Python i software lliure per a ús científic i en enginyeria

Francesc Alted

18 i 25 d'Octubre, Universitat Jaume I

Objectius del taller (part I)

- Introducció al software lliure
- Introducció a Python
- Python i càlcul matricial: Numeric i numarray
- Generació de gràfics amb IPython i matplotlib
- Tractament bàsic d'imatges usant numarray i matplotlib

Objectius del taller (part II)

- Introducció a SciPy, MATLAB portat a Python (i lliure!)
- Salvaguarda i recuperació d'informació: PyTables
- Exercicis que combinen Python, IPython, numarray, Numeric, matplotlib i PyTables com a mostra de la seua productivitat i eficàcia.

Per què software lliure?

- Fàcil de provar: descarregar, compilar i... usar!
  - Si s'usa una distribució de Linux, això pot ser tan senzill com fer:
    $ apt-get install python-tables
- Suport a través d'una comunitat molt activa. En molts casos, es té accés directe als autors del software.
- La qualitat depèn dels casos, però generalment és molt acceptable (versions ridículament baixes poden deixar-vos bocabadats).

Com escollir un bon projecte de software lliure?

- Evidentment, no tot el software lliure té per què ser bo...
- Recomanacions d'elecció:
  - S'adeqüe bé a les nostres necessitats
  - Comprovar que no és un projecte “mort” (però així i tot encara pot ser d'ajut!)
  - Si té una llista de distribució, consulteu-la per veure les opinions de la gent sobre el software
  - És convenient que dispose de bona documentació
  - Si hi ha alternatives, feu una búsqueda per Internet per veure que diu el personal...

Com funciona el software lliure?

- En principi, és la tasca de voluntaris distribuïts per la Xarxa que no cobren (al menys directament) pel seu esforç
- Normalment, el suport que es dóna és suficient, però a molts desenvolupadors se'ls pot contractar un manteniment
- Si es té una necessitat concreta:
  - Demanar ajut (alguns usuaris o els desenvolupadors principals et poden ajudar)
  - Fer tu mateix el codi (no oblides contribuir-lo)

Debat software lliure respecte software propietari

- Hi ha gent que pensa que el software lliure és millor que el software propietari i gent que pensa al revés
- El més intel·ligent és traure profit d'ambdós mons:

El software lliure i el propietari han de ser eines complementàries i no exclusives

Introducció a Python

- Característiques bàsiques:
  - Interpretat => interactivitat
  - Flexibilitat estructura de dades => manejabilitat
  - Gramàtica minimalista => Senzill de dependre
  - Llibreria molt completa => “Piles” incloses
- Molt adequat per a les necessitats del científic/enginyer:
  - El treball d'anàlisi de dades requereix interactivitat per a ser productiu
  - Programar és considerat normalment com un “mal necessari”
  - Interpretat no implica necessàriament ineficiència => enllaç amb C i Fortran

Primer exemple en Python

```
$ python
Python 2.3.4 (#2, Sep 24 2004, 08:39:09)
[GCC 3.3.4 (Debian 1:3.3.4-12)] on linux2
Type "help", "copyright", "credits" or "license"
for more information.
>>> print 'Benvinguts al taller'
Benvinguts al taller
>>> ctrl-D  # per a eixir
$
```

Variables i expressions artimètiques

```
>>> inicial = 1000
>>> interes = 0.03
>>> numanys = 5
>>> any = 1
>>> while any < numanys:
...           inicial = inicial*(1+interes)
...           print any, inicial
...           any += 1
...
1  1060.9
2  1092.727
3  1125.50881
4  1159.2740743
```

Cadenes (Strings)

```
>>> a = "Hola que tal!"
>>> b = 'Python es tan facil com diuen?'
>>> a[:4]
'Hola'
>>> a[5:7]
'qu'
>>> a[5:8]
'que'
>>> a[3]
'a'
>>> a[5:8]
'que'
>>> b[::2]
'Pto stnfclcmdun'
```

Llistes

```
>>> noms = [ "Juan", "Pep", "Krystian"]
>>> noms[2]
'Krystian'
>>> len(noms)
3
>>> noms[1] = "Josep"
>>> noms
['Juan', 'Josep', 'Krystian']
>>> [1,2,3] + [3,4]
[1, 2, 3, 3, 4]
>>> a=[1, "Pep", 3.14, ["Juan", 7, 9], 10]
>>> a
[1, 'Pep', 3.1400000000000001, ['Juan', 7, 9], 10]
>>> a[3][2]
9
```

Tuples

- Molt paregudes a les llistes
- La diferència més important és que són immutables

```
>>> a = (1,4,5,-9)
>>> b = (7,)
>>> persona = (nom, cognom, telefon)
>>> a[2] = 3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
```

Diccionaris

- Són com una llista, però els seus índex poden ser de qualsevol tipus, no només enters

```
>>> a = {"usuari": "falted",
...       "home": "/users/alted",
...       "uid": 2514,}
>>> a
{'home': '/users/alted', 'usuari': 'falted', 'uid': 2514}
>>> a["uid"]
2514
>>> a["shell"] = "/bin/bash"
>>> a.keys()
['home', 'shell', 'usuari', 'uid']
>>> del a["usuari"]
>>> a
{'home': '/users/alted', 'shell': '/bin/bash', 'uid': 2514}
```

Estructures mixtes

```
>>> a={"float": 121.12, "tupla": (12,15),
       "llista":[[4.,1.5],[.5,2.5]]}
>>> a["llista"]
[[4.0, 1.5], [0.5, 2.5]]
>>> a["llista"][1]
[0.5, 2.5]
>>> a["llista"][1][1]
2.5
>>> a["llista"][1][1]
2.5
>>> a["tupla"]
(12, 15)
>>> type(a["tupla"])
<type 'tuple'>
```

Condicionals

Tests simples:

```
>>> if a < b:
...     z = b
... else:
...     z = a
```

Expressions booleanes:

```
>>> if b >= a and b <= c:
...   print 'b esta entre a i c'
>>> if not (b < a or b> c):
...   print 'b encara esta entre a i c'
```

Bucles

Bucle normal:

```
>>> for i in range(1,10):
...     print '2 a elevat a %d es %d' % (i, 2**i)
```

Optimitzat en memòria:

```
>>> for i in xrange(1,10):
...     print '2 a elevat a %d es %d' % (i, 2**i)
```

```
>>> range(5)
[0, 1, 2, 3, 4]
>>> range(1,8)
[1, 2, 3, 4, 5, 6, 7]
>>> range(0,14,3)
[0, 3, 6, 9, 12]
>>> range(8,1,-1)
[8, 7, 6, 5, 4, 3, 2]
```

Exemples d'ús de range()

Llistes comprensives

Bucle normal:

```
>>> a=[]
>>> for i in range(10):
...   if i**3 % 4:
...     a.append(i**2)
...
>>> a
[1, 9, 25, 49, 81]
```

Llista comprensiva

```
>>> a = [ i**2 for i in range(10) if i**3 % 4 ]
>>> a
[1, 9, 25, 49, 81]
```

Funcions

```
>>> def rest(a,b):
...     c = a/b
...     r = a - c*b
...     return r
>>> rest(10,5)
0
>>> rest(11,5)
1
>>> def divideix(a,b):
...     c = a/b
...     r = a - c*b
...     return (c,r)
>>> divideix(11,5)
(2, 1)
```

Classes

- S'usen per a definir nous tipus de dades, així com per a programació d'objectes.

```
class Pila:
    def __init__(self):
        self.pila = []
    def posa(self, objecte):
        self.pila.append(objecte)
    def lleva(self):
        return self.pila.pop()
    def longitud(self):
        return len(self.pila)
```

Mòduls

- Permet trencar programes llargs en troços més curts (--> llibreries)

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__name__', 'acos', 'asin', 'atan',
'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e', 'exp',
'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log',
'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh']
>>>
>>> help(math)    # --> proveu això en la consola
```

Excepcions

- Quan passa un error en el programa, es dispara una “excepció” i apareix un error explicatiu:

```
>>> for i in range(10):
...   f.append(i)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in ?
NameError: name 'f' is not defined
```

- Les excepcions es poden “capturar” usant les instruccions “try” i “except”. Útil si no es vol deixar el programa acabar en eixe cas.

Exercici

- Fer un programa que, donada una llista de cadenes de text, genere una altra sense cap d'elles estiga repetit.
  - Ex:de la llista [“a”, “b”, “c”, “c”, “a”] ha de generar: [“a”, “b”, “c”]
  - Pista: useu diccionaris

On recórrer en Python

- Documentació on-line:
  - http://www.python.org/doc (anglés)
  - http://diveintopython.org/toc/ (anglés)
  - http://marmota.act.uji.es/MTP/pdf/python.pdf
- Llibres:
  - Python 2.1 Bible (Brueck, Tanner, 2001)
  - Python Essential Reference (Beazley, 2001)
  - Python Cookbook (Martelli, Ascher, 2003)
- Llistes de distribució:
  - python-es@aditel.org (llista en castellà)
  - python@python.org (llista en anglés)

Matrius multidimensionals: NumPy i numarray (http://numpy.sf.net)

- NumPy (també anomenat Numeric) és un conjunt de mòduls en Python per a manejar còmoda, però eficientment, matrius multi-dimensionals
- Té les següents funcionalitats:
  - Funcions per manipular matrius: transposició, búsqueda de valors, selecció per índex, ...
  - Operacions d'àlgebra lineal (Lapack), transformades de Fourier, nombres aleatoris,...
- En el 2002 apareix un successor: **numarray**

Diferències entre NumPy i numarray

- numarray es presenta com un successor de NumPy i presenta múltiples avantatges sobre aquest:
  - Arrays que poden funcionar com índexs
  - Fitxers mapejats en memòria
  - Objectes RecordArray (dades heterogènies)
  - Arrays de cadenes de text
  - Propietats avançades: “byteswapping”, alineament arbitrari (tant “offset” com “striding”)
- Però: crear arrays menuts és més lent que NumPy => això està alentint l'adopció de numarray

Per què NumPy i numarray?

- Manipular llistes, tuples o diccionaris amb milions de nombres porta molt de temps
- Si Python vol competir amb llenguatges com Fortran, C, MATLAB o S, li cal suport per a manipular conjunts de dades homogenis de manera eficient
- NumPy i numarray proveeixen eixe suport

Introducció a numarray (I)

```
>>> from numarray import *
>>> v1=array([1, 2, 3, 4])
>>> print v1
[1 2 3 4]
>>> M1 = array(([0,1],[1,3]))
>>> print M1
[[0 1]
 [1 3]]
>>> print v1.shape
(4,)
>>> print M1.shape
(2, 2)
>>> print M1+M1
[[0 2]
 [2 6]]
```

Introducció a numarray (II)

```
In [12]: M1 = array(([0,1],[1,3]))
In [13]: M2 = array(([1,2],[0,3]))
In [14]: v2=array([1,2])
In [15]: matrixmultiply(M1,v2)
Out[15]: array([2, 7])
```

```
In [24]: M1*M2
Out[24]:                Multiplicació element
array([[0, 2],          a element!!
       [0, 9]])

In [25]: matrixmultiply(M1,M2)
Out[25]:                        Producte matricial
array([[ 0,  3],
       [ 1, 11]])
```

Funcions universals (ufuncs)

- Són funcions que es poden aplicar a tots els elements d'un array, però també a qualsevol seqüència Python (llistes, tuples):

```
>>> print greater([1,2,4,5],[5,4,3,2])
[0 0 1 1]
>>> print add([1,2,4,5], [1,2,4,5])
[2 4 8 10]
>>> add.reduce([1,2,3,4,5])
15                              #=1+2+3+4+5
>>> a = arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> print sin(a)
[ 0.          0.84147098  0.90929743  0.14112001
-0.7568025  -0.95892427
 -0.2794155   0.6569866   0.98935825  0.41211849]
```

Tipus de dades suportades (numarray i NumPy)

Numarray	Numarray Str	Numarray Code	Numeric Str	Numeric Code
Int8	'Int8'	'i1'	'Byte'	'1'
UInt8	'UInt8'	'u1'	'UByte'	
Int16	'Int16'	'i2'	'Short'	's'
UInt16	'UInt16'	'u2'	'UShort'	
Int32	'Int32'	'i4'	'Int'	'i'
UInt32	'UInt32'	'u4'	'UInt'	'u'
Int64	'Int64'	'i8'		
UInt64	'UInt64'	'u8'		
Float32	'Float32'	'f4'	'Float'	'f '
Float64	'Float64'	'f8'	'Double'	'd'
Complex32	'Complex32'	'c8'		'F'
Complex64	'Complex64'	'c16'	'Complex'	'D'
Bool	'Bool'			

Exercici

- Creeu una matriu 4x4 en doble precisió i tragueu la seua inversa
  - Pista 1: useu una matriu no singular, com:

    ```
    a = reshape(arange(16.0), (4,4)) + identity(4)
    ```

  - Pista 2: l'operació 'inverse' es troba a numarray.linear_algebra:

    ```
    from numarray import linear_algebra as la
    ```

- Multipliqueu la matriu inversa per l'original i comproveu que dóna la identitat

Funcionalitat en numarray

- convolve: Mòdul de convolucions i correlacions
- fft: Mòdul per a transformades de Fourier ràpides
- linear_algebra: Mòdul que proveeix una interfície senzilla per a operacions habituals d'àlgebra linial
- ma: Suport d'arrays amb màscares
- mlab: Funcions compatibles amb MATLAB
- random_array: Interfície d'alt nivell per a RANLIB
- nd_image: Funcions per a anàlisi d'imatges multi-dimesionals

Operacions d'Àlgebra Lineal

- Usa per defecte un mòdul Lapack_Lite que ve amb numarray
- Si es vol més velocitat, es pot linkar amb un Lapack optimitzat per a la plataforma (Atlas)

```
>>> from numarray import linear_algebra as la
>>> a = array([[1,2],[3,15]])
>>> print la.determinant(a)
9.0
>>> print b
[[ 1.    0.    0.    0. ]
 [ 0.    2.    0.    0.01]
 [ 0.    0.    5.    0. ]
 [ 0.    0.01 0.    2.5 ]]
>>> print la.eigenvalues(b)
[ 2.50019992 1.99980008 1.  5. ]
```

Operacions amb imatges

- numarray.nd_image pot manejar no tan sols imatges bidimensionals, sinó de dimensions més grans també
- Inclou funcions per a filtrat lineal i no-lineal, morfologia binària, interpolacions B-spline, mesures d'objectes...

```
>>> from numarray import nd_image as nd
>>> a = array([[1,2],[3,15]])
>>> nd.fourier_shift(a, 1.0)
array([[  1. +0.00000000e+00j,  -2. +2.44921271e-16j],
       [ -3. +3.67381906e-16j,  15. -3.67381906e-15j]])
```

numarray i LAPACK

- numarray ve amb un subconjunt de les llibreries LAPACK integrat: LAPACK lite.
- Això permet fer les operacions més bàsiques d'àlgebra lineal de manera ràpida.
- De moment, numarray no suporta bé el compilar contra llibreries LAPACK externes (ex: ATLAS).
- Això li impedeix traure rendiments màxims
- Se suposa que la situació canviarà en el futur proper.

LAPACK lite vs ATLAS

- ATLAS és una implementació eficient de LAPACK que aprofita l'arquitectura de la màquina.
- Amb una màquina Xeon IV @ 2.4 GHz:
  - Usant LAPACK lite, una multiplicació de matrius 500x500 genèrica (SGEMM) tarda 3.7 s ==> 70 MFlops
  - Usant ATLAS, la mateixa operació costa 0.48 s ==> 520 Mflops
- NumPy suporta enllaços contra llibreries externes ==> pot usar ATLAS

IPython (http://ipython.sf.net)

- Es tracta d'una consola en mode text per a Python però amb certes característiques interessants:
  - Introspecció dinàmica d'objectes (tecla '?')
  - Auto-completat de noms (tecla 'TAB')
  - Suporta cerques en històrics de comandos
  - Suport d'àlies
  - Accés a la Shell (tecla '!')
  - Auto-parèntesi de les cridades
  - I... el més interessant:
    Suport d'entorns de gràfics
    (GnuPlot i matplotlib)

matplotlib
http://matplotlib.sf.net

- Llibreria per a fer gràfics de qualitat i que tracta d'imitar la interfície de MATLAB
- Producte jove (< 1 any entre el públic), però d'alta qualitat
- Utilitzant al STSci (els autors de numarray) com a plataforma estàndard de gràfics. Estan ajudant en el seu desenvolupament.
- Integrada en IPython matplotlib

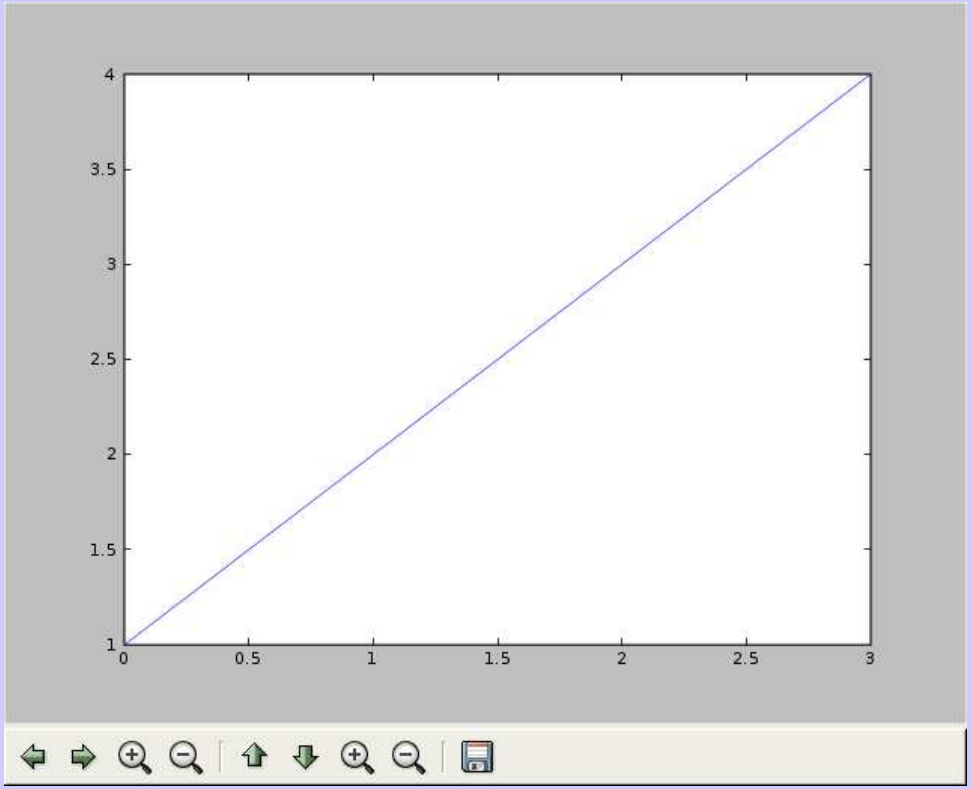
Un primer exemple

- Consola Python estàndard:

```
$ python
>>> from matplotlib.matlab import *
>>> plot([1,2,3,4])
>>> show()
```

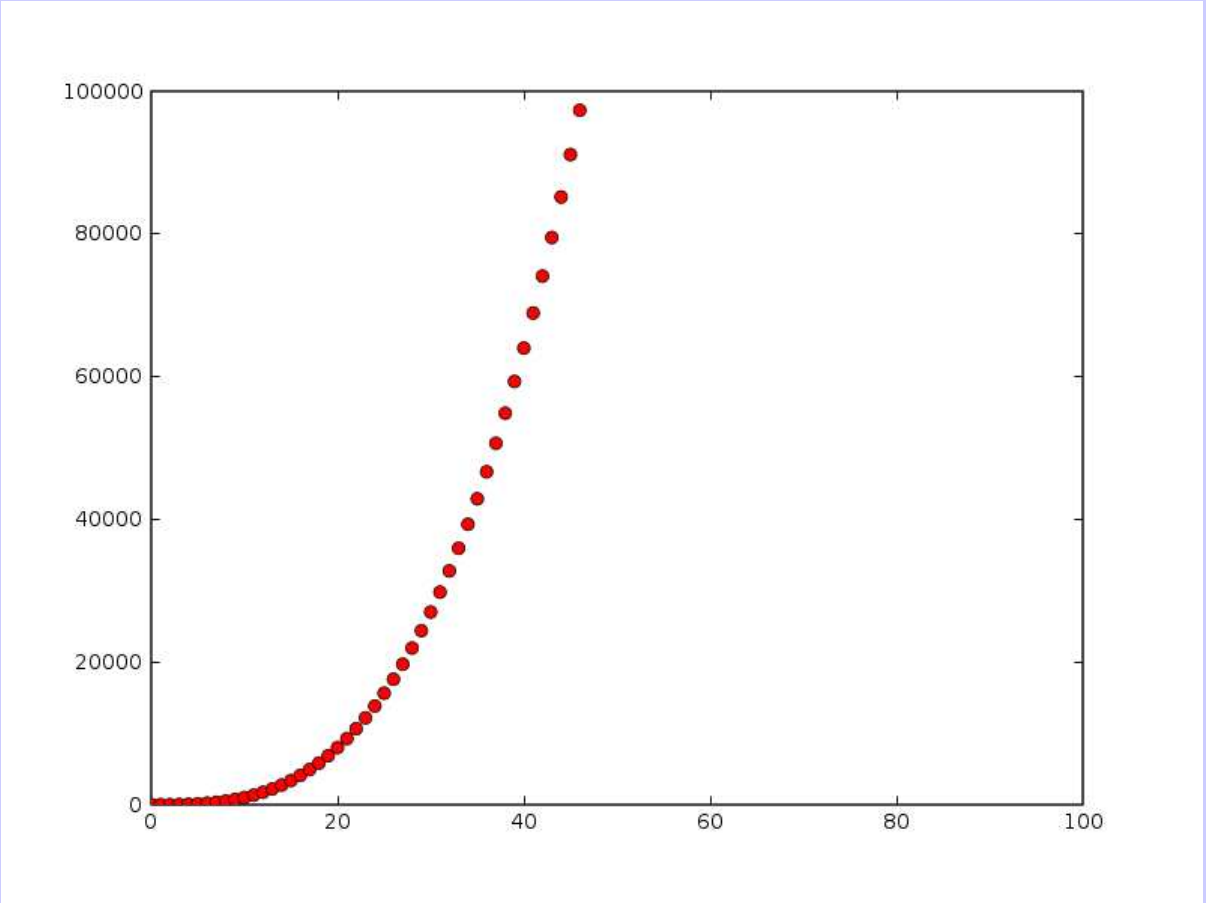
- Amb IPython:

```
$ ipython -pylab
In [1]: plot([1,2,3,4])
```



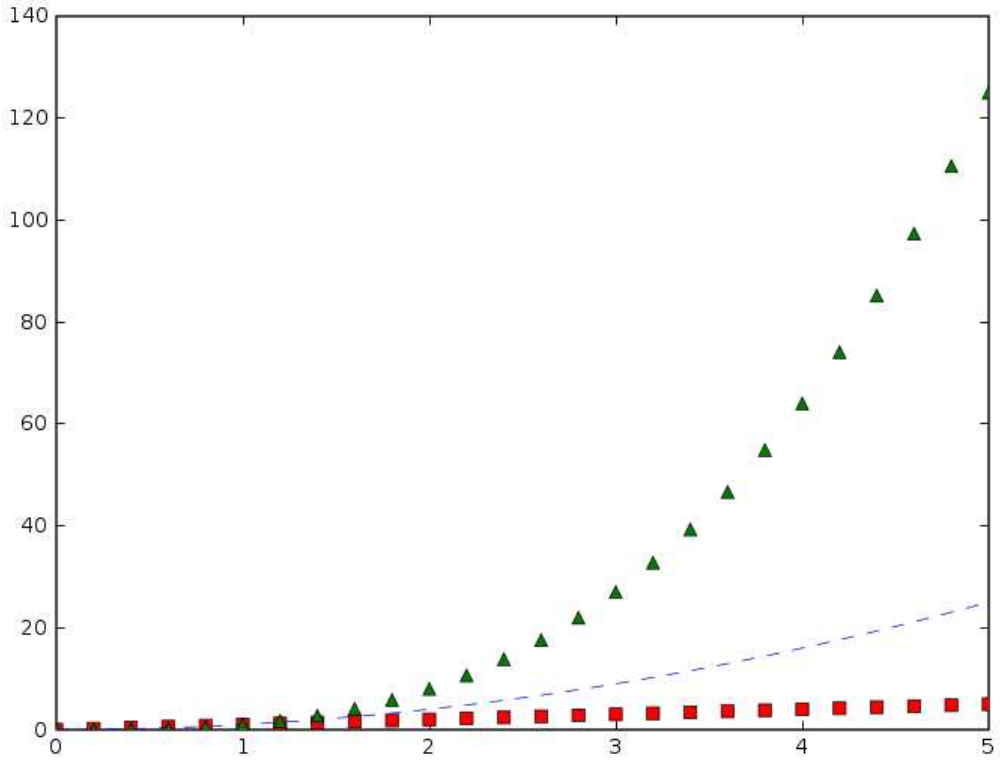
Canviant propietats del traç

```
plot(arange(100.)**3, 'ro')
axis([0,100,0,1000*100])
```

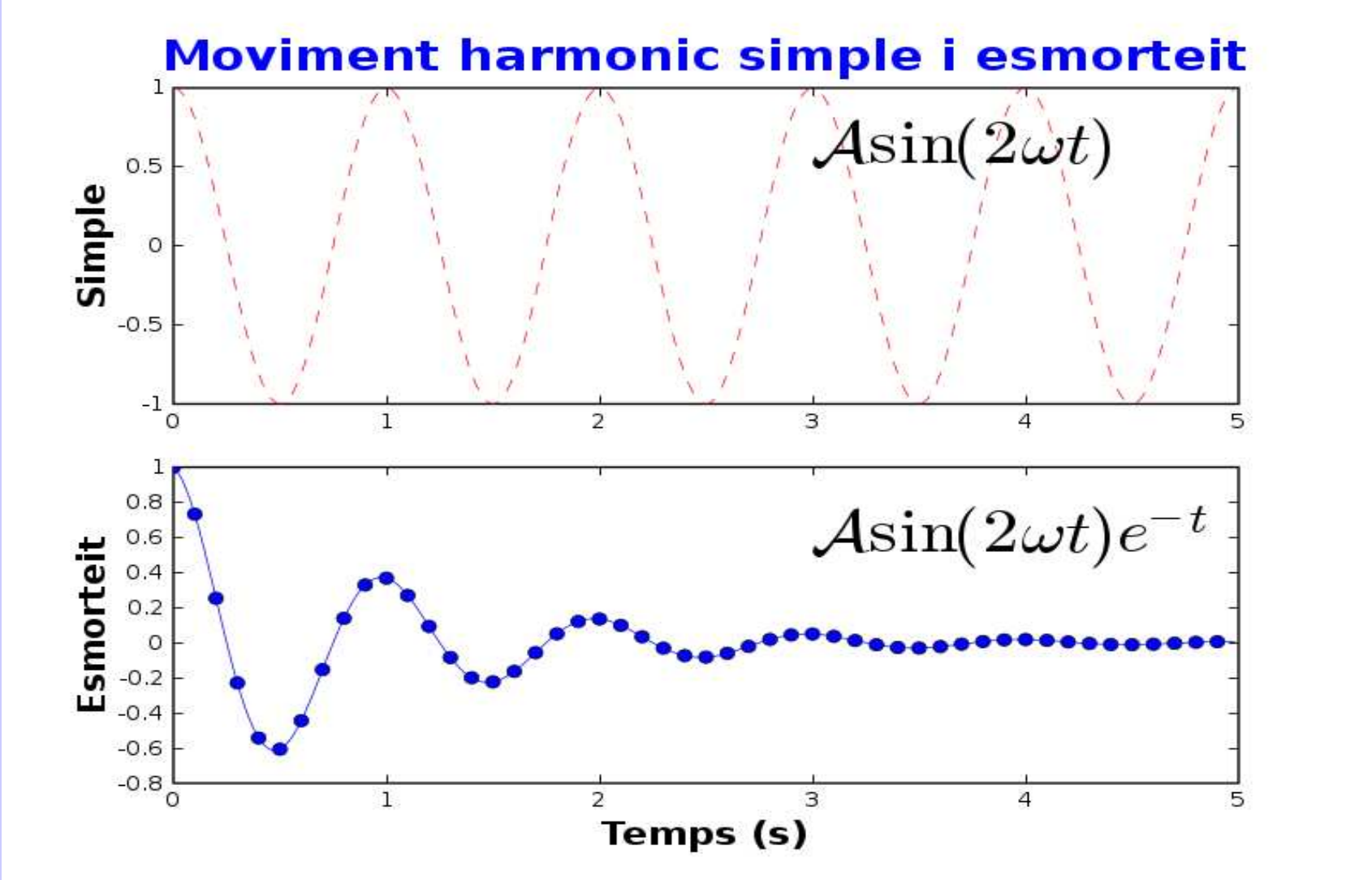


Mútiples línies en un sol comando

```
t = arange(0.0, 5.2, 0.2)
# quadrats rojos, guions blaus i triangles verds
plot(t, t, 'rs', t, t**2, 'b--', t, t**3, 'g^')
```



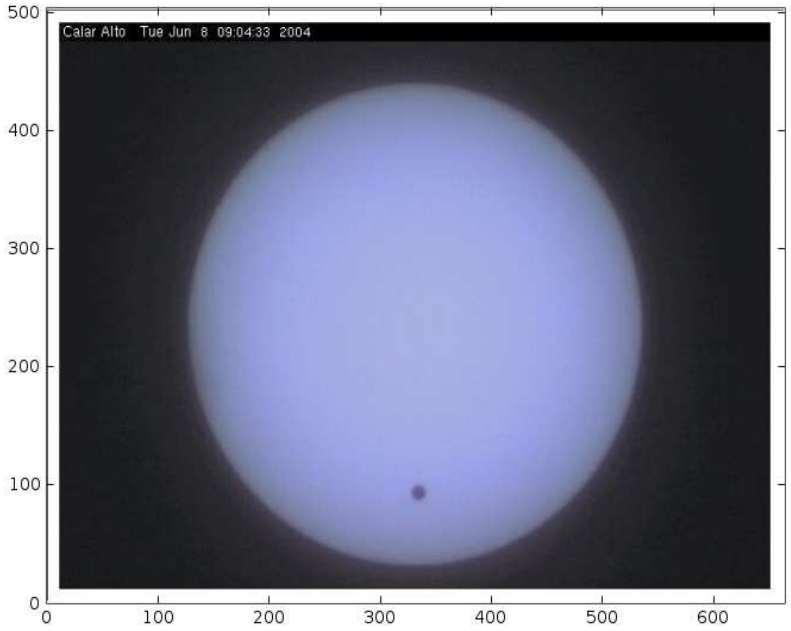
Un exemple més complet

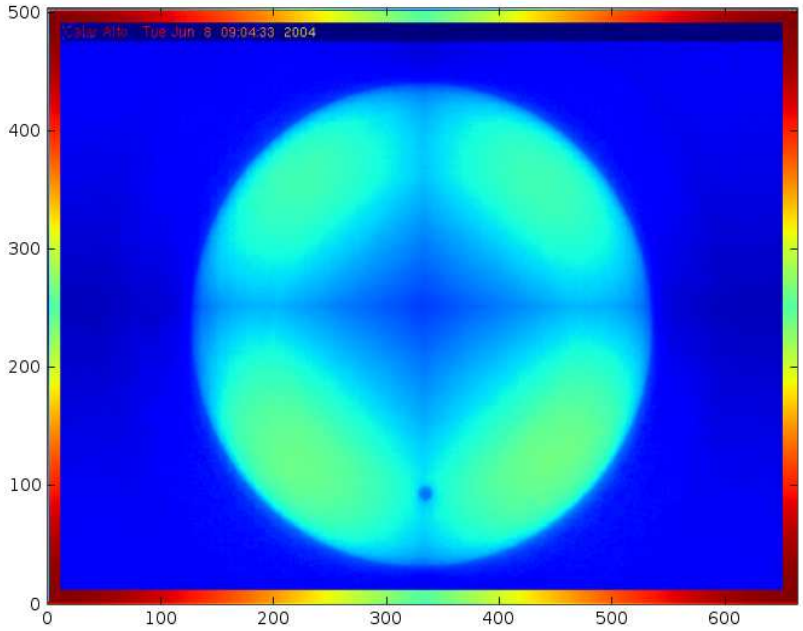


http://matplotlib.sourceforge.net/tutorial.html

Tractament d'imatges (I)

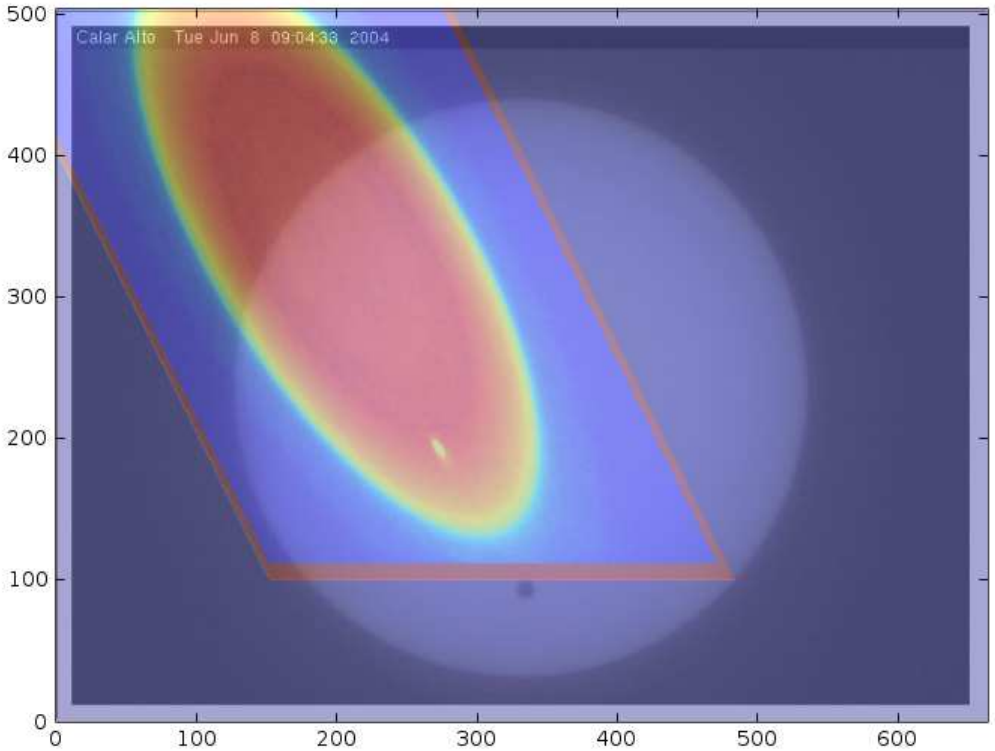
```
a=imread("venus.png")
imshow(a)
b=a[...,2]
from numarray import nd_image
imshow(nd_image.fourier_gaussian(b, 0.4))
```





Tractament d'imatges (II)

Transformacions
afins
i
Alpha blending




```
imshow(nd_image.affine_transform(b,[[1,0],[-1,2]],
100), alpha=.6)
```