Python i software lliure per a ús científic i en enginyeria (Part II)

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.

SciPy
http://www.scipy.org

- Llibreria de software obert orientada a càlcul científic
- És una col·lecció d'algorismes i funcions matemàtiques construïdes al voltant de NumPy.
- Combinat amb (I)Python i matplotlib es converteix en una ferrament molt potent per a manipular i visualitzar dades
- Aquesta combinació rivalitza en funcionalitat amb Matlab, IDL, Octave, R-Lab o SciLab.

Funcionalitat en SciPy

- Optimització
- Integració, càlcul de derivades
- Funcions especials (Kolmogorov, Beta, Gamma, Legendre, Chebyshev, Fresnel, etc...)
- Algorismes genètics
- Resolució d'equacions diferencials ordinàries
- Processament de senyals i imatges
- Gràfics (plotting), I/O
- ...

SciPy i la seua interacció amb NumPy

- Totes les funcions de NumPy (similars a les de numarray), s'han inclòs a SciPy
  --> no cal fer '`import Numeric`'
- S'han alterat les ufuncs per a no donar excepcions quan es troben errors amb nombres reals (es tornen NaN i Inf's als resultats) ==> nous operadors: `isnan, isfinite, isinf`
- Els operadors de comparació suporten nombres complexos (comparen la part real)

Trucs inicials (I)

- Conversió de tipus (*casting*):

```
In [1]: from scipy import *
In [2]: b=array([1,2])
In [3]: b.typecode()
Out[3]: 'l'
In [4]: a = cast['f'](b)
In [5]: a.typecode()
Out[5]: 'f'
```

Matrius

```
In [36]: pi
Out[36]: 3.1415926535897931
In [37]: cast['f'](pi)
Out[37]: 3.14159274101
In [38]: isscalar(cast['f'](pi))
Out[38]: True
```

Escalars

Trucs inicials (II)

- Creació d'arrays amb diferents notacions

```
In [54]: r_[3, [0]*5, -1:0.5:4j]     # Noteu la 'j'
Out[54]: array([ 3. ,  0. ,  0. ,  0. ,  0. ,  0. , -1. , -0.5,  0. ,  0.5])
In [55]: concatenate(([3], [0]*5, arange(-1, .502, .5)))
Out[55]: array([ 3. ,  0. ,  0. ,  0. ,  0. ,  0. , -1. , -0.5,  0. ,  0.5])
```

Cas unidimensional

```
In [77]: r_[[[3,3]]*2,[[2,2]]*2]
Out[77]:
array([[3, 3],
       [3, 3],
       [2, 2],
       [2, 2]])
In [78]: c_[[[3,3]]*2,[[2,2]]*2]
Out[78]:
array([[3, 3, 2, 2],
       [3, 3, 2, 2]])
```

Cas multidimensional

Algunes funcions útils

```
>>> a=r_[1:10:100j]
>>> b=arange(10)
>>> c=reshape(arange(12, typecode = 'd'), (4,3))
>>> who
<function who at 0xb727c994>
>>> who()
Name            Shape           Bytes           Type
===========================================================

a               100             800             double
b               10              40              long integer
c               4 x 3           96              double

Upper bound on total bytes  =       936
```

- Proveu també (`ipython -pylib`):

```
factorial, comb, rand i randn
factorial(200, exact=1)
hist(randn(10000),100)
```

Vectorize()

```
In [15]: def sumairesta(a,b):
   ....:     if a > b: return a - b
   ....:     else: return a + b
   ....:
In [16]: vec_sumairesta = vectorize(sumairesta)
In [20]: vec_sumairesta([1,2,3], 2)
Out[20]: array([3, 4, 1])
In [22]: vec_sumairesta([1,2,3], [0,1,4])
Out[22]: array([1, 1, 7])
```

- Useu vectorize() per a convertir funcions Python que accepten i retornen paràmetres escalars per a “vectoritzar-les”

Derivades

```
In [41]: a = r_[0:pi:5j]
In [42]: a
Out[42]: array([ 0.        ,  0.78539816,  1.57079633,  2.35619449,
3.14159265])
In [43]: sin(a)
Out[43]:
array([  0.00000000e+00,   7.07106781e-01,   1.00000000e+00,
7.07106781e-01,  1.22460635e-16])

In [44]: derivative(sin, a, 0.01, n=1)
Out[44]: array([ 0.99998333,  0.707095  ,  0.        , -0.707095  ,
-0.99998333])

In [45]: cos(a)
Out[45]:
array([1.00000000e+00, 7.07106781e-01, 6.12303177e-17, -7.07106781e-01,
-1.00000000e+00])

In [46]: derivative(sin, a, 0.01, n=2)
Out[46]: array([ 0. , -0.70710089, -0.99999167, -0.70710089, 0.   ])
```

Exercici

- Feu una funció que, donat un array 'x' com a paràmetre d'entrada, dóne com a eixida:
  `sin(x[i]) – cos(x[i]) si sin(x[i]) > cos(x[i])`
  `sin(x[i]) + cos(x[i]) si sin(x[i]) <= cos(x[i])`
- Després, calculeu la seua derivada primera i segona en el rang `r_[0:4*pi:100j]`
- Representeu la funció inicial, la primera i la segona derivada en una sola gràfica (useu matplotlib)
- Pista: useu vectorize()

Integrals numèriques (I)

$$I = \int_0^{4.5} J_{2.5}(x)\, dx$$

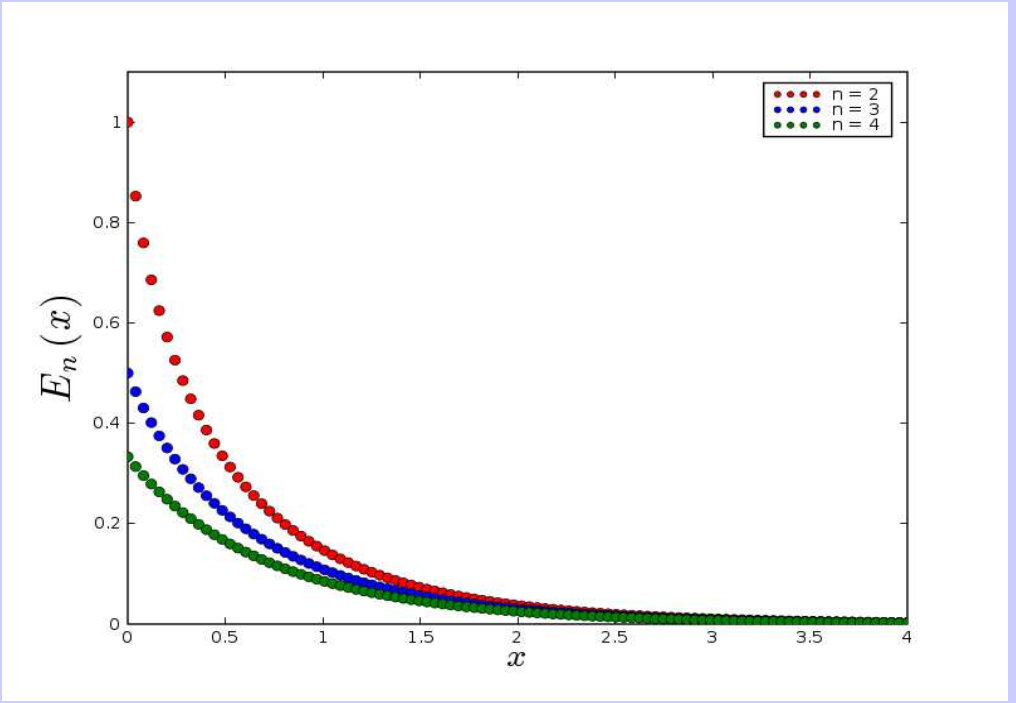
```
In [10]: integrate.quad(lambda x: special.jv(2.5,x), 0, 4.5)
Out[10]: (1.1781793807832 49, 7.866317182537 2322e-09)
```

Valor de l'integral

Valor absolut de l'error

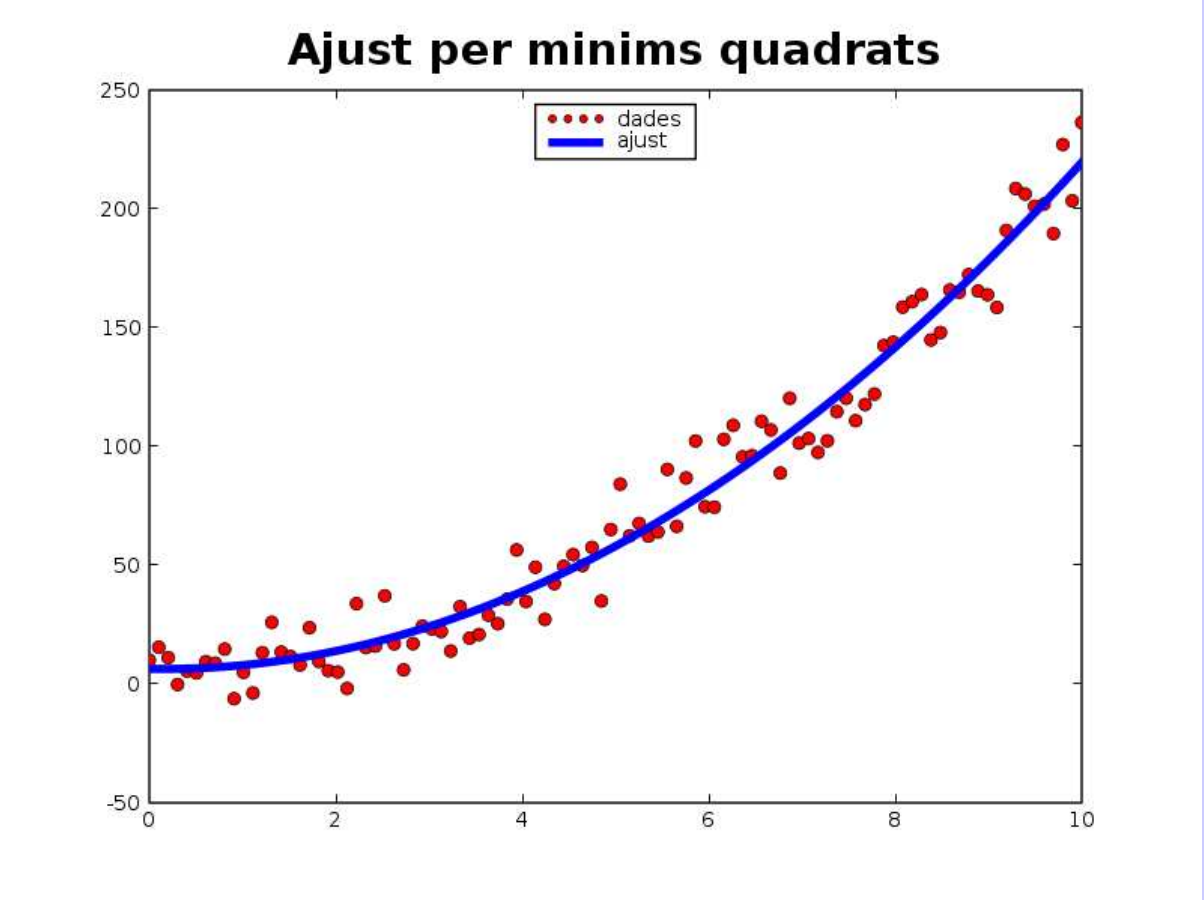
Integrals numèriques (II)

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n}\, dt$$




```
In [45]: def integrand(t,n,x):
   ....:     return exp(-x*t) / t**n
   ....:
In [46]: def expint(n,x):
   ....:     return integrate.quad(integrand, 1, inf, args=(n, x))[0]
   ....:
In [47]: vec_expint = vectorize(expint)
In [48]: a = r_[0:4.0:100j]
In [49]: plot(a, vec_expint(2, a),'ro', label=”n=2”)
In [50]: plot(a, vec_expint(3, a),'bo', label=”n=3”)
In [51]: plot(a, vec_expint(4, a),'go', label=”n=4”)
In [52]: legend()
```

Optimització (minimització)




```
xdata = r_[0:10:100j]
ydata = 3 + 1.5*xdata + 2*xdata*xdata + 10*randn(len(xdata))
# construeix una matriu per ajustar y = a + b*x + c*x^2
matrix=transpose(array([[1]*len(xdata), xdata, xdata*xdata]))
coeffs = linalg.basic.lstsq(matrix, ydata)[0]
```

I/O en SciPy

- Diferents mètodes de lectura/escriptura:
  - Fitxers ASCII (read_array, write_array)
  - Fitxers binaris (fopen)
    - Fitxers Fortran
    - Conversió de big/little endian
  - loadmat: llig un fitxer MATLAB (v4 o v5)
  - savemat: guarda com un fitxer MATLAB (<= 4)
  - Suport per a format Matrix Market (http://math.nist.gov/MatrixMarket/)

- En general, el suport de I/O és poc flexible i no permet estructurar les dades de manera eficient. Si es vol més flexibilitat i potència ==> PyTables

PyTables (i família)
http://www.pytables.org

- Fins ara hem vist un suport limitat per a entrada/eixida, tant en SciPy com en numarray
- Molts científics/enginyers tenen necessitat de manipular grans quantitats d'informació de manera ràpida i eficient
- Existeixen molts formats per a guardar informació, però un que últimament està adquirint gran rellevància és HDF5
- PyTables és una interfície en Python per a HDF5 però també moltes coses més

Què és PyTables?

- Es tracta d'una llibreria Python per a facilitar el manegament de grans quantitats de dades en el disc dur
- Està pensada tant per a un ús interactiu com en programes d'anàlisi
- Llicència BSD ==> Software obert i gratuït
- Un parell de “germans” en desenvolupament:
  - ViTables: Interfície gràfica basada en PyTables
  - CSTables: Versió Client-Servidor

PyTables: Trets bàsics

- **Facilitat d'ús**: Adopció de l'anomenat natural i la bona interacció amb Python
- **Ús eficient de la memòria**: 1 byte de dades al disc es representat per (1+x, x<<1) byte quan es carrega en memòria
- **Rapidesa d'execució**: Les parts crítiques com ara la escriptura/lectura/selecció de dades estan fetes en C (usant Pyrex)
- **Bona documentació**: Disposa d'un manual amb tutorials, referència completa de la llibreria, una secció d'optimització, ...

Altres característiques interessants

- Base de dades orientada a grans quantitats d'informació
- Suporta de manera nativa objectes NumPy i numarray
- Permet classificar d'una manera jeràrquica els diferents objectes
- Suporta compressió on-line de les dades
- Totes les cel·les als contenidors de dades poden allotjar matrius **multidimensionals**

Què hi ha al darrere de PyTables?

- Python ==> Interactivitat i flexibilitat
- numarray ==> Manipulació efectiva de dades homogènies i heterogènies
- HDF5 ==> Format de fitxer de dades i llibreria (C, Fortran i Java) dissenyada per a manipular grans quantitats d'informació i estructurar-la de manera jeràrquica
- Pyrex ==> Permet l'elaboració de extensions en C per a Python d'una manera molt senzilla i eficient

Quin software suporta fitxers PyTables (HDF5)?

- Els fitxers PyTables, a l'estar basats en el format HDF5, estan suportats en una gran varietat de software usat en càlcul científic
- Software obert
  - OpenDX (Open Data Explorer)
  - Gnu Octave
  - R, un sistema de càlcul estadístic i gràfics
  - Visad, toolkit Java per a visualització i anàlisi de dades numèriques
  - Zori, programa de càlcul de propietats atòmiques i moleculars basat en tècniques de Montecarlo
- Programes comercials
  - MATLAB, IDL, Intel Array-Visualizer, TecPlot, LabView
- Molts més (http://hdf.ncsa.uiuc.edu/tools5.html)

On pot ser útil PyTables?

- En tots aquells on cal la manipulació de grans quantitats d'informació
  - Aplicacions científiques
    - Meteorologia, Oceanografia
    - Astrofísica
    - Simulació numèrica en general
    - Medicina (sensors biològics)
  - Aplicacions industrials
    - Adquisició de dades de sensors
    - Monitorització en temps real
  - Adquisició de dades en sistemes informàtics
    - Traces de dades de routers
    - Monitorització de sistemes de logs
    - Alertes de seguretat (Firewalls, IDS, ...)

Un primer exemple

~alted/taller/pt1.py

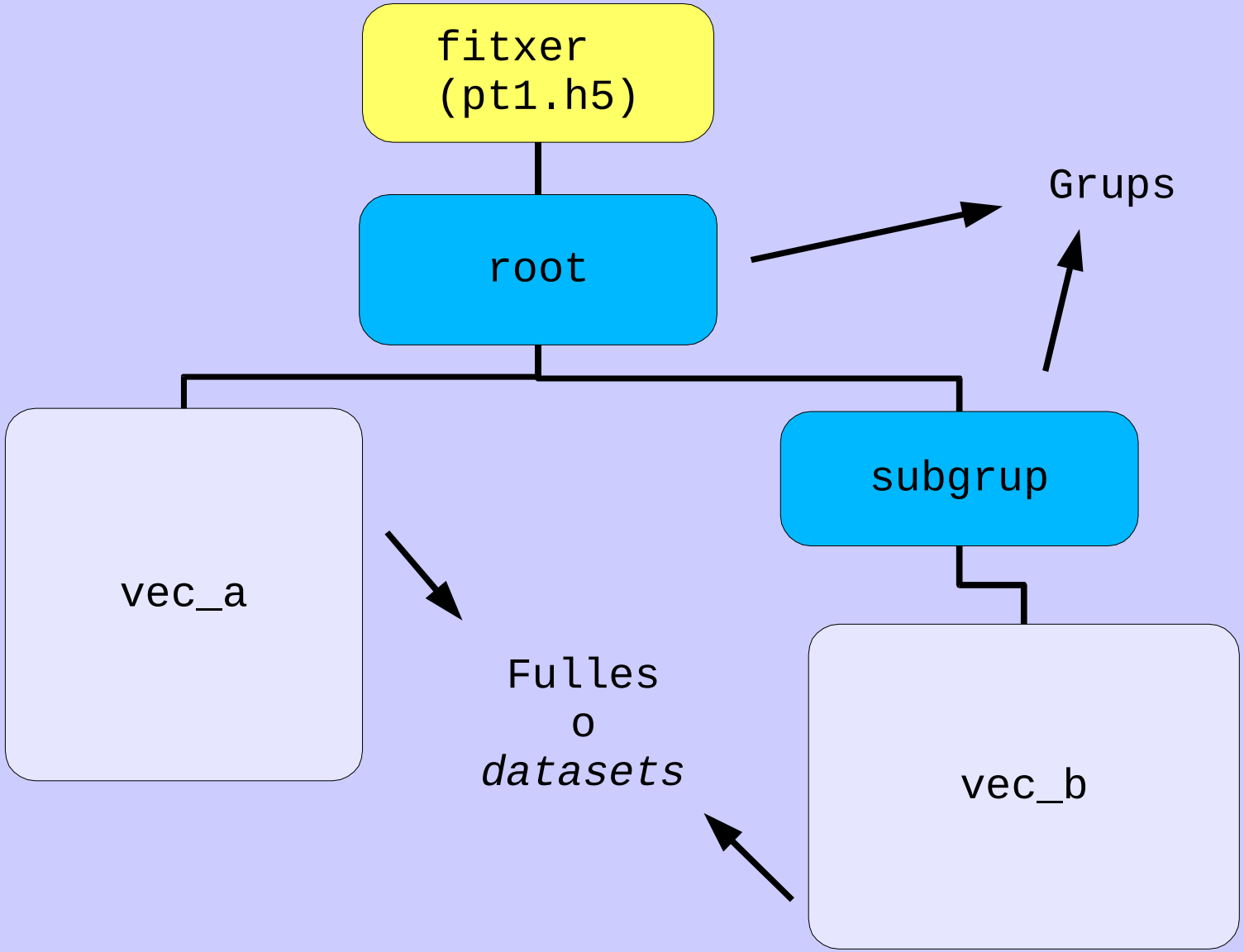
```
from scipy import *
import tables

fitxer = tables.openFile("pt1.h5", "w")
a=arange(10.)
b=arange(10.)*2
fitxer.createArray(fitxer.root, 'vec_a', a, "Vector a")
subgrup = fitxer.createGroup(fitxer.root, 'subgrup', "SG1")
fitxer.createArray(subgrup, 'vec_b', b, "Vector b")
fitxer.close()
```

I ara, proveu des de la shell el següent:

```
$ ptdump pt1.h5
$ ptdump -d pt1.h5
$ ptdump -v pt1.h5
$ ptdump -va pt1.h5
```

Estructura de fitxers PyTables: L'arbre d'objectes



Mètodes bàsics

- Obrir o crear un fitxer:

```
f=openFile(filename, mode='r', title='', trMap={},
           rootUEP="/", filters=None)
```

- Crear un grup:

```
f.createGroup(where, name, title='', filters=None)
```

- Crear una fulla (o 'dataset'):

```
f.createArray(where, name, object, title='')
f.createEArray(where, name, atom, title='', filters=None,
               expectedrows=1000)
f.createVLArray(where, name, atom=None, title='',
                filters=None, expectedsizeinMB=1.0)
```

```
f.createTable(where, name, description, title='',
              filters=None, expectedrows=10000)
```

Un exemple més complet

- Copieu el fitxer `~alted/taller/pt2.py` i executeu-lo (`./pt2.py`)
- Visualitzeu el contingut de l'eixida:
  `$ ptdump [-v] [-d] [-av] pt2.h5`
- Entreu a ipython i navegueu pel seu contingut (feu ús de la tecla TAB):

```
In [1]: from tables import *
In [2]: f=openFile("pt2.h5")
In [3]: f.root
In [4]: f.root.taules  # mostra info de grup
In [5]: f.root.taules.lectura  # mostra propietats taula
In [6]: f.root.taules.lectura.attrs  # Mostra atributs
In [7]: f.root.taules.lectura.cols  # mostra columnes
In [8]: f.root.taules.lectura.cols.compteADC # info columna
In [9]: f.root.taules.lectura.cols.compteADC[:] # dades!
```

Classes bàsiques: File

- És la que conté els mètodes per a crear, copiar, renomenar i esborrar els diferents nodes de l'arbre d'objectes:
  `createGroup, createArray, createEArray, createVLArray, createTable, removeNode, copyFile, copyChildren, removeNode, renameNode, flush, close`
- També suporta mètodes per a fer la navegació més fàcil:
  `listNodes, walkGroups, walkNodes`
- Mètodes especials:
  `__iter__(), __repr__(), __str__()`

Classes bàsiques: Group

- És l'objecte bàsic que permet dotar d'estructura a un fitxer
- Permet copiar, renomenar i esborrar els diferents nodes fills que allotja:
  `_f_remove, _f_rename, _f_copyChildren`
- Mètodes per a navegació:
  `_f_listNodes, _f_walkGroups, _f_walkNodes`
- Assignació d'atributs:
  `_f_getAttr, _f_setAttr`
- Mètodes especials:
  `__iter__(), __repr__(), __str__()`

Exemple de navegació

http://pytables.sourceforge.net/html/tut/tutorial1-2.html

```
# List all the nodes (Group and Leaf objects) on tree
print h5file   # ús de __str__()

# List all the nodes (using File iterator) on tree
print "Nodes in file:"
for node in h5file:  # ús de __iter__()
    print node

# Now, only list all the groups on tree
print "Groups in file:"
for group in h5file.walkNodes(classname="Group"):
    print group   # ús de __str__()

# List only the arrays hanging from /
print "Arrays in file (I):"
for group in h5file.walkGroups("/"):
    for array in h5file.listNodes(group, classname = 'Array'):
        print array   # ús de __str__()
```

Classe *Array*

- Pensada per albergar objectes NumPy i numarray de manera molt senzilla
- Atributs més importants:
  `flavor, nrows, nrow, type, itemsize`
- Mètodes de lectura:
  `iterrows(start=None, stop=None, step=1)`
  Exemple:
  ```
  result = [ row for row in array.iterrows(step=4) ]
  ```
  `read(start=None, stop=None, step=1)`
  Exemple:
  ```
  result = array.read(start=2, stop=40)
  ```
- Mètodes especials:
  `__iter__(), __getitem__()`
  Exemple:
  ```
  array4 = array[1, ..., ::2, 1:4, 4:]
  ```

Exercici: Repositori d'imatges

- Copieu el fitxer `~alted/taller/imatges.h5`, obriu-lo i navegueu per ell:

```
$ ipython -pylab
import tables
f=tables.openFile("imatges.h5")
imshow(f.root.venus[:])  # vella amiga
print f.root.microsoft
f.root.microsoft.attrs
f.root.microsoft.attrs.observ
imshow(f.root.microsoft[:]) # Hola!
```

- Mireu el contingut del grup '`forges`' :-)
- Afegiu al fitxer HDF5 la imatge:
  `~alted/taller/volca.png`

Classe declarativa: Atom

- S'usa per a definir propietats de l'element base en EArrays i VLArrays, com ara el tipus, el *shape*, o el “sabor”:

```
Atom(dtype="Float64", shape=1, flavor="NumArray")
```

```
StringAtom(shape=1, length=None, flavor="CharArray")
BoolAtom(shape=1, flavor="NumArray")
IntAtom(shape=1, itemsize=4, sign=1, flavor="NumArray")
FloatAtom(shape=1, itemsize=8, flavor="NumArray")
ComplexAtom(shape=1, itemsize=16, flavor="NumArray")
```

```
fileh.createVLArray(fileh.root, 'vlarray1',
                    Int32Atom(1, flavor="Numeric"),
                    "ragged array of ints",
                    filters = Filters(1))
```

Classe *EArray*

- És filla de **Array** i per tant, hereta les seues propietats
- A l'igual que **Array**, conté arrays de nombre homogenis, però al contrari que **Array**, **EArray** es pot *expandir* (encara que en una sola dimensió) i suporta *filtres*.
- Atributs més importants:
  `atom, extdim, nrows`
- Mètodes:

`append(object)`
Exemple:

```
atom = tables.StringAtom(shape=(0,), length=8)
array_c = fileh.createEArray(fileh.root, 'array_c', atom)
array_c.append(strings.array(['a'*2, 'b'*4], itemsize=8))
```

Classe VLArray

- Té files formades per objectes homogenis (els *àtoms*), però contenint un nombre *variable* d'aquests.
- Atributs:
  `atom, nrow, nrows`
- Mètodes:
  `append, iterrows, read`

```
atom = tables.Int32Atom(flavor="Numeric",
                        "ragged array of ints", Filters(complevel=1))
vlarray = fileh.createVLArray(fileh.root, 'vlarray1', atom)
# Append some (variable length) rows
vlarray.append(array([5, 6]))
vlarray.append(array([5, 6, 7]))
```

Classe declarativa: Col i IsDescription

- S'usen per a definir propietats bàsiques de les columnes en objectes **Table**, com ara el tipus, el *shape*, o la indexació:

```
Col(dtype="Float64", shape=1, dflt=None, pos=None, indexed=0)
```

```
StringCol(length=None, dflt=None, shape=1, pos=None, indexed=0)
BoolCol(dflt=0, shape=1, pos=None, indexed=0)
IntCol(dflt=0, shape=1, itemsize=4, sign=1, pos=None,indexed=0)
FloatCol(dflt=0.0, shape=1, itemsize=8, pos=None, indexed=0)
ComplexCol(dflt=0.+0.j, shape=1, itemsize=16, pos=None)
```

```
class MD(IsDescription):
    string16_1D  = StringCol(20,shape=(2,))
    uint16_2D    = UInt16Col(shape=(3,4))
    float_64_3D  = Float64Col(shape=(2,3,4))
    complex32_1D = Complex32Col(shape=(3,))
```

Classe Table

- Es fa servir per a guardar dades *inhomogènies*
- Atributs més importants:
  `description, row, nrows, rowsize, cols, colnames, coltypes, colshapes, indexed`
- Mètodes:
  `append, iterrows, itersequence, read, modifyRows, modifyColumns, removeRows, removeIndex, where, getWhereList`
- Mètodes especials:
  `__iter__, __getitem__, __setitem__`

Classe d'ajuda: Filters

```
Filters(complevel=0, complib="zlib", shuffle=1, fletcher32=0)
```

- Utilitzada per a definir filtres aplicats a fulles (excepte la classe **Array**), grups, sub-jerarquies o fitxers complets
- Paràmetres:
  - *complevel*: nivell de compressió
  - *complib*: compressor (“zlib”, “lzo”, “ucl”)
  - *shuffle*: si s'usa el pre-filtre “*shuffle*” o no
  - *fletcher32*: si s'usa un checksum o no
- Què fa el pre-filtre “*shuffle*”?
  Reordena els bytes de enters o nombres reals de manera que optimitza la compressió del conjunt de dades

Efecte dels diferents filtres de compressió

Table 5.2: Comparison between different compression libraries, with and without shuffling. The tests have been conducted on a Pentium 4 at 2 GHz and a hard disk at 4200 RPM.

Compr. Lib	File size (MB)	Time writing (s)	Time reading (s)	Speed writing (MB/s)	Speed reading (MB/s)
NO COMPR	165.4	24.5	17.13	6.6	9.6
Zlib (lvl 1)	26.4	22.2	5.77	7.3	28.4
Zlib+shuffle	4.0	19.0	5.94	8.6	27.6
LZO (lvl 1)	44.9	17.8	4.13	9.2	39.7
LZO+shuffle	4.3	16.4	5.03	9.9	32.6
UCL (lvl 1)	27.4	48.8	5.02	3.3	32.7
UCL+shuffle	3.5	38.1	5.31	4.3	30.9

- En general, s'usa 'lzo' quan es vol gran velocitat de descompressió i 'zlib' per a una relació òptima nivell de compressió/velocitat de descompressió
- 'shuffle' pot millorar molt el nivell de compressió a costa d'una lleugera pèrdua de velocitat en lectura

Classe AttributeSet: afegint meta-informació

- Facilita afegir atributs d'usuari als diferents objectes de l'arbre
- S'accedeix a través de la variable de classe `attr` per a les fulles i `_v_attr` per als grups
- Molt senzill d'usar:

```
In [2]: f=openFile("pt2.h5", "a")
In [3]: lectura = f.root.taules.lectura
In [4]: lectura.attrs.data = "2004-10-25 18:05 CEST"
In [5]: lectura.attrs.observ = "Feia molta calor"
In [6]: lectura.attrs.npersones = 2
In [7]: lectura.attrs.altres = [“sds”, 1.2, 4.4]
In [8]: lectura.attrs.dicc = {“lloc”:[12,34], “mes”:”prou”}
```

Selecció d'informació en taules

- Donada la definició de taula següent:

```
Menuda = {
  "var1" : Int8(),
  "var2" : IntCol(),
  "var3" : FloatCol(),
  }
```

Què penses que tornaran les següents expressions?:

1)

```
[row['var1'] for row in table if 3 < row['var2'] <= 20])
```

2)

```
sum([row['var1'] for row in table if 3<row['var2']<= 20])
```

3)

```
sum(row['var1'] for row in table if 3<row['var2']<= 20)
```

Pista: 3) usa expressions generadores (Python 2.4)

Indexació: Accelerant la búsqueda

- Adoneu-vos del paràmetre indexed a la següent definició:

```
Menuda = {
  "var1" : Int8Col(),
  "var2" : IntCol(indexed=1),
  "var3" : FloatCol(),
  }
```

Ara podem usar l'indexació sobre la columna “var2”:

1) Ús d'indexació sobre columna “var2”

```
[row['var1'] for row in table.where(if 3<row['var2']<= 20)]
```

2) “where” també es pot usar sobre columnes no indexades per a més velocitat ==> seleccions in-kernel

```
[row['var1'] for row in table.where(if 3<row['var3']<= 20)]
```

3) Tan sols es pot passar una condició a where ==> Seleccions mixtes

```
[row['var1'] for row in table.where(3<table.cols.var2<=20)
             if row['var3'] >= 2. and row['var1'] > 4]
```

Nota: Passeu sempre la condició més restrictiva a where

Exercici: Diferents seleccions

- Proveu a fer diferents seleccions sobre la taula: `pt2.h5:/taules/lectura`
- Useu les seleccions estàndard i in-kernel
  - Estàndard

```
lectura = f.root.taules.lectura
row = lectura.row
[row['compteADC'] for row in lectura if row['pressio'] < 20]
```

  - In-kernel

```
pressio = lectura.cols.pressio
[row['compteADC'] for row in lectura.where(pressio < 20)]
```

Exercici: Creació d'indexos

- Indexeu les columns 'pressio' i 'seqid' a la taula: `pt2.h5:/taules/lectura`

Pista: Useu el mètode `createIndex()` de la classe Cols :

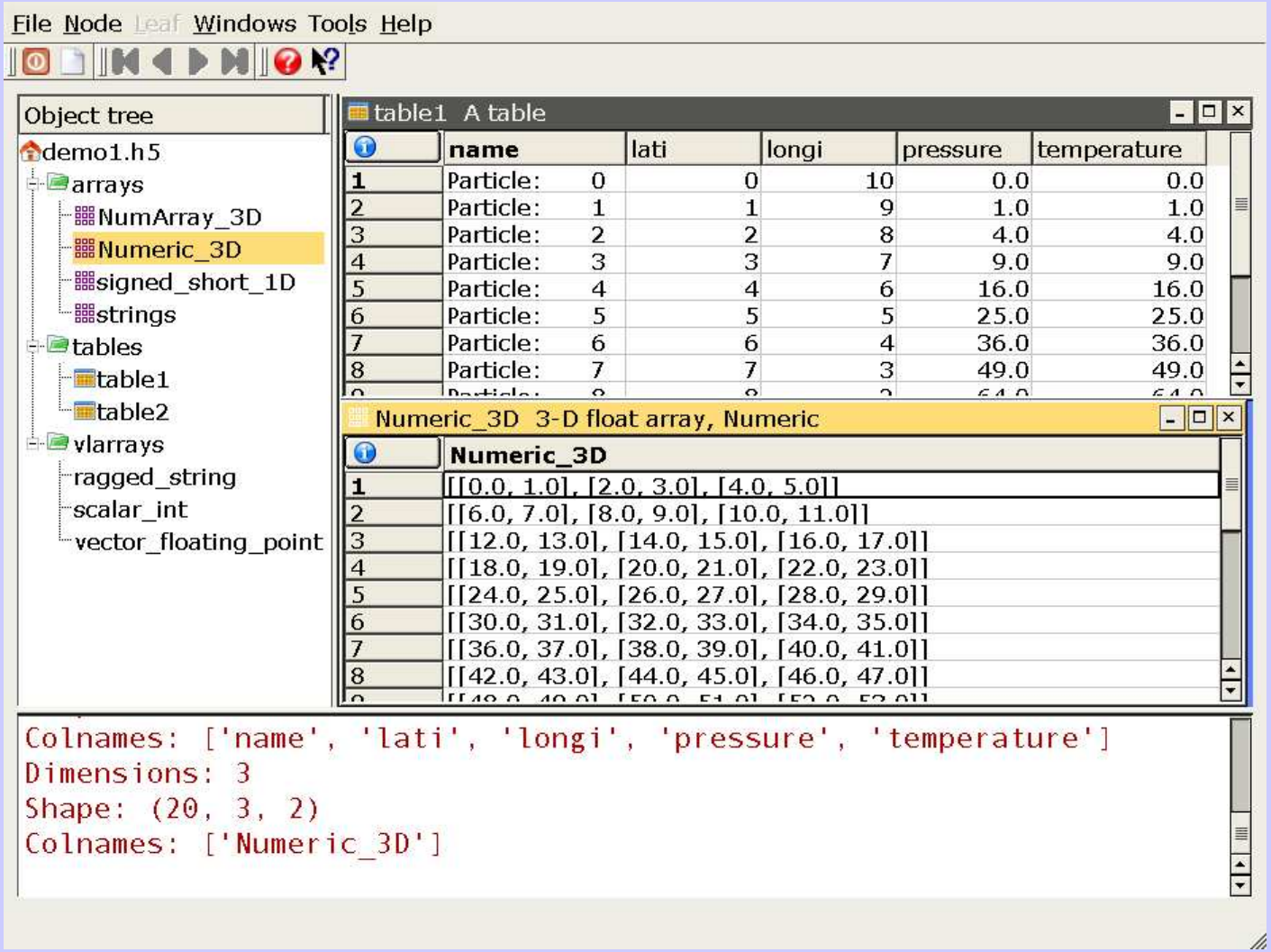
```
lectura.cols.pressio.createIndex()
```

Exercici: seleccionant dades d'una taula enorme (10^9 files)

- Des de ipython, obriu el fitxer:
  `/scratch1/alted/data/test1G.h5`
- Proveu a fer unes lectures i seleccions:

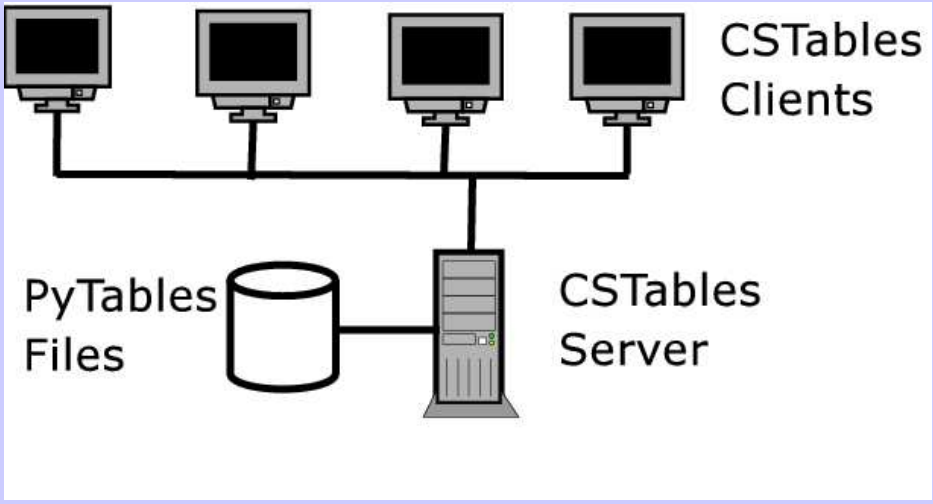
```
fcl2=f.root.table.cols.FCl2[:]
fcl2=f.root.table.cols.FCl2[300:350]
taula = f.root.table
fila = f.root.table.row
IC = taula.cols.IntCol
# Cerca per primera vegada
[fila['FCl2'] for fila in taula.where(10<IC<30)]
# Cerca per segona vegada (índex a la caché)
[fila['FCl2'] for fila in taula.where(20<IC<50)]
# Búsqueda sense indexació
[fila['FCl2'] for fila in taula if 20<IntCol<50]
```

ViTables: Una interfície gràfica per a PyTables



CSTables: PyTables esdevé Client-Servidor

- CSTables és un software addicional que permet fer aplicacions que guarden/recuperen les seues dades emmagatzemades en un servidor des d'un client a través de TCP/IP



Per acabar...

- Existeix **molt de software lliure** a la Xarxa per tal de facilitar l'adquisició i anàlisi de les dades que obté el científic/enginyer
- El propòsit d'aquest curs ha estat introduir-vos ***algunes*** d'aquestes ferramentes per que les pugueu afegir al vostre repositori
- El software lliure **no és**, de cap manera, un competidor del software comercial. Un ús equilibrat de la funcionalitat proveïda pels dos mons és, normalment, la millor opció

Preguntes? Sugerències?

- Coses que se'n vagen un poc del temari?