    VGroup(
        HGroup(
            Item("add_button"),
            Item("del_button"),
            Item("current_name", editor = EnumEditor(name="ob",
            Item("save_button"),
            Item("unsave_button"),
            show_labels = False
        ),
        Item("figure", editor=MPLFigureEditor(), show_label=False),
        resizable = True,
        height = 350,
        width = 600,
        title = u"",
        handler = IFSHandler()
    )
)

def _current_name_changed(self):
    self.ifs_triangle.set_points( self.ifs_points[ self.ifs_nam

def _add_button_fired(self):
    """
    """
    self.ifs_triangle.add_triangle()

def _del_button_fired(self):
    self.ifs_triangle.del_triangle()

def _unsave_button_fired(self):
    if self.current_name in self.ifs_names:
        index = self.ifs_names.index(self.current_name)
        del self.ifs_names[index]
        del self.ifs_points[index]
        self.save_data()

def _save_button_fired(self):
    """
    """
    ask = AskName(name = self.current_name)
    if ask.configure_traits():
        if ask.name not in self.ifs_names:

```

```

        self.ifs_names.append( ask.name )
        self.ifs_points.append( self.ifs_triangle.points )
    else:
        index = self.ifs_names.index(ask.name)
        self.ifs_names[index] = ask.name
        self.ifs_points[index] = self.ifs_triangle.points
    self.save_data()

def save_data(self):
    with file("IFS.data", "wb") as f:
        pickle.dump(self.ifs_names[:], f) # ifs_nameslistlist
        for data in self.ifs_points:
            np.save(f, data) #

def ifs_calculate(self):
    """
    """

def draw_points(x, y, c):
    if len(self.ax2.collections) < ITER_TIMES:
        try:
            self.ax2.scatter(x, y, s=1, c=c, marker="s",
            self.ax2.set_axis_off()
            self.ax2.axis("equal")
            self.figure.canvas.draw()
        except:
            pass

def clear_points():
    self.ax2.clear()

while 1:
    try:
        if self.exit == True:
            break
        if self.clear == True:
            self.clear = False
            self.initpos = [0, 0]
            # 100
            x, y, c = ifs( self.ifs_triangle.get_areas(),
            self.initpos = [x[-1], y[-1]]
            self.ax2.clear()

            x, y, c = ifs( self.ifs_triangle.get_areas(), self.
            if np.max(np.abs(x)) < 1000000 and np.max(np.abs(
                self.initpos = [x[-1], y[-1]]
                wx.CallAfter( draw_points, x, y, c )
```

```

        time.sleep(0.05)
    except:
        pass

@on_trait_change("ifs_triangle.version")
def on_ifs_version_changed(self):
    """
    """

    self.clear = True

def _figure_default(self):
    """
    figureFigure
    """
    figure = Figure()
    self.ax = figure.add_subplot(121)
    self.ax2 = figure.add_subplot(122)
    self.ax2.set_axis_off()
    self.ax.set_axis_off()
    figure.subplots_adjust(left=0, right=1, bottom=0, top=1, wspace=0)
    figure.patch.set_facecolor("w")
    return figure

def init_gui_component(self):
    self.ifs_triangle = IFSTriangles(self.ax)
    self.figure.canvas.draw()
    thread.start_new_thread( self.ifs_calculate, () )
    try:
        with file("ifs.data","rb") as f:
            self.ifs_names = pickle.load(f)
            self.ifs_points = []
            for i in xrange(len(self.ifs_names)):
                self.ifs_points.append(np.load(f))

        if len(self.ifs_names) > 0:
            self.current_name = self.ifs_names[-1]
    except:
        pass

designer = IFSDesigner()
designer.configure_traits()
designer.exit = True

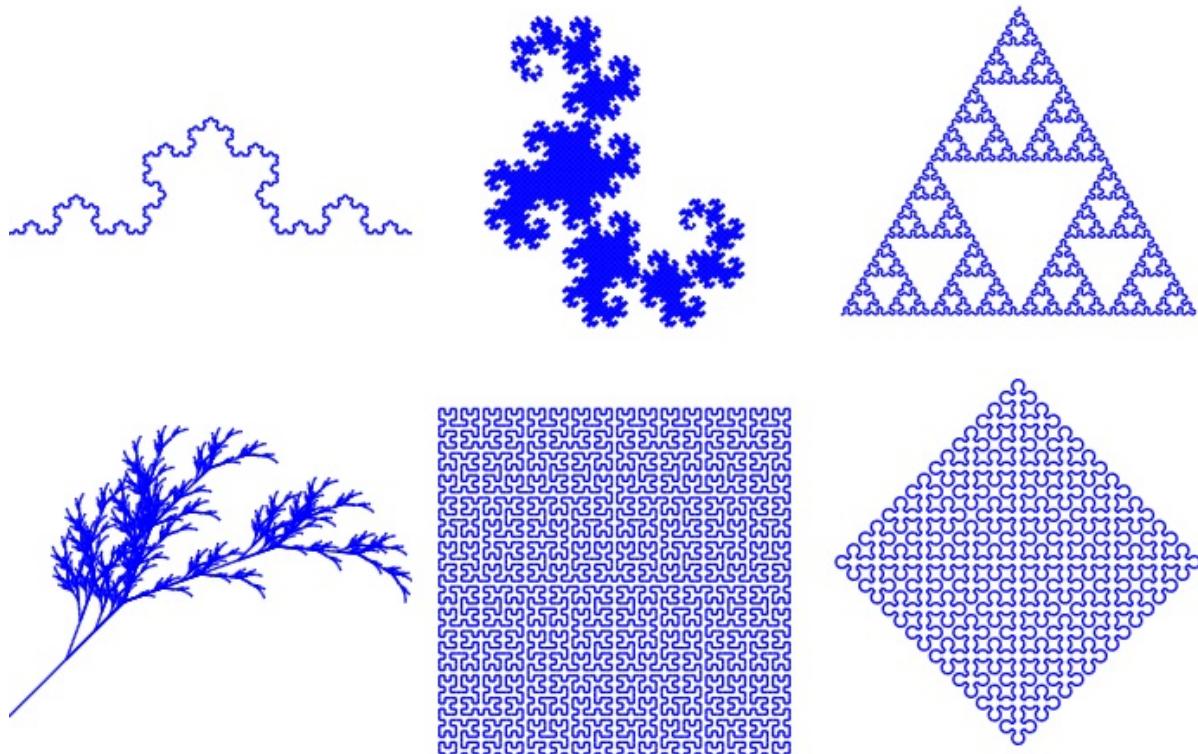
```


Python >> >>

← (Alt+X) |

L-System

L-System



```
# -*- coding: utf-8 -*-
#L-System(Lindenmayer system)
from math import sin, cos, pi
import matplotlib.pyplot as pl
from matplotlib import collections

class L_System(object):
    def __init__(self, rule):
        info = rule['S']
        for i in range(rule['iter']):
            ninfo = []
            for c in info:
                if c in rule:
                    ninfo.append(rule[c])
                else:
                    ninfo.append(c)
            info = ''.join(ninfo)
        self.rule = rule
```

```

self.info = info

def get_lines(self):
    d = self.rule['direct']
    a = self.rule['angle']
    p = (0.0, 0.0)
    l = 1.0
    lines = []
    stack = []
    for c in self.info:
        if c in "Ff":
            r = d * pi / 180
            t = p[0] + l*cos(r), p[1] + l*sin(r)
            lines.append(((p[0], p[1]), (t[0], t[1])))
            p = t
        elif c == "+":
            d += a
        elif c == "-":
            d -= a
        elif c == "[":
            stack.append((p,d))
        elif c == "]":
            p, d = stack[-1]
            del stack[-1]
    return lines

rules = [
    {
        "F": "F+F--F+F", "S": "F",
        "direct": 180,
        "angle": 60,
        "iter": 5,
        "title": "Koch"
    },
    {
        "X": "X+YF+", "Y": "-FX-Y", "S": "FX",
        "direct": 0,
        "angle": 90,
        "iter": 13,
        "title": "Dragon"
    },
    {
        "f": "F-f-F", "F": "f+F+f", "S": "f",
        "direct": 0,
        "angle": 60,
        "iter": 7,
        "title": "Triangle"
    }
]

```

```

},
{
    "X":"F-[ [X]+X]+F[+FX]-X", "F":"FF", "S":"X",
    "direct":-45,
    "angle":25,
    "iter":6,
    "title":"Plant"
},
{
    "S":"X", "X":"-YF+XF+FY-", "Y":"+XF-YFY-FX+",
    "direct":0,
    "angle":90,
    "iter":6,
    "title":"Hilbert"
},
{
    "S":"L--F--L--F", "L": "+R-F-R+", "R": "-L+F+L-",
    "direct":0,
    "angle":45,
    "iter":10,
    "title":"Sierpinski"
},
]

def draw(ax, rule, iter=None):
    if iter!=None:
        rule["iter"] = iter
    lines = L_System(rule).get_lines()
    linecollections = collections.LineCollection(lines)
    ax.add_collection(linecollections, autolim=True)
    ax.axis("equal")
    ax.set_axis_off()
    ax.set_xlim(ax.dataLim.xmin, ax.dataLim xmax)
    ax.invert_yaxis()

fig = pl.figure(figsize=(7,4.5))
fig.patch.set_facecolor("w")

for i in xrange(6):
    ax = fig.add_subplot(231+i)
    draw(ax, rules[i])

fig.subplots_adjust(left=0,right=1,bottom=0,top=1,wspace=0,hspace=0)
pl.show()

```

Python >> >>

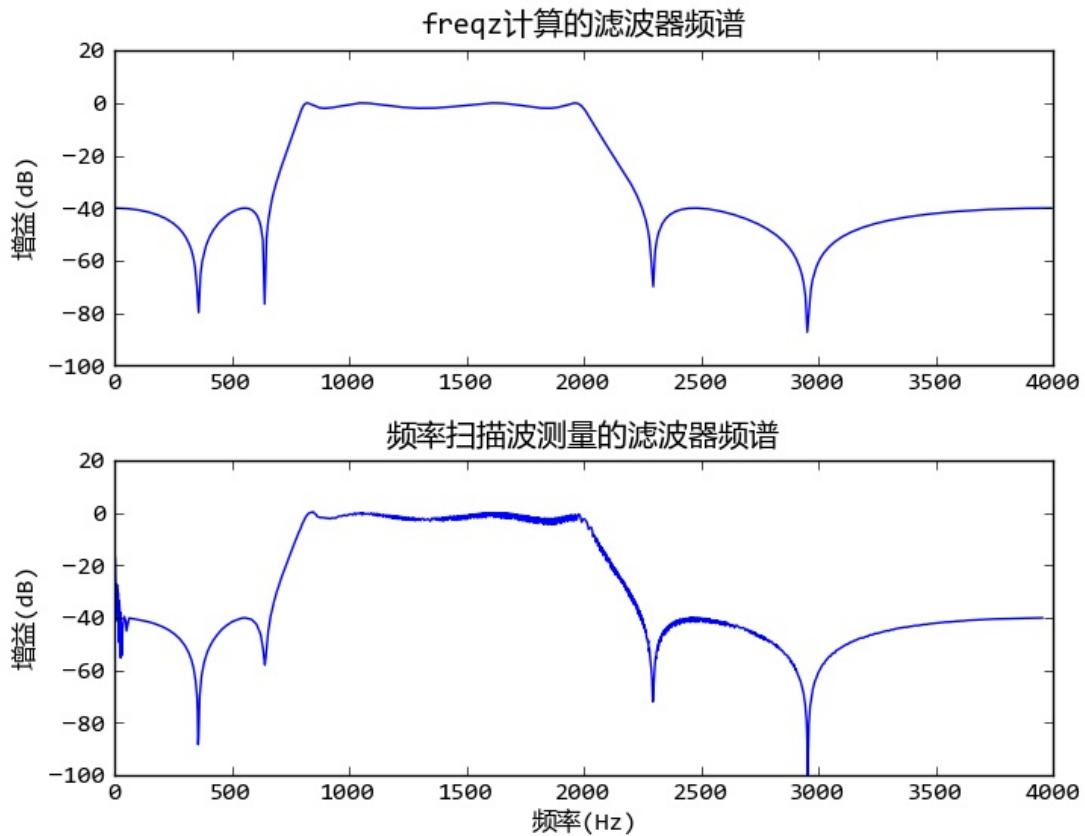
← (Alt+X) |

Python »

← (Alt+X) |

SciPy-

scipy.signal



```
# -*- coding: utf-8 -*-
import scipy.signal as signal
import pylab as pl
import numpy as np

#8kHz
sampling_rate = 8000.0

#
## 0.2*4000 - 0.5*4000
## <0.1*4000, >0.6*4000
## 2dB
```

```

# 40dB
b, a = signal.iirdesign([0.2, 0.5], [0.1, 0.6], 2, 40)

# freq
w, h = signal.freqz(b, a)

#
power = 20*np.log10(np.clip(np.abs(h), 1e-8, 1e100))

#
pl.subplot(211)
pl.plot(w/np.pi*sampling_rate/2, power)
pl.title(u"freqz")
pl.ylim(-100,20)
pl.ylabel(u"(dB)")

# 2sampling_rate Hz
# 0 sampling_rate/2
t = np.arange(0, 2, 1/sampling_rate)
sweep = signal.chirp(t, f0=0, t1 = 2, f1=sampling_rate/2)

#
out = signal.lfilter(b, a, sweep)

#
out = 20*np.log10(np.abs(out))

#
index = np.where(np.logical_and(out[1:-1] > out[:-2], out[1:-1] >

#
pl.subplot(212)
pl.plot(t[index]/2.0*4000, out[index] )
pl.title(u"")
pl.ylim(-100,20)
pl.ylabel(u"(dB") )
pl.xlabel(u"(Hz") )

pl.subplots_adjust(hspace=0.3)
pl.show()

```