

Python for people who know matlab

Fedor Baart

July 17, 2011

```
○○○○  
○○○  
○○○  
○○○○○○○○○○
```

```
○○○○○○○○○○  
○○○  
○○○
```

```
○○○○○  
○○○○  
○○○○○
```

Introduction



```
oooo
ooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
ooo
ooo
```

```
ooooo
oooo
ooooo
```

1 Setting up

- Working environment
- Where to get help?
- Some nice features

2 The language

- Data types
- Reflection and namespaces
- Performance

3 Python nice libraries

- Glue
- Plotting
- Gis

4 Python common surprises

○○○○
○○○
○○○
○○○○○○○○○○

○○○○○○○○○○
○○○
○○○
○○○

○○○○○
○○○○
○○○○○
○○○○○○

Outline

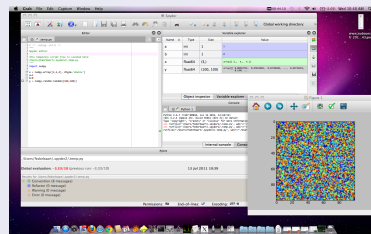
- 1 Setting up
 - Working environment
 - Where to get help?
 - Some nice features
- 2 The language
 - Data types
 - Reflection and namespaces
 - Performance
- 3 Python nice libraries
 - Glue
 - Plotting
 - Gis
- 4 Python common surprises

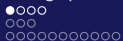


Most popular python IDE (@SO)

15 Spyder

Screenshot



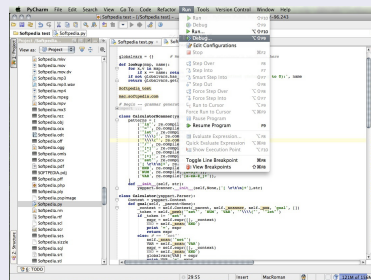


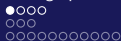
Most popular python IDE (@SO)

4 PyCharm

15 Spyder

Screenshot





Most popular python IDE (@SO)

- 3 emacs
- 4 PyCharm
- 15 Spyder

Screenshot

```

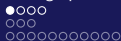
Python 2.7.2 (default, Jun 27 2011; 08:46:42)
[GCC 4.2.3 (Apple Inc. build 5666) (mac32) on darwin
Type "help", "copyright", "credits" or "license()" for more
>>>

```

```

import numpy as np
a = np.array([[1,2,3],[4,5,6]])
b = [1,2,3]
a[0,0] = 7
print(a)
a
print(b)
def test(a):
    pass

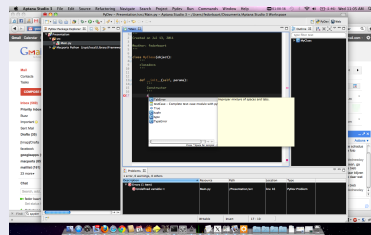
```



Most popular python IDE (@SO)

- 2 PyDev
- 3 emacs
- 4 PyCharm
- 15 Spyder

Screenshot





Most popular python IDE (@SO)

- 1 vim
- 2 PyDev
- 3 emacs
- 4 PyCharm
- 15 Spyder

Screenshot

```

Terminal -- vim -- 132x48 -- 381
class MemberKwargs:
    """
    A simple class for the AllOff-Member projection, an equal-size map
    projection.
    """
    http://en.wikipedia.org/wiki/Member_projection

    # The projection must specify a name. This will be used by the
    # user to select the projection, i.e. '~obj.all()'.
    # projection='member'

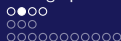
    name = 'member'

    def __init__(self, merge, reorganize):
        self._merge = merge
        self._reorganize = reorganize
        self._cls = None

    def __init_subclass__(self):
        self._merge = Merge.AllOff(self)
        self._reorganize = Reorganize.AllOff(self)

    # Note: merge values of dicts with options -- do this on
    # self._init_merge() -- and! MemberKwargs(cls) works.
    # self._init_reorganize() -- and! MemberKwargs(cls) works.
    self._update_reorganize()

    def __call__(self):
        """
        Override to set up some reasonable defaults.
  
```



Which version?

Python

- Python 2.6, Released October 1st, 2008. Still used by python xy
- Python 2.7, Released July 3rd, 2010. Should be in python xy soon.
- Python 3.x, 3.0 released December 3rd, 2008, start using when all your packages are available.



How to install?

Windows

- Python x,y
- Enthought Python Distribution

Linux

- yum
- apt-get (some extra ppd)
- emerge



How to install?

OSX

- macports (from source)
- homebrew

Extra modules

- pip install module
- pypi.python.org
- python setup.py install



Python

```
help(function) #inline help
pydoc # from command line
pydoc -p 10000 # webserver
```

most python modules have a sphinx doc website:

<http://sphinx.pocoo.org/examples.html>

- 1 docs.python.org
- 2 docs.scipy.org
- 3 matplotlib.sourceforge.net

Matlab

```
help function
doc function
```



Where to get help?

Communities

Python

```

>>> dirb = {}
>>> dirb["a"] = 1
>>> dirb["a"]
1
>>> dirb["a"] = 2
>>> dirb["a"]
2
>>> dirb["a"] = 3
>>> dirb["a"]
3
>>> dirb["a"] = 4
>>> dirb["a"]
4
>>> dirb["a"] = 5
>>> dirb["a"]
5
>>> dirb["a"] = 6
>>> dirb["a"]
6
>>> dirb["a"] = 7
>>> dirb["a"]
7
>>> dirb["a"] = 8
>>> dirb["a"]
8
>>> dirb["a"] = 9
>>> dirb["a"]
9
>>> dirb["a"] = 10
>>> dirb["a"]
10
>>> dirb["a"] = 11
>>> dirb["a"]
11
>>> dirb["a"] = 12
>>> dirb["a"]
12
>>> dirb["a"] = 13
>>> dirb["a"]
13
>>> dirb["a"] = 14
>>> dirb["a"]
14
>>> dirb["a"] = 15
>>> dirb["a"]
15
>>> dirb["a"] = 16
>>> dirb["a"]
16
>>> dirb["a"] = 17
>>> dirb["a"]
17
>>> dirb["a"] = 18
>>> dirb["a"]
18
>>> dirb["a"] = 19
>>> dirb["a"]
19
>>> dirb["a"] = 20
>>> dirb["a"]
20
>>> dirb["a"] = 21
>>> dirb["a"]
21
>>> dirb["a"] = 22
>>> dirb["a"]
22
>>> dirb["a"] = 23
>>> dirb["a"]
23
>>> dirb["a"] = 24
>>> dirb["a"]
24
>>> dirb["a"] = 25
>>> dirb["a"]
25
>>> dirb["a"] = 26
>>> dirb["a"]
26
>>> dirb["a"] = 27
>>> dirb["a"]
27
>>> dirb["a"] = 28
>>> dirb["a"]
28
>>> dirb["a"] = 29
>>> dirb["a"]
29
>>> dirb["a"] = 30
>>> dirb["a"]
30
>>> dirb["a"] = 31
>>> dirb["a"]
31
>>> dirb["a"] = 32
>>> dirb["a"]
32
>>> dirb["a"] = 33
>>> dirb["a"]
33
>>> dirb["a"] = 34
>>> dirb["a"]
34
>>> dirb["a"] = 35
>>> dirb["a"]
35
>>> dirb["a"] = 36
>>> dirb["a"]
36
>>> dirb["a"] = 37
>>> dirb["a"]
37
>>> dirb["a"] = 38
>>> dirb["a"]
38
>>> dirb["a"] = 39
>>> dirb["a"]
39
>>> dirb["a"] = 40
>>> dirb["a"]
40
>>> dirb["a"] = 41
>>> dirb["a"]
41
>>> dirb["a"] = 42
>>> dirb["a"]
42
>>> dirb["a"] = 43
>>> dirb["a"]
43
>>> dirb["a"] = 44
>>> dirb["a"]
44
>>> dirb["a"] = 45
>>> dirb["a"]
45
>>> dirb["a"] = 46
>>> dirb["a"]
46
>>> dirb["a"] = 47
>>> dirb["a"]
47
>>> dirb["a"] = 48
>>> dirb["a"]
48
>>> dirb["a"] = 49
>>> dirb["a"]
49
>>> dirb["a"] = 50
>>> dirb["a"]
50
>>> dirb["a"] = 51
>>> dirb["a"]
51
>>> dirb["a"] = 52
>>> dirb["a"]
52
>>> dirb["a"] = 53
>>> dirb["a"]
53
>>> dirb["a"] = 54
>>> dirb["a"]
54
>>> dirb["a"] = 55
>>> dirb["a"]
55
>>> dirb["a"] = 56
>>> dirb["a"]
56
>>> dirb["a"] = 57
>>> dirb["a"]
57
>>> dirb["a"] = 58
>>> dirb["a"]
58
>>> dirb["a"] = 59
>>> dirb["a"]
59
>>> dirb["a"] = 60
>>> dirb["a"]
60
>>> dirb["a"] = 61
>>> dirb["a"]
61
>>> dirb["a"] = 62
>>> dirb["a"]
62
>>> dirb["a"] = 63
>>> dirb["a"]
63
>>> dirb["a"] = 64
>>> dirb["a"]
64
>>> dirb["a"] = 65
>>> dirb["a"]
65
>>> dirb["a"] = 66
>>> dirb["a"]
66
>>> dirb["a"] = 67
>>> dirb["a"]
67
>>> dirb["a"] = 68
>>> dirb["a"]
68
>>> dirb["a"] = 69
>>> dirb["a"]
69
>>> dirb["a"] = 70
>>> dirb["a"]
70
>>> dirb["a"] = 71
>>> dirb["a"]
71
>>> dirb["a"] = 72
>>> dirb["a"]
72
>>> dirb["a"] = 73
>>> dirb["a"]
73
>>> dirb["a"] = 74
>>> dirb["a"]
74
>>> dirb["a"] = 75
>>> dirb["a"]
75
>>> dirb["a"] = 76
>>> dirb["a"]
76
>>> dirb["a"] = 77
>>> dirb["a"]
77
>>> dirb["a"] = 78
>>> dirb["a"]
78
>>> dirb["a"] = 79
>>> dirb["a"]
79
>>> dirb["a"] = 80
>>> dirb["a"]
80
>>> dirb["a"] = 81
>>> dirb["a"]
81
>>> dirb["a"] = 82
>>> dirb["a"]
82
>>> dirb["a"] = 83
>>> dirb["a"]
83
>>> dirb["a"] = 84
>>> dirb["a"]
84
>>> dirb["a"] = 85
>>> dirb["a"]
85
>>> dirb["a"] = 86
>>> dirb["a"]
86
>>> dirb["a"] = 87
>>> dirb["a"]
87
>>> dirb["a"] = 88
>>> dirb["a"]
88
>>> dirb["a"] = 89
>>> dirb["a"]
89
>>> dirb["a"] = 90
>>> dirb["a"]
90
>>> dirb["a"] = 91
>>> dirb["a"]
91
>>> dirb["a"] = 92
>>> dirb["a"]
92
>>> dirb["a"] = 93
>>> dirb["a"]
93
>>> dirb["a"] = 94
>>> dirb["a"]
94
>>> dirb["a"] = 95
>>> dirb["a"]
95
>>> dirb["a"] = 96
>>> dirb["a"]
96
>>> dirb["a"] = 97
>>> dirb["a"]
97
>>> dirb["a"] = 98
>>> dirb["a"]
98
>>> dirb["a"] = 99
>>> dirb["a"]
99
>>> dirb["a"] = 100
>>> dirb["a"]
100

```

Stack Overflow page showing tagged questions and a line graph titled "Posting rate in python.comp.python.general". The graph shows the number of messages per day over time, with a peak around 01.01.05 and a general upward trend.



Where to get help?

Communities

Python

Google Python Class Day 2 Part 2
The Google Code Channel | 992 videos | Subscribe

27,955 Likes

2 Part 2 by GoogleDevelopers

PyCon 2011: Algorithmic Generation of OpenGL Geometry
PyCon US Videos - 2009, 2010, 2011

8 Comments | Hey, the slides (which include an RST which contains a reasonable

○○○○
○○○
○○○
●○○○○○○○○○

○○○○○○○○○○
○○○
○○○

○○○○○
○○○○
○○○○○

Some nice features

nice features

```
>>> x = 5
>>> 1 < x < 10
True
>>> 10 < x < 20
False
```

3


```
○○○○
○○○
○●○○○○○○○○
```

```
○○○○○○○○○○
○○○
○○○
```

```
○○○○○
○○○
○○○○○
```

Some nice features

nice features

```
>>> re.compile(
    "[a-z0-9]*" # zero or more small letters or numbers
    "$",       # followed by an end of line
    re.DEBUG   # explain what we're doing
)
max_repeat 0 65535
in
    range (97, 122)
    range (48, 57)
at at_end
```

○○○○
○○○
○○○
○○●○○○○○○○○

○○○○○○○○○○
○○○
○○○

○○○○○
○○○○
○○○○○

nice features

```
>>> x=(x**2 for x in [0,1,2,3,4] if x>0)
<generator object>
>>> sum(x)
30
```

○○○○
○○○
○○○●○○○○○○○

○○○○○○○○○○
○○○
○○○

○○○○○
○○○
○○○○○

nice features

```
a = [10, 20, 30, 40, 50]
for index, item in enumerate(a):
    print index, item
```

```
0 10
1 20
2 30
3 40
4 50
```

○○○○
○○○
○○○●○○○○○○

○○○○○○○○○○
○○○
○○○

○○○○○
○○○○
○○○○○

nice features

```
>>> a, b = 1, 2
>>> b, a = a, b
>>> a, b
(2, 1)
```

○○○○
○○○
○○○
○○○○○●○○○○○

○○○○○○○○○○
○○○
○○○
○○○

○○○○○
○○○○
○○○○○
○○○○○○

nice features

```
@cache  
def somethingdifficult():  
    time.sleep(1000)
```

1

○○○○
○○○
○○○○○○●○○○○

○○○○○○○○○○
○○○
○○○○

○○○○○
○○○○
○○○○○○

nice features

```
def point(x, y):  
    # do some magic  
point(3, 4)  
point(3, y=4)  
point(x=3, y=4)  
a_tuple = (3, 4)  
a_dict = {'y': 3, 'x': 2}  
draw_point(*point_foo) # pass in each element  
draw_point(**point_bar) # pass named elements
```

○○○○
○○○
○○○
○○○○○○○●○○○

○○○○○○○○○○
○○○
○○○

○○○○○
○○○○
○○○○○

nice features

```
def add(x, y):  
    """  
    add 2 numbers  
    >>> add(3, 4)  
    7  
    """  
  
import doctest  
doctest.testmod()
```

○○○○
○○○
○○○
○○○○○○○○●○○

○○○○○○○○○○
○○○
○○○
○○○○

○○○○○
○○○○
○○○○○

nice features

```
"We are at {lat},{lon}".format(lat=52, lon=3)
```

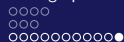

○○○○
○○○
○○○
○○○○○○○○○●○

○○○○○○○○○○
○○○
○○○
○○○

○○○○○
○○○○
○○○○○

nice features

```
>>> a = {1,2,3,4} # set([]) for python < 2.7?
>>> b = {3,4,5,6}
>>> a | b # Union
{1, 2, 3, 4, 5, 6}
>>> a & b # Intersection
{3, 4}
```



nice features

```
>>> str(round(1234.5678, -2))  
"1200.0"
```

○○○○
○○○
○○○
○○○○○○○○○○○○○○○○○○○○
○○○
○○○
○○○○○○○○
○○○○
○○○○○
○○○○○○

Outline

- 1 Setting up
 - Working environment
 - Where to get help?
 - Some nice features
- 2 The language
 - Data types
 - Reflection and namespaces
 - Performance
- 3 Python nice libraries
 - Glue
 - Plotting
 - Gis
- 4 Python common surprises

```
oooo
ooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
ooo
ooo
```

```
ooooo
oooo
ooooo
```

What aspects are relevant when choosing a language?

- People (What are the other people making?)
- Paradigma (Object Oriented, Procedural, Functional)
- Help (Documentation, community)
- Data types (dict, list, strings, numbers)
- Type system (int a = 1 vs a = 1)
- Syntax (Keywords, whitespace, braces)
- Libraries (What can you reuse of others?)

```

○○○○
○○○
○○○○○○○○○○

```

```

○○○○○○○○○○
○○○
○○○
○○○

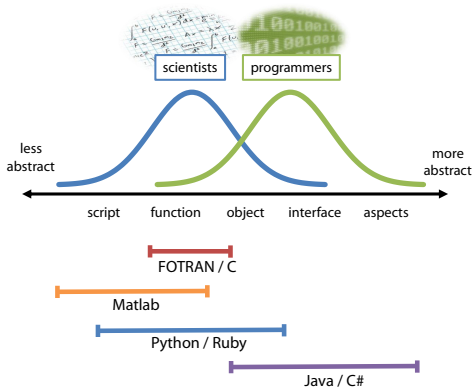
```

```

○○○○○
○○○
○○○○○

```

People and paradigma



```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
●○○○○○○○○○
○○○
○○○
```

```
○○○○○
○○○
○○○○○
```

Python

```
>>> x1 = 1 # integer
>>> x2 = 2.0 # float
>>> x3 = "three" # string
>>> x4 = [4,4,4,4] # list
>>> x5 = {5, "five"} # set
>>> x6 = {"six":6} #
    dictionary
>>> x7 = (4,4,4,4) # tuple
```

Matlab

```
>> x1 = int16(1) % integer
    (16bit)
>> x2 = 2 % double
>> x3 = "three" % string
>> x7 = {4,4,4,4} % cell
>> %x5 no matlab equivalent
>> x6 = struct('six',6) %
    not quite the same
>> %x7 no matlab equivalent
```

```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○●○○○○○○○○
○○○
○○○
```

```
○○○○○
○○○○
○○○○○
```

Python

```
>>> x1 = 1 # integer
>>> x2 = 2.0 # float
>>> x3 = "three" # string
>>> x4 = [4,4,4,4] # list
>>> x5 = {5, "five"} # set
>>> x6 = {"six":6} #
    dictionary
>>> x7 = (4,4,4,4) # tuple
```

Matlab

```
>> x1 = int16(1) % integer
    (16bit)
>> x2 = 2 % double
>> x3 = "three" % string
>> x7 = {4,4,4,4} % cell
>> %x5 no matlab equivalent
>> x6 = struct('six',6) %
    not quite the same
>> %x7 no matlab equivalent
```

```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○○●○○○○○○○
○○○
○○○
```

```
○○○○○
○○○○
○○○○○
```

Data types

Python

```
>>> x1 = 1
>>> x1 == 1
True
>>> x1 + 1
2
>>> x1 + 2.0
3.0
>>> x1 + "three"
... unsupported +: 'int'
and 'str'
>>> 2 * "three"
'threethree'
>>> x2 = 2.0
>>> x2 * "three"
... can't multiply sequence
by 'float'
```

Matlab

```
>> x1=1;
>> x1==1;
>> x1+1;
>> x1+2.0;
>> x1+'three'
ans =
    117    105    115    102
         102
>> 2 * 'three'
ans =
    232    208    228    202
         202
>> 2.0*'three'
ans =
    232    208    228    202
         202
```



```
oooo
ooo
ooooo
oooooooooooo
```

```
ooo●oooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

Python

```
>>> 9223372036854775807 + 1
9223372036854775808L
>>> 2/3 # 2//3 is explicit
integer division
0 (0.67 in python3)
```

Matlab

```
>> int64
(9223372036854775807)+1
ans =
9223372036854775807
>> int64(2)/int64(3)
ans =
1
```

```

○○○○
○○○
○○○
○○○○○○○○○○

```

```

○○○○●○○○○
○○○
○○○

```

```

○○○○○
○○○○
○○○○○

```

Data types

Python

```

>>> 2.0.is_integer()
True
>>> 2.5.as_integer_ratio()
(5, 2)
>>> 2.0.imag
0.0

```

Matlab

```

>> isinteger(2.0)
ans =
    0
>> [a,b] = rat(2.5)
a =          b =
    5          2
>> imag(2.0)
ans =
    0

```

```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○●○○○○
○○○
○○○○
```

```
○○○○○
○○○
○○○○○
```

Data types

```
>>> a = "Фёдор"
>>> len(a)
10
>>> print(a)
Фёдор
>>> a = u"Фёдор"
>>> len(a)
5
>>> a.encode("ascii")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-4: ordinal no
t in range(128)
>>> a.encode("ascii", "xmlcharrefreplace")
'&#1060;&#1105;&#1076;&#1086;&#1088;'
```

```

○○○○
○○○
○○○
○○○○○○○○○○

```

```

○○○○○○●○○○
○○○
○○○
○○○

```

```

○○○○○
○○○○
○○○○○

```

Python

```

>>> x1 = array([[1,2,3],
                [4,5,6]])
>>> x1
array([[1, 2, 3],
        [4, 5, 6]])
>>> x1[0,0]
1
>>> x1[1:,1:]
array([[5, 6]])

```

Matlab

```

>> x1 = [1 2 3; 4 5 6]
x1 =
     1     2     3
     4     5     6
>> x1(1,1)
ans =
     1
>> x1(2:end, 2:end)
ans =
     5     6

```

```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○●○○
○○○
○○○○
```

```
○○○○○
○○○○
○○○○○○
```

Python

```
>>> x1*x1
array([[ 1,  4,  9],
       [16, 25, 36]])
>>> x1.dot(x1.T)
array([[14, 32],
       [32, 77]])
```

Matlab

```
>> x1 .* x1
>> x1 * x1
```

```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○●○
○○○
○○○
```

```
○○○○○
○○○○
○○○○○
```

Python

```
>>> print(np.zeros((100,100)))
[[ 0.  0.  0. ...,  0.]
 [ 0.  0.  0. ...,  0.]
 ...,
 [ 0.  0.  0. ...,  0.]
 [ 0.  0.  0. ...,  0.]]
```

Matlab

```
>> zeros(100)
```

```
ans =
```

Columns 1 through 13

0	0	0	0
0	0	0	0
0	0	0	0

```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○●
○○○
○○○
```

```
○○○○○
○○○○
○○○○○
```

Python

```
>>> a = np.array
      ([[1,2,3],[4,5,6]])
>>> b = a[:2,:2]
>>> a[0,0] = 7
>>> print(a)
[[7 2 3]
 [4 5 6]]
>>> print(b)
[[7 2]
 [4 5]]
```

Matlab

```
a = [1 2 3; 4 5 6];
b = a(1:2,1:2);
a(1,1) = 7
a =
      7      2      3
      4      5      6
b
b =
      1      2
      4      5
```

```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
●○○○
○○○○
```

```
○○○○○
○○○○
○○○○○
```

Python

```
>>> a = 1
>>> a
1
>>> type(a)
<type 'int'>
```

Matlab

```
>> a = 1
```

```
a =
```

```
1
```

```
>> whos a
```

Name	Class
a	double


```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○●○○
○○○
```

```
○○○○○
○○○
○○○○○
```

Python

```
import numpy
numpy.array([])
import numpy as np
np.array([])
from numpy import *
array([])
from numpy import array
array([])
```

Matlab

```
% Matlab>7.6 +parallel/+gpu
/GPUArray.m
import parallel.gpu.*
GPUArray([])
```

```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○○●
○○○○
```

```
○○○○○
○○○○
○○○○○
```

Python

```
>>> dir(1)
['__abs__', ..., 'bit_length', '
    conjugate', ...]
>>> inspect.getfile(inspect)
'/opt/local/.../python2.7/inspect.pyc'
>>> def a(a=1):
...     pass
>>> inspect.getcallargs(a)
{'a': 1}
>>> getframeinfo(currentframe())
Traceback(filename='<stdin>', lineno=1,
...)
```

Matlab

```
% no equivalent to dir (I think)
>> which which
built-in (/Applications/MATLAB_R2011a.
    app/toolbox/matlab/general/which)
>> ?object % only for objects
>> dbstack % only in debugging?
```



Python

Startup do nothing and shutdown time.

```
$time python -c "pass"
real 0m1.471s
user 0m0.030s
sys 0m0.036s
```

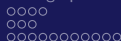
Matlab

Startup do nothing and shutdown time.

```
$ time matlab -r exit
real 1m8.891s
user 0m11.889s (2s with -
      nodesktop -nojvm)
sys 0m3.184s
```

Other languages

C: 0.000s, Java: 0.3s, Perl 0.001s, Bash 0.001s

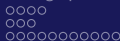


Python

```
>>> setupcode = """
import numpy
x=numpy.zeros((1000,1000))
"""
>>> timeit.timeit('a=x.dot(
x)',setupcode, number
=10)
1.5948209762573242
```

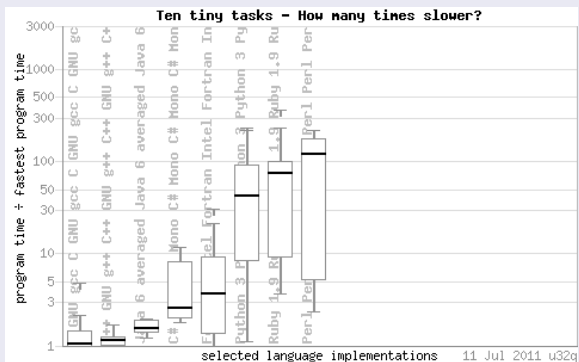
Matlab

```
>> x=zeros(1000);
>> tic;for i=1:10;a=x*x;end
;toc
Elapsed time is 1.796690
seconds.
```



Performance shootout

Benchmark (10 different problems)



shootout.alioth.debian.org

Laplace equation

Benchmark using different python techniques (500x500 grid for 100)

- Python: 1500.0s
- Python + NumPy: 29.3s
- Matlab: 29.0s
- Weave 2.3, 4.3, 9.5s
- Fortran 77: 2.9s
- Cython: 2.5s
- Pure C++: 2.16s

www.scipy.org/PerformancePython

```
oooo
ooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
ooo
ooo
```

```
ooooo
oooo
ooooo
```

Outline

- 1 Setting up
 - Working environment
 - Where to get help?
 - Some nice features
- 2 The language
 - Data types
 - Reflection and namespaces
 - Performance
- 3 Python nice libraries
 - Glue
 - Plotting
 - Gis
- 4 Python common surprises

```
oooo
ooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
ooo
ooo
```

```
ooooo
oooo
ooooo
```

- glue (mlabwrap: matlab, ctypes: dll's, fwrap: fortran90)
- plotting (2d: matplotlib, 3d: mayavi, graphs: networkx)
- gis (gdal: data + operations, pyproj: projections)
- data (pydap+netcdf: scientific, sqlalchemy: relational)


```
○○○○  
○○○  
○○○  
○○○○○○○○○○
```

```
○○○○○○○○○○  
○○○  
○○○
```

```
●○○○○  
○○○  
○○○○○
```

Talking to matlab

```
>>> import numpy  
>>> from mlabwrap import mlab  
>>> X = numpy.random.random((500,500))  
>>> mlab.imshow(X)  
array([[ 174.00366211]])
```

Talking to a dll

```
>>> import ctypes
>>> libz = ctypes.CDLL('/opt/local/lib/libz.dylib')
>>> ver = libz.zlibVersion
>>> ver.restype = ctypes.c_char_p
>>> ver()
'1.2.5'
```

5

Wrapping f90

```

function add(x, y, z)
  implicit none
  ! add two vectors and secretly add 1 to the first
    element of x
  real(kind=8), dimension(:, :), intent(inout) :: x
  real(kind=8), dimension(:, :), intent(inout) :: y
  real(kind=8), dimension(:, :), intent(inout) :: z
  integer :: add
  z = x + y
  add = 1
end function add

```

```
oooo
ooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
ooo
oooo
```

```
ooo●o
ooo
ooooo
```

Wrapping f90

```
fwrapc --name=somefunction --fcompiler=gnu95 test.f90
cd somefunction
python setup.py install
python
```

```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○○○
○○○
```

```
○○○○●
○○○
○○○○○
```

Wrapping f90

```
>>> import numpy
>>> from somefunction import *
>>> x = numpy.ones((5,5), dtype=fwr_real_8, order='F')
>>> y = numpy.ones((5,5), dtype=fwr_real_8, order='F')
>>> z = numpy.empty((5,5), dtype=fwr_real_8, order='F')
>>> (a,b,c,d) = add(x,y,z)
>>> a
1
>>> b is x and c is y
True
>>> d
array([[ 2.,  2.,  2.,  2.,  2.],
       [ 2.,  2.,  2.,  2.,  2.],
       [ 2.,  2.,  2.,  2.,  2.],
       [ 2.,  2.,  2.,  2.,  2.],
       [ 2.,  2.,  2.,  2.,  2.]])
```

```
○○○○
○○○
○○○
○○○○○○○○○○
```

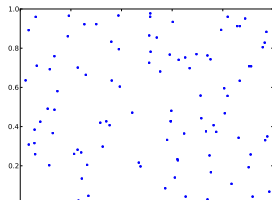
```
○○○○○○○○○○
○○○
○○○
○○○
```

```
○○○○○
●○○○
○○○○○
```

Plotting

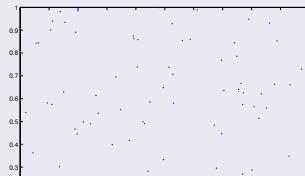
Python

```
>>> plt.plot(random.uniform(
    (0,1,100),
                random.uniform
                (0,1,100),
                ','))
>>> plt.savefig('
    plot1python.pdf')
```



Matlab

```
>> h = plot(
    random('unif',0,1,1,100)
        ,
    random('unif',0,1,1,100)
        ,
    ',.')
h =
    174.0028
>> saveas(h,'plot1matlab.
    pdf')
```



```

○○○○
○○○
○○
○○○○○○○○○○

```

```

○○○○○○○○○○
○○○
○○○
○○○

```

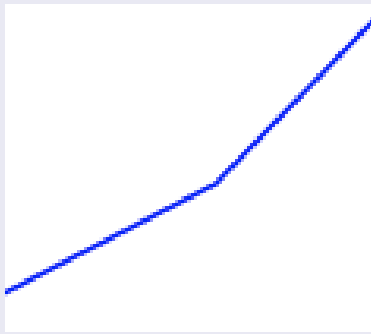
```

○○○○○
○●○○○
○○○○○

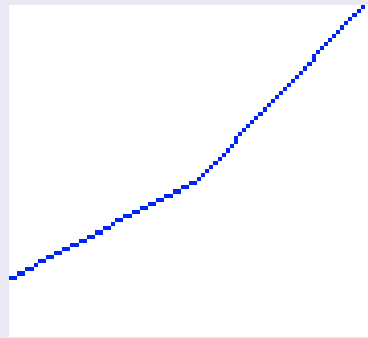
```

Plotting

Python



Matlab



```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○○○
○○○
```

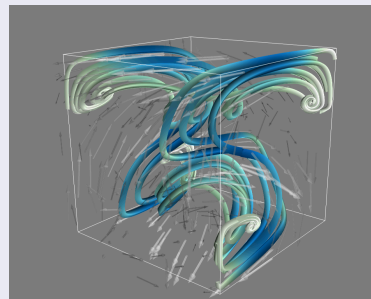
```
○○○○○
○○●○○
○○○○○
```

3d plotting

```
import numpy as np
x, y, z = np.mgrid[0:1:20j,
                   0:1:20j, 0:1:20j]

u = np.sin(np.pi*x) * np
    .cos(np.pi*z)
v = -2*np.sin(np.pi*y) * np
    .cos(2*np.pi*z)
w = np.cos(np.pi*x)*np.sin(
    np.pi*z) +
    np.cos(np.pi*y)*np.sin
    (2*np.pi*z)
mlab.quiver3(u,v,w)
mlab.flow(u,v,w)
```

Mayavi plot




```

○○○○
○○○
○○○○○○○○○○

```

```

○○○○○○○○○○
○○○
○○○○

```

```

○○○○○
○○○●
○○○○○

```

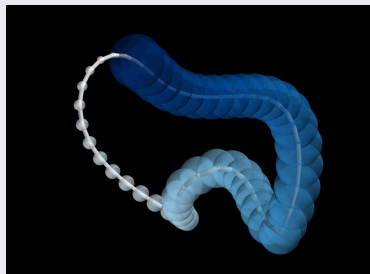
networkx

```

import networkx as nx
H = nx.cycle_graph(50)
G = nx.
    convert_node_labels_to_integers(
        H)
# layout in 3d
pos = nx.spring_layout(G,
    dim=3)
mlab.points3d( ... )
mlab.pipeline.tube( ... )

```

Networkx plot using mayavi



networkx.lanl.gov

```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○○○
○○○
```

```
○○○○○
○○○
●○○○○
```

Python with GDAL

```
>>> dataset = ogr.Open('test.kml')
>>> layer = dataset[0]
>>> feature = layer[0]
>>> geometry = feature.geometry()
>>> geometry.GetPoint()
(-122.0822035425683, 37.42228990140251,
 0.0)
>>> geometry.Buffer(3.0).ExportToKML()
'<Polygon>
  <outerBoundaryIs><LinearRing>
    <coordinates>
      -119.082203542568294,37.422289901402507
    ...
```

KML

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml
  /2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground.
      Intelligently places itself
      at the height of the underlying
      terrain.</description>
    <Point>
      <coordinates>
        -122.0822035425683,37.422289901402
      </coordinates>
    </Point>
  </Placemark>
</kml>
```

○○○○
○○○
○○○○○○○○○○

○○○○○○○○○○
○○○
○○○

○○○○○
○○○
○●○○○

Pyproj

```

from pyproj import Geod
old = Geod(ellps='bessel')
new = Geod(ellps='WGS84')
ams_lat = 52.35
ams_lon = 4.917
nyc_lat = 40.+(47./60.)
nyc_lon = -73.-(58./60.)
args = ams_lon,ams_lat,nyc_lon,nyc_lat
az12,az21, olddist = old.inv(*args) # 5872 km
az12,az21, newdist = new.inv(*args) # 5873 km

```

3

8

```
○○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○○○
○○○
```

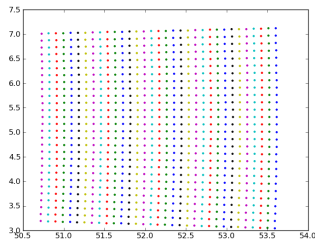
```
○○○○○
○○○
○○●○○
```

Python

```
url =
'http://opendap.deltares.nl/thredds/
dodsC/opendap/tno/ahn100m/mv100.nc
'

import pydap.client
ds = pydap.client.open_url(url)
ds.keys()
['x', 'y', 'longitude', 'latitude', '
epsg', 'wgs84', 'depth']
lat = ds['latitude']['latitude']
lon = ds['longitude']['longitude']
depth = ds['depth']['depth']
plt.plot(lat[::100,::100], lon
[::100,::100], '.')
```

ahn lat/lon



```
○○○○
○○○
○○○
○○○○○○○○○○
```

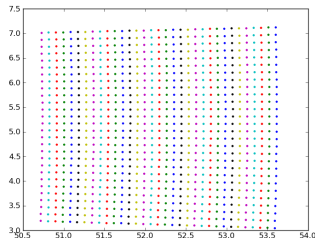
```
○○○○○○○○○○
○○○
○○○
○○○
```

```
○○○○○
○○○○
○○○○○
○○●○○
```

Python

```
lat100 = lat[:, :100, :100]
lon100 = lon[:, :100, :100]
idx = reduce(np.logical_and, [
    lat100 < 52.8,
    lat100 > 52.6,
    lon100 > 5.7,
    lon100 < 6.1])
xidx, yidx = np.where(idx)
xslice = slice(xidx.min()*100, xidx.max()
              *100)
yslice = slice(yidx.min()*100, yidx.max()
              *100)
noplant = lat[xslice, yslice]
noplon = lon[xslice, yslice]
nopdep = depth[xslice, yslice]
```

ahn lat/lon



```
○○○○
○○○
○○○
○○○○○○○○○○
```

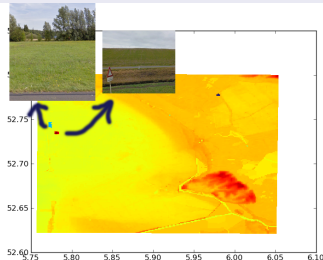
```
○○○○○○○○○○
○○○
○○○
○○○
```

```
○○○○○
○○○○
○○○○●○
```

Python

```
plt.pcolormesh(noplon,
               noplat, nopdep)
```

ahn lat/lon



```
○○○○
○○○
○○○
○○○○○○○○○○
```

```
○○○○○○○○○○
○○○
○○○
○○○
```

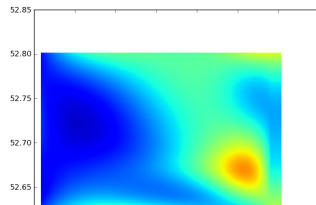
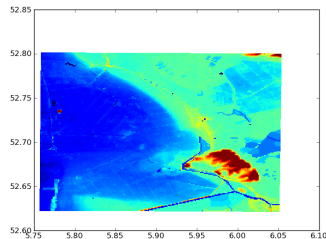
```
○○○○
○○○
○○○○●
```

Can we use a rough height map

```
from scipy.interpolate import bisplrep,
    bisplev
roughdep = nopdep[:,10,:]
roughlat = noplat[:,10,:]
roughlon = noplon[:,10,:]
f = bisplrep(roughlon,roughlat,roughdep)
# expects increasing x and y
newdep = bisplev(noplon[:,0], noplat
    [0,:,-1], f)[:,:,-1].T
plt.pcolormesh(noplon[:,0], noplat[0,:],
    newdep, vmin=-5, vmax=5)
```

4

ahn 100/1000



```
oooo
ooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
ooo
ooo
```

```
ooooo
oooo
ooooo
```

Outline

- 1 Setting up
 - Working environment
 - Where to get help?
 - Some nice features
- 2 The language
 - Data types
 - Reflection and namespaces
 - Performance
- 3 Python nice libraries
 - Glue
 - Plotting
 - Gis
- 4 Python common surprises

○○○○
○○○
○○○○○○○○○○

○○○○○○○○○○
○○○
○○○○

○○○○○
○○○○
○○○○○○

```
>>> "%s=%s" % (str(3*0.3), repr(3*0.3))  
'0.9=0.8999999999999999'
```

2

```
oooo
ooo
ooooo
```

```
oooooooooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

```
x = 1.0 / 3
y = 0.333333333333333
print x #: 0.333333333333333
print y #: 0.333333333333333
print x == y #: False
```

2

repr prints too many digits:

7

```
print repr(x) #: 0.3333333333333333331
print repr(y) #: 0.3333333333333300003
print x == 0.333333333333333333 #: True
```

```
oooo
ooo
ooooo
```

```
oooooooooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

```
>>> def a(a=[]):
...     a.append(1)
...     print(a)
...
>>> a()
[1]
>>> a()
[1, 1]
```

4

9

```
oooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

```
>>> 01
1
>>> 07
7
>>> 08
File "<stdin>", line 1
    08
    ~
SyntaxError: invalid token
```

```
oooo
ooo
ooooo
```

```
oooooooooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

```
>>> a=[[1,2,3]]*3
```

```
[[1, 2, 3],
 [1, 2, 3],
 [1, 2, 3]]
```

```
>>> a[0][0] = 2
```

```
[[4, 2, 3],
 [4, 2, 3],
 [4, 2, 3]]
```

1

6

```
oooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

```
>>> i = 1
```

```
>>> ++i
```

```
1
```

```
>>> i
```

```
1
```

2

```
oooo
ooo
oooooooooooo
```

```
oooooooooooo
ooo
oooo
```

```
ooooo
oooo
oooooo
```

```
    a = 1
a = 2
```