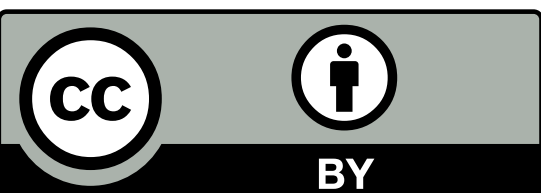


Python

Digital Audio Coding



SEBASTIEN.BOISGERAULT@MINES-PARISTECH.FR

Learn Python 2.7

Tutorials

<http://docs.python.org/tutorial/>

<http://www.diveintopython.net>

<http://tinyurl.com/OCW-python>

Scientific Computing

<http://tinyurl.com/python-Matlab>

http://www.scipy.org/NumPy_for_Matlab_Users

<http://mathesaurus.sourceforge.net/matlab-numpy.html>

First Program

file hello.py

```
message = "Hello World!"  
print message
```

Shell / Console

```
$ python hello.py  
Hello World !
```

Python Interpreter

```
>>> import hello  
Hello World !  
  
>>> hello.message  
'Hello World !'
```

IPython

```
In [1]: %run prg  
Hello World !  
  
In [2]: message  
Out[2]: 'Hello World !'
```

Primitive Types

>>> length = 411852	—————→	int
>>> frequency = 44.1	—————→	float
>>> magic = "RIFF"	—————→	str
>>> stereo = True	—————→	bool
>>> author = None	—————→	(NoneType)

Containers

Lists

```
>>> items = [4, 5, 6]
>>> items.append(7)
>>> items.insert(0, 3)
>>> items
[3, 4, 5, 6, 7]
>>> items[2]
5
>>> items.pop()
7
```

Dicts

```
>>> p = {"heads": 0.5, "tails": 0.5}
>>> p["edge"] = 0.01
>>> p["heads"] = 0.5 - p["edge"]
>>> p
{'tails': 0.5, 'edge': 0.01, 'heads': 0.49}
>>> p.get("stolen by a magpie", 0.0)
0.0
```

Control Structures

```
if x > 0.0:
```

```
    sign = 1
```

```
elif x < 0.0:
```

```
    sign = -1
```

```
else:
```

```
    sign = 0
```

```
while not_ready():
```

```
    wait(10.0)
```

```
for i in [1, 2, 3, 4, 5]:
```

```
    print i
```

SEE ALSO: **pass, continue, break.**

Iteration

```
>>> items = [4, 5, 6]
```

```
>>> len(items)
```

```
3
```

```
>>> for item in items:
```

```
...     print item,
```

```
4 5 6
```

```
>>> p = {"heads": 0.5, "tails": 0.5}
```

```
>>> len(items)
```

```
2
```

```
>>> for key in p:
```

```
...     print key,
```

```
heads tails
```

SEE ALSO: `enumerate`, `zip`

Tests

```
>>> n = 5
```

```
>>> 0 < n <= 7
```

```
True
```

```
>>> one = n == 1
```

```
>>> odd = bool(n % 2)
```

```
>>> odd and not one
```

```
True
```

```
>>> numbers = [0, 1, 2, 3]
```

```
>>> numbers == [0, 1, 2, 3]
```

```
True
```

```
>>> numbers is [0, 1, 2, 3]
```

```
False
```

```
>>> numbers is None or \
```

```
... numbers == []
```

```
False
```


Functions

define

```
def fib(n, start=(0, 1)):
    result = []
    a, b = start
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

call

```
>>> numbers = fib(10)
>>> numbers
[0, 1, 1, 2, 3, 5, 8]
>>> numbers = fib(20, start=(5,8))
>>> numbers
[5, 8, 13]
```

Objects

>>> complex —————> TYPE / CLASS

<type 'complex'>

>>> c = complex(2.0, 1.0) —————> CONSTRUCTOR (CALL)

>>> c —————> INSTANCE

(2+1j)




>>> c.imag —————> ATTRIBUTE


1.0


>>> c.conjugate() —————> METHOD (CALL)

(2-1j)

Objects: Definition

```
class complex(object):  TYPE / CLASS
    def __init__(self, real=0.0, imag=0.0):  CONSTRUCTOR
        self.real = real
        self.imag = imag  ATTRIBUTES

    def conjugate(self):  METHOD
        return complex(self.real, -self.imag)

    def __str__(self):  METHOD
        return "{0} + j{1}".format(self.real, self.imag) (SPECIAL)
```

Files

```
filename = "07MomentofClarity.mp3"
```

```
>>> file = open(filename)    or    >>> url = "http://stereo.lu/grey/" + filename
```

```
>>> import urllib
```

```
>>> file = urllib.urlopen(url)
```

```
>>> raw = file.read(); file.close()
```

```
>>> raw[0:128]
```

```
"ID3\x02\x00\x00\x00\x0fBwTT2\x00\x00\x12\x00Moment  
of ClarityTAL\x00\x00\x0f\x00The Grey AlbumTYE\x00\x00\x05  
\x002004CM1\x00\x00\x11\x00The STFU Man WCRTRK\x00\x00  
\x03\x007\x00TP1\x00\x00\x19\x00Jay-Z + DJ Danger Mouse"
```

```
>>> copy = open("moment-of-clarity.mp3", "w")
```

```
>>> copy.write(raw)
```

```
>>> copy.close()
```



WAVE Files

(RIFF, Linear 16-bit PCM)

```
>>> import wave
```

```
>>> data = wave.read("sound.wav")
```

```
>>> data
```

```
array([ 1., 0.99803566, ..., 0.99803566])
```

```
>>> wave.write(data, "sound-copy.wav")
```

```
>>> wave.write([0, 1, 2], "mono.wav")
```

```
>>> wave.write([[0, 1, 2], [0, 1, 2]], "stereo.wav")
```

```
>>> wave.write([-1.0, 0.0, 1.0], "float.wav")
```

Bit Streams

```
from bitstream import BitStream
```

Create: `stream = BitStream()`

Write: `stream.write(data, type)`

Read: `data = stream.read(type, n)`

stream

```
>>> stream = BitStream()
>>> stream.write(True, bool) 1
>>> stream.write(False, bool) 10
>>> stream.write(-128, int8) 1010000000
>>> stream.write("AB", str) 101000000000100000101000010
>>> stream.read(bool, 2) 1000000000100000101000010
[True, False]
>>> stream.read(int8, 1) 0100000101000010
array([-128], dtype=int8)
>>> stream.read(str, 2)
"AB"
```

NumPy / SciPy

Scientific Computing in Python

N-dimensional **ARRAY** object

+

- Linear Algebra,
- Fourier Analysis,
- Random Numbers,
- Optimization,
- etc.

http://www.scipy.org/more_about_SciPy

Matplotlib

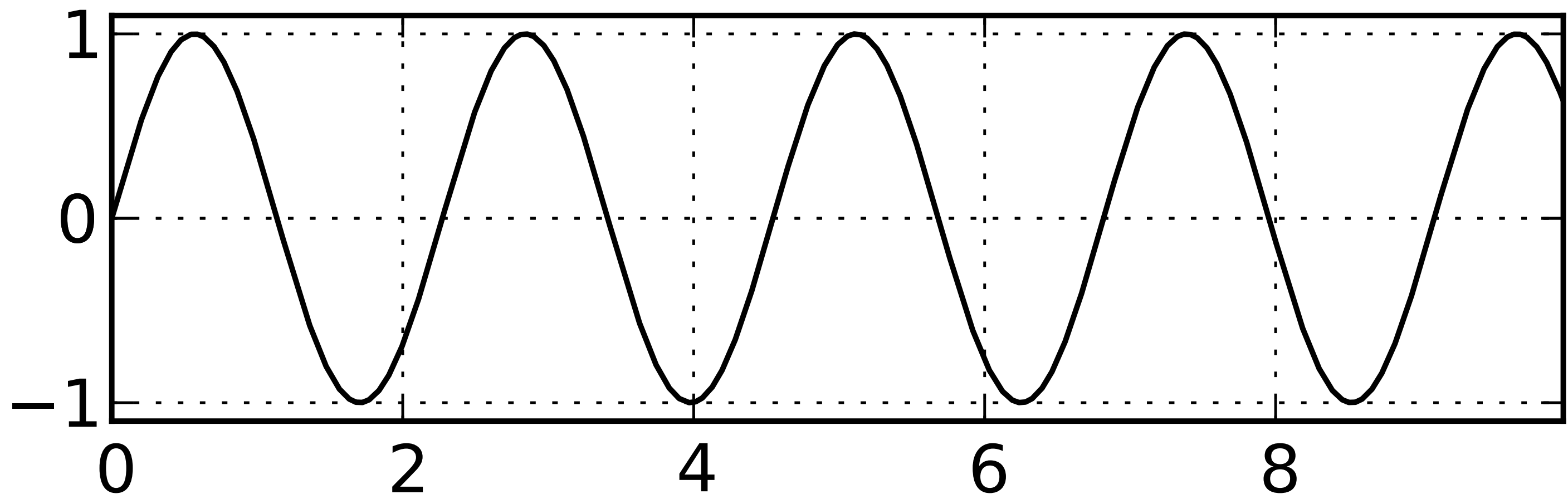


```
from pylab import *
```

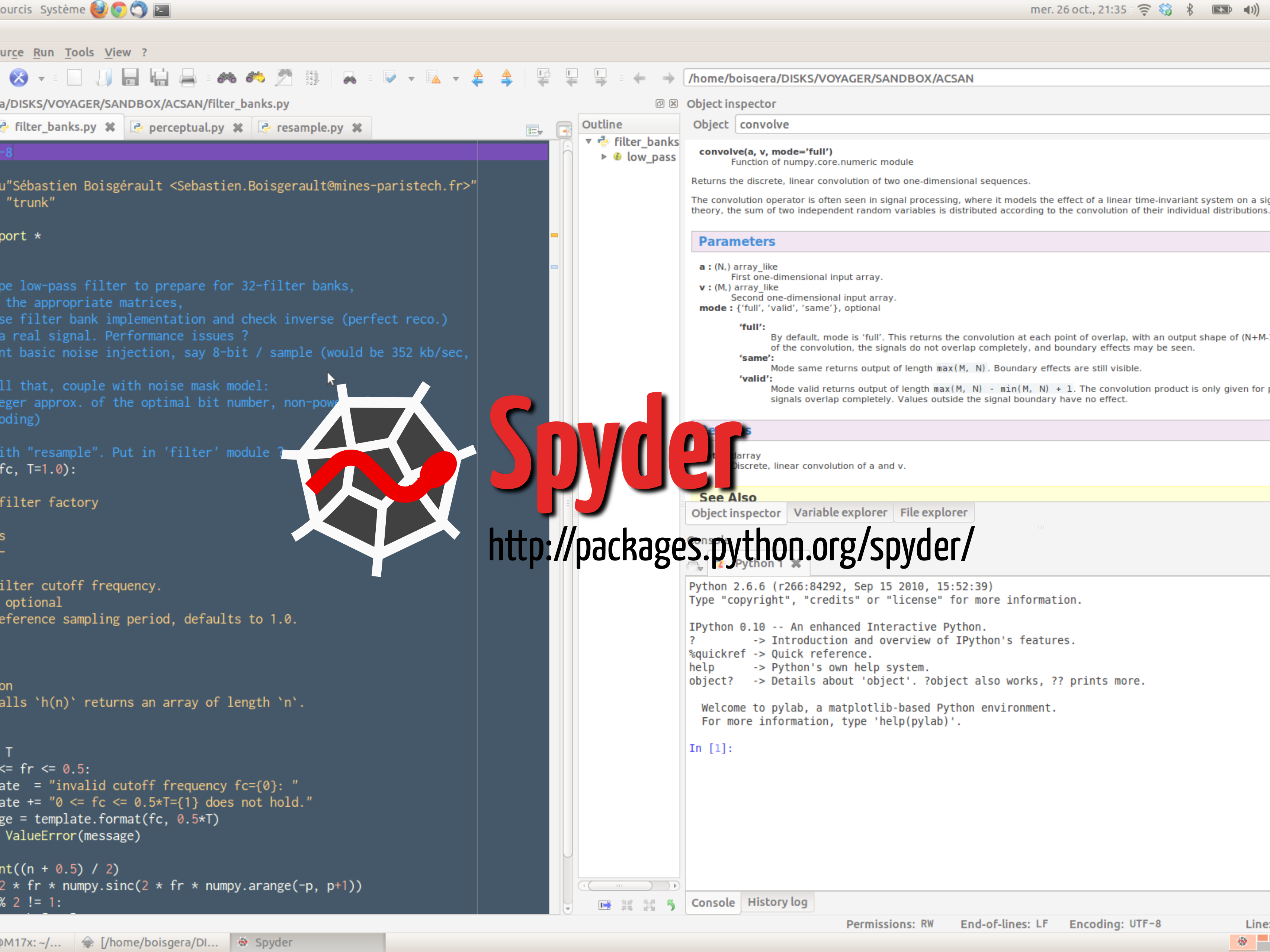
```
f = 440.0; df = 44100.0; dt = 1.0 / df
```

```
t = r_[0.0:0.01: dt]; x = sin(2*pi*f*t)
```

```
plot(1000*t, x)
```



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



array = data + shape + type

```
>>> a0 = array(1.0)
```

```
>>> a1 = array([1, 2, 3])
```

```
>>> a2 = array([[1.0+1.0j], [1.0-1.0j]])
```

```
>>> shape(a0)  
()
```

```
>>> shape(a1)  
(3,)
```

```
>>> shape(a2)  
(2, 1)
```

```
>>> a0.dtype  
dtype('float64')
```

```
>>> a1.dtype  
dtype('int32')
```

```
>>> a2.dtype  
dtype('complex128')
```

Transform Array Shape

reshape

```
>>> reshape(a0, (1,1))  
array([[ 1.]])  
>>> reshape(a1, (3,1))  
array([[1],  
       [2],  
       [3]])  
>>> reshape(a2, (2,))  
array([ 1.+1.j, 1.-1.j])
```

flatten

```
>>> ravel(a0)  
array([ 1.])  
>>> ravel(a1)  
array([1, 2, 3])  
>>> ravel(a2)  
array([ 1.+1.j, 1.-1.j])
```

Transform Array Data Type

create

```
>>> a0 = array(1, dtype=float64)
>>> a1 = array([int8(1), int16(2), int32(3)])
>>> a2 = array([1.0, 1.0]) + 1j * array([1.0, -1.0])
```

convert

```
>>> a0.astype(int8)
array(1, dtype=int8)
>>> a1.astype(float32)
array([ 1.,  2.], dtype=float32)
>>> a2.astype(uint16)
array([[1],
       [1]], dtype=uint16)
```

array: build from blocks

basic arrays

```
>>> zeros((1,2))      >>> arange(0, 6, 2)      >>> identity(2)
array([[ 0.,  0.]])    array([0, 2, 4])      array([[ 1.,  0.],
>>> ones((3,))        >>> linspace(0.0, 1.0, 3)      [ 0.,  1.]])
array([ 1.,  1.,  1.]) array([ 0.,  0.5,  1.])
>>> arange(5)          >>> logspace(-1, 1, 3)  >>> eye(2,k=1)
array([0, 1, 2, 3, 4]) array([ 0.1,  1., 10.])  array([[ 0.,  1.],
                                     [ 0.,  0.]])
```

concatenate

```
>>> r_[zeros(2), ones(2), 2*ones(2)]
array([ 0.,  0.,  1.,  1.,  2.,  2.])
>>> c_[zeros(2), ones(2), 2*ones(2)]
array([[ 0.,  1.,  2.],
       [ 0.,  1.,  2.]])
```

array: indexing + slicing

```
>>> a = zeros((2,2))
>>> a[0, 1] = 1.0
>>> a[1, :] = [1.0, 2.0]
>>> a
array([[ 0.,  1.],
       [ 1.,  2.]])
>>> a[1, 1]
2.0
>>> a[0, :]
array([ 0.,  1.]
```

```
>>> a = arange(10)
>>> a[1:9:2]
array([1, 3, 5, 7])
>>> a[::-1]
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
>>> a[3:-3]
array([3, 4, 5, 6])
>>> a[[0, 1, 3, 6, 9]]
array([0, 1, 3, 6, 9])
>>> a[a>5]
array([6, 7, 8, 9])
```

array: ufuncs + linear algebra

```
>>> sin([0, pi/2])  
array([ 0.000..., 1.000...])  
>>> exp(arange(3))  
array([ 1.      , 2.718..., 7.389...])  
>>> array([0.0, 1.0]) + ones(2)  
array([ 1.,  2.])  
>>> array([0.0, 1.0]) * ones(2)  
array([ 0.,  1.])  
>>> floor([0.0, 0.5, 1.0])  
array([ 0.,  0.,  1.])
```

```
>>> A = array([[1,2], [3,4]])  
>>> x = array([-1, 1])  
>>> dot(A, x)  
array([1, 1])  
>>> dot(x, A)  
array([2, 2])  
>>> dot(x, x)  
2  
>>> dot(A, ones((2,2)))  
array([[ 3.,  3.],  
       [ 7.,  7.]])
```