4. numpy

May 5, 2014

1 La biblioteca numpy: "numerical Python" (Python numérico)

```
In [1]: import numpy
```

Numpy provee *vectores* (en el sentido matemático)
Recordemos que al sumar dos listas, no hace lo que queremos que haga:

```
In [2]: l = [1, 2, 3]
        l + l

Out[2]: [1, 2, 3, 1, 2, 3]

In [3]: numpy.array  # es una funcion que acepta una lista

Out[3]: <function numpy.core.multiarray.array>

In [6]: v = numpy.array(l)
```

Acepta una lista y regresa un `array` de `numpy`

```
In [7]: v + v

Out[7]: array([2, 4, 6])
```

Ahora sí se comporta como un vector!

```
In [9]: 3.5 * v  # multiplica cada elemento

Out[9]: array([  3.5,   7. ,  10.5])

In [10]: numpy.array( [2., 1.] )

Out[10]: array([ 2.,  1.])

In [11]: dot(v, v)


    ---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

    <ipython-input-11-093f352c32e0> in <module>()
----> 1 dot(v, v)


    NameError: name 'dot' is not defined
```

```
In [12]: %who

a        l        numpy        v

In [13]: numpy.dot(v, v)

Out[13]: 14

In [14]: numpy.dot

Out[14]: <function numpy.core.multiarray.dot>

In [15]: v

Out[15]: array([1, 2, 3])

In [16]: type(v)

Out[16]: numpy.ndarray
```

Matrices:

```
In [18]: M = numpy.array([ [2., 1.], [1., 1.] ])

In [19]: M

Out[19]: array([[ 2.,  1.],
                [ 1.,  1.]])

In [20]: print(M)

[[ 2.  1.]
 [ 1.  1.]]

In [21]: M + M

Out[21]: array([[ 4.,  2.],
                [ 2.,  2.]])

In [23]: M * M  # atención: componente por componente; NO es multiplicacion matricial normal

Out[23]: array([[ 4.,  1.],
                [ 1.,  1.]])

In [25]: numpy.dot(M, M)

Out[25]: array([[ 5.,  3.],
                [ 3.,  2.]])
```

$(AB)_{ij} = \sum_k A_{ik} B_{kj}$
Es una contracción de índices
Normalmente:

```
In [26]: import numpy as np

In [27]: np.array([4, 5, 5])

Out[27]: array([4, 5, 5])
```

No usemos `from numpy import *` – importa ~500 funciones

`numpy` tiene cualquier operación sobre vectores y matrices (y su equivalente en más dimensiones – tensores de más alto rango)

Unos ejemplos:

```
In [28]: np.zeros(5)

Out[28]: array([ 0.,  0.,  0.,  0.,  0.])

In [29]: np.zeros( (10, 10) )  # (10, 10) es un par ordenado (tupla)

Out[29]: array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])

In [30]: (10, 10)

Out[30]: (10, 10)

In [31]: np.ones(10)

Out[31]: array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])

In [33]: 2. * np.ones(10)

Out[33]: array([ 2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.])

In [34]: np.identity(5)

Out[34]: array([[ 1.,  0.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.,  0.],
                [ 0.,  0.,  1.,  0.,  0.],
                [ 0.,  0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  0.,  1.]])

In [35]: np.diag([1, 2, 3])

Out[35]: array([[1, 0, 0],
                [0, 2, 0],
                [0, 0, 3]])
```

1.1 Cómo extraer elementos de una matriz:

```
In [40]: M = np.diag(range(5))
         M

Out[40]: array([[0, 0, 0, 0, 0],
                [0, 1, 0, 0, 0],
                [0, 0, 2, 0, 0],
                [0, 0, 0, 3, 0],
                [0, 0, 0, 0, 4]])
```

```
In [41]: M = np.array([1,2,3,4])

In [42]: M

Out[42]: array([1, 2, 3, 4])

In [43]: M.shape

Out[43]: (4,)

In [45]: M = np.array([1,2,3,4]).reshape(2, 2)
         M

Out[45]: array([[1, 2],
                [3, 4]])

In [47]: M[0]  # primer renglon

Out[47]: array([1, 2])

In [48]: M[0][1]

Out[48]: 2

In [49]: M[0, 1]

Out[49]: 2

In [54]: M[0, 0:2]

Out[54]: array([1, 2])

In [55]: M[0, 0:-1]

Out[55]: array([1])

In [56]: M[0, :]  # toma por defecto el inicio y fin del vector

Out[56]: array([1, 2])

In [58]: M[:, 0]  # primera columna

Out[58]: array([1, 3])

In [60]: M.T  # da la transpuesta

Out[60]: array([[1, 3],
                [2, 4]])

In [61]: M.T[0]

Out[61]: array([1, 3])

In [62]: M[:,0]

Out[62]: array([1, 3])

In [65]: M = np.diag(range(5))

In [66]: M

Out[66]: array([[0, 0, 0, 0, 0],
                [0, 1, 0, 0, 0],
                [0, 0, 2, 0, 0],
                [0, 0, 0, 3, 0],
                [0, 0, 0, 0, 4]])

In [68]: M[1:4, 2:4]  # submatriz

Out[68]: array([[0, 0],
                [2, 0],
                [0, 3]])
```

1.2 Álgebra lineal

Hay un submódulo de `numpy` que se llama `linalg` que provee las funciones básicas de álgebra lineal:

```
In [69]: from numpy import linalg

In [70]: linalg

Out[70]: <module 'numpy.linalg' from '/usr/local/Cellar/python/2.7.5/Frameworks/Python.framework/Versio

In [71]: M

Out[71]: array([[0, 0, 0, 0, 0],
                [0, 1, 0, 0, 0],
                [0, 0, 2, 0, 0],
                [0, 0, 0, 3, 0],
                [0, 0, 0, 0, 4]])

In [72]: linalg.inv(M)


    ---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)

    <ipython-input-72-8f8888e69724> in <module>()
----> 1 linalg.inv(M)


    /usr/local/Cellar/python/2.7.5/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packa
    443     """
    444     a, wrap = _makearray(a)
--> 445     return wrap(solve(a, identity(a.shape[0], dtype=a.dtype)))
    446
    447


    /usr/local/Cellar/python/2.7.5/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packa
    326     results = lapack_routine(n_eq, n_rhs, a, n_eq, pivots, b, n_eq, 0)
    327     if results['info'] > 0:
--> 328         raise LinAlgError('Singular matrix')
    329     if one_eq:
    330         return wrap(b.ravel().astype(result_t))


    LinAlgError: Singular matrix


In [73]: linalg.det(M)

Out[73]: 0.0

In [74]: M = np.array([1, 2, 3, 4])

In [77]: N = M.reshape(2, 2)  # es una vista ("view") de la matriz original
         # NO es una copia; son los mismos datos

In [78]: N
```

```
Out[78]: array([[1, 2],
                [3, 4]])

In [79]: N[0, 1] = 100

In [80]: N

Out[80]: array([[  1, 100],
                [  3,   4]])

In [81]: M

Out[81]: array([  1, 100,   3,   4])

In [82]: N

Out[82]: array([[  1, 100],
                [  3,   4]])

In [83]: N[0, 1] = 2

In [83]:

In [84]: N

Out[84]: array([[1, 2],
                [3, 4]])

In [86]: linalg.det(N)

Out[86]: -2.0000000000000004

In [87]: linalg.inv(N)

Out[87]: array([[-2. ,  1. ],
                [ 1.5, -0.5]])

In [89]: linalg.inv??

In [91]: linalg.eig(N)

Out[91]: (array([-0.37228132,  5.37228132]), array([[-0.82456484, -0.41597356],
                 [ 0.56576746, -0.90937671]]))
```

Regresa una tupla (par ordenado) de dos elementos: e-vals y e-vecs

```
In [93]: lamb, v = linalg.eig(N)
         # tupla = tupla -- hace desempaque de tuplas

In [94]: lamb

Out[94]: array([-0.37228132,  5.37228132])

In [95]: v

Out[95]: array([[-0.82456484, -0.41597356],
                [ 0.56576746, -0.90937671]])
```

Ejercicio: Averigüa dónde están los eigenvectores
Tarea: Método de potencias para calcular un eigenvector y un eigenvalor

```
In [6]: import numpy as np
        from numpy import linalg

In [3]: M = np.array([2., 1., 1., 1.]).reshape(2, 2)

In [4]: c = np.array([1., 1.])
```

Quiero resolver el sistema $\mathsf{M} \cdot \mathbf{x} = \mathbf{c}$:

```
In [8]: x = linalg.solve(M, c)
        x

Out[8]: array([ 0.,  1.])

In [13]: linalg.solve?
```

1.3 Transformadas de Fourier

```
In [97]: from numpy import fft  # fast Fourier transform

In [99]: x = np.linspace(0, 100, 1024)

In [100]: x

Out[100]: array([  0.00000000e+00,   9.77517107e-02,   1.95503421e-01, ...,
                   9.98044966e+01,   9.99022483e+01,   1.00000000e+02])

In [101]: len(x)

Out[101]: 1024

In [102]: f = fft.fft(x)

In [103]: x2 = fft.ifft(f)

In [104]: x2

Out[104]: array([  3.55271368e-15 -2.49800181e-15j,
                   9.77517107e-02 -2.22044605e-16j,
                   1.95503421e-01 -5.01889652e-15j, ...,
                   9.98044966e+01 +3.88578059e-15j,
                   9.99022483e+01 +1.82027949e-14j,   1.00000000e+02 -1.17267307e-14j])

In [105]: x

Out[105]: array([  0.00000000e+00,   9.77517107e-02,   1.95503421e-01, ...,
                   9.98044966e+01,   9.99022483e+01,   1.00000000e+02])

In [107]: x2.real

Out[107]: array([  3.55271368e-15,   9.77517107e-02,   1.95503421e-01, ...,
                   9.98044966e+01,   9.99022483e+01,   1.00000000e+02])

In [108]: x2.real - x  # errores

Out[108]: array([  3.55271368e-15,   7.02216063e-15,  -1.66533454e-16, ...,
                   0.00000000e+00,   5.68434189e-14,  -4.26325641e-14])

In [109]: linalg.norm(x2.real - x)

Out[109]: 3.7388984958239193e-13
```

1.4 Números aleatorios

Hay un submódulo de `numpy` para producir números (seudo-)aleatorios.

```
In [1]: from numpy import random

In [2]: random.rand

Out[2]: <function rand>

In [26]: # numero aleatorio uniforme entre 0 y 1:
         random.rand()

Out[26]: 0.22360738532725755

In [27]: random.rand(3)

Out[27]: array([ 0.72638377,  0.22926055,  0.95029934])

In [29]: random.rand(3, 3)

Out[29]: array([[ 0.48644947,  0.76612385,  0.63281789],
                [ 0.46679283,  0.15790784,  0.30661562],
                [ 0.56408886,  0.90344024,  0.59620715]])

In [31]: random.randn(100)  # distribucion normal estandar

Out[31]: array([ 1.96923414, -0.2141449 ,  0.77744524,  0.81325598,  0.92252027,
                 0.1928308 ,  0.42165311,  0.45520047, -0.58336955, -0.85428152,
                 0.80092534, -1.2563949 ,  1.01468299,  1.11516912, -0.02865218,
                 0.18785725, -1.07156541,  1.18276602,  1.37330898,  0.48599099,
                -2.2630245 , -1.27926794,  0.07919993, -1.02950659, -0.6633516 ,
                -1.03719324,  0.16202782, -1.49051598,  0.6922923 ,  0.8307979 ,
                -0.74656157, -0.11054679,  0.30952972, -2.07747786,  0.57449484,
                 0.06757248,  0.59923819, -0.23171694,  1.86531943,  0.57038927,
                 1.02167588, -0.33153726,  0.26783818, -1.47657545,  1.83127475,
                -1.90195421,  0.88035848, -0.40454902, -2.10864041,  0.57904506,
                -1.13749481,  0.87165092, -0.4041457 , -0.53853878,  0.04560867,
                -0.7052234 ,  0.58162055,  0.59935702,  0.10278931, -0.26116051,
                 1.36011493, -1.00785712, -0.75239444,  2.03494972,  1.61191708,
                 0.80986506, -1.92907844,  0.30636403,  0.6661681 , -1.1178212 ,
                -0.25599829, -1.02935863, -0.53004217,  1.92130323, -0.24335478,
                 0.19597113,  0.66874136,  0.66831047, -0.50718818,  0.39543966,
                 0.33500129,  1.6545728 ,  0.68671895, -0.49427429, -2.24993033,
                -0.42802322,  0.79950429, -0.72465434,  0.12479739, -0.71969141,
                 1.21309528, -1.21261034, -2.25926955, -0.96484993, -0.62112468,
                -1.16547805,  0.77298085, -1.46450965,  0.44122903,  0.58649299])

In [33]: random?

In [34]: M = random.rand(1000, 1000)

In [35]: M = M + M.T  # produce matriz simetrica

In [36]: from numpy import linalg

In [37]: %time linalg.eigvalsh(M)
```

```
CPU times: user 440 ms, sys: 67.7 ms, total: 507 ms
Wall time: 139 ms

Out[37]: array([ -2.59571854e+01,  -2.57299787e+01,  -2.54592018e+01,
                 -2.49899836e+01,  -2.48942731e+01,  -2.46988124e+01,
                 -2.46797772e+01,  -2.44949018e+01,  -2.42906139e+01,
                 -2.41135869e+01,  -2.39871592e+01,  -2.38994431e+01,
                 -2.38540030e+01,  -2.37959517e+01,  -2.37655362e+01,
                 -2.36804965e+01,  -2.36255161e+01,  -2.33977906e+01,
                 -2.31621569e+01,  -2.30941569e+01,  -2.30088420e+01,
                 -2.29286029e+01,  -2.28504189e+01,  -2.28084933e+01,
                 -2.27789567e+01,  -2.25754548e+01,  -2.25250255e+01,
                 -2.24436788e+01,  -2.23909176e+01,  -2.22839870e+01,
                 -2.21472659e+01,  -2.21144888e+01,  -2.20553463e+01,
                 -2.19504286e+01,  -2.19323138e+01,  -2.17937363e+01,
                 -2.17110782e+01,  -2.16580700e+01,  -2.16423028e+01,
                 -2.15461161e+01,  -2.14595710e+01,  -2.13990163e+01,
                 -2.13684184e+01,  -2.12742759e+01,  -2.11878344e+01,
                 -2.11295253e+01,  -2.10653198e+01,  -2.10335375e+01,
                 -2.09480234e+01,  -2.09167383e+01,  -2.08469695e+01,
                 -2.07307874e+01,  -2.06416286e+01,  -2.05741497e+01,
                 -2.05250138e+01,  -2.04490390e+01,  -2.03405377e+01,
                 -2.02796617e+01,  -2.02460154e+01,  -2.01530022e+01,
                 -2.01084493e+01,  -2.00757188e+01,  -1.99006780e+01,
                 -1.98496158e+01,  -1.97561349e+01,  -1.96964970e+01,
                 -1.96656308e+01,  -1.95761000e+01,  -1.95206148e+01,
                 -1.94358784e+01,  -1.93951500e+01,  -1.93543071e+01,
                 -1.92968342e+01,  -1.92615048e+01,  -1.92245567e+01,
                 -1.92167195e+01,  -1.90951318e+01,  -1.90196994e+01,
                 -1.89254097e+01,  -1.88326114e+01,  -1.87988905e+01,
                 -1.87722206e+01,  -1.86947710e+01,  -1.86610355e+01,
                 -1.85928960e+01,  -1.85828244e+01,  -1.85276863e+01,
                 -1.84067701e+01,  -1.83712463e+01,  -1.83303537e+01,
                 -1.82568289e+01,  -1.81884963e+01,  -1.81501701e+01,
                 -1.80660149e+01,  -1.79789636e+01,  -1.79103198e+01,
                 -1.78857926e+01,  -1.78109120e+01,  -1.77559564e+01,
                 -1.76888325e+01,  -1.76338487e+01,  -1.75793250e+01,
                 -1.75627352e+01,  -1.74880395e+01,  -1.74488502e+01,
                 -1.74043985e+01,  -1.73412819e+01,  -1.72794186e+01,
                 -1.72509437e+01,  -1.71702516e+01,  -1.71248681e+01,
                 -1.70643767e+01,  -1.69984289e+01,  -1.69873667e+01,
                 -1.68437489e+01,  -1.68266454e+01,  -1.67952980e+01,
                 -1.67850035e+01,  -1.67276305e+01,  -1.66528077e+01,
                 -1.65818271e+01,  -1.65237633e+01,  -1.64813059e+01,
                 -1.64776294e+01,  -1.63796952e+01,  -1.63510813e+01,
                 -1.62763064e+01,  -1.62697821e+01,  -1.61704188e+01,
                 -1.61445620e+01,  -1.60800758e+01,  -1.60141007e+01,
                 -1.59689061e+01,  -1.59384472e+01,  -1.59059373e+01,
                 -1.58500212e+01,  -1.57663899e+01,  -1.57309559e+01,
                 -1.57168118e+01,  -1.56219017e+01,  -1.55764615e+01,
                 -1.54883244e+01,  -1.54562439e+01,  -1.54413983e+01,
                 -1.54353194e+01,  -1.53611900e+01,  -1.53412804e+01,
                 -1.52204081e+01,  -1.51463658e+01,  -1.51101475e+01,
                 -1.50884139e+01,  -1.50264508e+01,  -1.49679469e+01,
```

```
-1.49014151e+01,  -1.48584081e+01,  -1.47842729e+01,
-1.47582447e+01,  -1.47431217e+01,  -1.46750662e+01,
-1.46470457e+01,  -1.45725317e+01,  -1.45274933e+01,
-1.44884967e+01,  -1.44624752e+01,  -1.43877117e+01,
-1.43365492e+01,  -1.42899282e+01,  -1.42306994e+01,
-1.41688938e+01,  -1.41039756e+01,  -1.40703656e+01,
-1.40336978e+01,  -1.40236085e+01,  -1.39064143e+01,
-1.38711941e+01,  -1.38474939e+01,  -1.38086233e+01,
-1.37317268e+01,  -1.36988139e+01,  -1.36526632e+01,
-1.36020624e+01,  -1.35306068e+01,  -1.34786099e+01,
-1.34707666e+01,  -1.34149486e+01,  -1.33540104e+01,
-1.33181162e+01,  -1.32782382e+01,  -1.32381997e+01,
-1.31888894e+01,  -1.31010430e+01,  -1.30810553e+01,
-1.30304858e+01,  -1.29675313e+01,  -1.29308300e+01,
-1.29102870e+01,  -1.28277364e+01,  -1.27954067e+01,
-1.27410790e+01,  -1.27041295e+01,  -1.26661560e+01,
-1.26109586e+01,  -1.25771604e+01,  -1.25546716e+01,
-1.24854415e+01,  -1.24587167e+01,  -1.24292781e+01,
-1.23404477e+01,  -1.22985216e+01,  -1.22762812e+01,
-1.22290008e+01,  -1.22049958e+01,  -1.21431863e+01,
-1.20738053e+01,  -1.20530760e+01,  -1.20216106e+01,
-1.19693308e+01,  -1.19525696e+01,  -1.18276638e+01,
-1.17966541e+01,  -1.17598343e+01,  -1.17406869e+01,
-1.16831908e+01,  -1.16309480e+01,  -1.16007653e+01,
-1.15708534e+01,  -1.15044144e+01,  -1.14548983e+01,
-1.13914190e+01,  -1.13615504e+01,  -1.12999048e+01,
-1.12369793e+01,  -1.12035583e+01,  -1.11563185e+01,
-1.11265537e+01,  -1.10884291e+01,  -1.10422362e+01,
-1.09702386e+01,  -1.09280712e+01,  -1.09026715e+01,
-1.08276184e+01,  -1.08019426e+01,  -1.07435488e+01,
-1.07088804e+01,  -1.06833961e+01,  -1.06241329e+01,
-1.06231339e+01,  -1.05910200e+01,  -1.05019187e+01,
-1.04708317e+01,  -1.04498283e+01,  -1.03971356e+01,
-1.03452566e+01,  -1.02934208e+01,  -1.02829330e+01,
-1.02363012e+01,  -1.01570987e+01,  -1.01157648e+01,
-1.01038492e+01,  -1.00428702e+01,  -1.00224661e+01,
-9.99091428e+00,  -9.93606062e+00,  -9.84986632e+00,
-9.80590960e+00,  -9.75354728e+00,  -9.67215482e+00,
-9.62954216e+00,  -9.61138495e+00,  -9.52117657e+00,
-9.50637732e+00,  -9.48797123e+00,  -9.42814308e+00,
-9.41918762e+00,  -9.33197848e+00,  -9.28389233e+00,
-9.24798258e+00,  -9.19995713e+00,  -9.15391583e+00,
-9.12250409e+00,  -9.04918794e+00,  -9.00261348e+00,
-8.96144563e+00,  -8.91964258e+00,  -8.86767062e+00,
-8.79529232e+00,  -8.76887184e+00,  -8.71548932e+00,
-8.68441888e+00,  -8.66408639e+00,  -8.60194115e+00,
-8.55396889e+00,  -8.54527491e+00,  -8.50324819e+00,
-8.44781931e+00,  -8.39975777e+00,  -8.38520504e+00,
-8.36224647e+00,  -8.27616356e+00,  -8.24040854e+00,
-8.20062129e+00,  -8.16166923e+00,  -8.13807134e+00,
-8.08690840e+00,  -8.05668245e+00,  -8.02535123e+00,
-7.99509478e+00,  -7.95063882e+00,  -7.86410407e+00,
-7.81581285e+00,  -7.79994104e+00,  -7.75768802e+00,
-7.71728798e+00,  -7.67410989e+00,  -7.56868362e+00,
```

```
-7.55751288e+00,  -7.53530726e+00,  -7.47601252e+00,
-7.39970192e+00,  -7.36746057e+00,  -7.36146026e+00,
-7.30106447e+00,  -7.24730095e+00,  -7.23234473e+00,
-7.21119832e+00,  -7.20218540e+00,  -7.12466112e+00,
-7.04368553e+00,  -7.01695652e+00,  -6.97827461e+00,
-6.93814251e+00,  -6.92654590e+00,  -6.90070719e+00,
-6.86047485e+00,  -6.76470893e+00,  -6.68990354e+00,
-6.67195689e+00,  -6.60936256e+00,  -6.56868978e+00,
-6.53847275e+00,  -6.51674288e+00,  -6.48409982e+00,
-6.43932825e+00,  -6.41749550e+00,  -6.33694592e+00,
-6.29699510e+00,  -6.22203468e+00,  -6.21883905e+00,
-6.19497730e+00,  -6.13575936e+00,  -6.11172019e+00,
-6.06590051e+00,  -6.05178938e+00,  -5.97333997e+00,
-5.94472712e+00,  -5.87239088e+00,  -5.84697365e+00,
-5.80308806e+00,  -5.76629494e+00,  -5.73838231e+00,
-5.67313944e+00,  -5.61589025e+00,  -5.57766253e+00,
-5.54148082e+00,  -5.53081248e+00,  -5.48913487e+00,
-5.42502742e+00,  -5.38462400e+00,  -5.36159152e+00,
-5.32274226e+00,  -5.27404680e+00,  -5.21927038e+00,
-5.20897005e+00,  -5.14787593e+00,  -5.10255174e+00,
-5.08357479e+00,  -4.99966442e+00,  -4.98773297e+00,
-4.92354158e+00,  -4.90335964e+00,  -4.86794871e+00,
-4.84767760e+00,  -4.81401962e+00,  -4.76567402e+00,
-4.70108711e+00,  -4.67911255e+00,  -4.62884639e+00,
-4.58146949e+00,  -4.52455900e+00,  -4.49549489e+00,
-4.45349088e+00,  -4.42509127e+00,  -4.39064423e+00,
-4.34519024e+00,  -4.30156662e+00,  -4.28089794e+00,
-4.21016269e+00,  -4.19921134e+00,  -4.12720603e+00,
-4.10552700e+00,  -4.04690623e+00,  -4.00916481e+00,
-3.96505293e+00,  -3.91110200e+00,  -3.87142943e+00,
-3.84061083e+00,  -3.76595453e+00,  -3.74141667e+00,
-3.68939216e+00,  -3.60415148e+00,  -3.59776529e+00,
-3.55182192e+00,  -3.53202249e+00,  -3.50253856e+00,
-3.43374875e+00,  -3.42463029e+00,  -3.38251130e+00,
-3.30717559e+00,  -3.26825443e+00,  -3.21341107e+00,
-3.16168713e+00,  -3.10211072e+00,  -3.07592270e+00,
-3.04067915e+00,  -3.01278093e+00,  -2.97913078e+00,
-2.93490626e+00,  -2.89663876e+00,  -2.85315837e+00,
-2.83200603e+00,  -2.80084949e+00,  -2.71683487e+00,
-2.69026683e+00,  -2.66416840e+00,  -2.60032847e+00,
-2.54564307e+00,  -2.52435350e+00,  -2.50127729e+00,
-2.39713070e+00,  -2.38006005e+00,  -2.34461406e+00,
-2.30120654e+00,  -2.24314760e+00,  -2.19299066e+00,
-2.16101394e+00,  -2.13718696e+00,  -2.10708191e+00,
-2.08452297e+00,  -2.02181328e+00,  -1.98957655e+00,
-1.94153664e+00,  -1.91055085e+00,  -1.90177094e+00,
-1.86754842e+00,  -1.84073306e+00,  -1.78260681e+00,
-1.75104968e+00,  -1.65971321e+00,  -1.62249044e+00,
-1.58501974e+00,  -1.54702801e+00,  -1.49507368e+00,
-1.47781367e+00,  -1.45369122e+00,  -1.37535284e+00,
-1.34487636e+00,  -1.29331745e+00,  -1.24661047e+00,
-1.22477881e+00,  -1.14342475e+00,  -1.13806620e+00,
-1.10624632e+00,  -1.08089193e+00,  -1.04830365e+00,
-9.98989484e-01,  -9.45161969e-01,  -9.17987820e-01,
```

```
-8.60095472e-01,  -8.01355909e-01,  -7.95330938e-01,
-7.60153544e-01,  -7.31655773e-01,  -6.63854694e-01,
-6.19353874e-01,  -5.01394076e-01,  -4.71402501e-01,
-4.49038386e-01,  -3.88085300e-01,  -3.82107868e-01,
-3.67170886e-01,  -3.28664285e-01,  -3.00076248e-01,
-2.27336989e-01,  -1.77987045e-01,  -1.32798777e-01,
-9.96902445e-02,  -8.46472462e-02,  -4.64780055e-02,
-2.02319471e-04,   1.29400267e-02,   1.97352900e-02,
 9.73379249e-02,   1.40554657e-01,   1.59852712e-01,
 2.07969777e-01,   2.98464561e-01,   3.42276374e-01,
 3.72220337e-01,   4.24934887e-01,   4.50273491e-01,
 4.68897322e-01,   5.26150981e-01,   5.69375480e-01,
 6.18675533e-01,   6.72822777e-01,   7.06303026e-01,
 7.33013333e-01,   7.46063673e-01,   7.64240589e-01,
 8.52423621e-01,   8.85857953e-01,   9.28162847e-01,
 1.00571230e+00,   1.02123473e+00,   1.10251785e+00,
 1.11270215e+00,   1.14078393e+00,   1.21057420e+00,
 1.26870628e+00,   1.28877530e+00,   1.33080487e+00,
 1.34010953e+00,   1.39264931e+00,   1.41968125e+00,
 1.44810805e+00,   1.48937967e+00,   1.50135085e+00,
 1.56635536e+00,   1.61821201e+00,   1.69251127e+00,
 1.74086917e+00,   1.77136535e+00,   1.82182480e+00,
 1.83657321e+00,   1.87508915e+00,   1.90959938e+00,
 1.97368382e+00,   1.98651934e+00,   2.01261248e+00,
 2.06143705e+00,   2.06925163e+00,   2.10297848e+00,
 2.16939131e+00,   2.22420090e+00,   2.26066766e+00,
 2.28270026e+00,   2.34909163e+00,   2.36904415e+00,
 2.41730205e+00,   2.47134559e+00,   2.51552640e+00,
 2.55390111e+00,   2.59448711e+00,   2.63502494e+00,
 2.64457678e+00,   2.68727718e+00,   2.73845498e+00,
 2.84821510e+00,   2.87585120e+00,   2.88600751e+00,
 2.94849461e+00,   2.97068492e+00,   2.98253610e+00,
 3.00062186e+00,   3.10217897e+00,   3.15823904e+00,
 3.19646574e+00,   3.23628556e+00,   3.25834023e+00,
 3.26216962e+00,   3.32710676e+00,   3.37223782e+00,
 3.44219010e+00,   3.45824577e+00,   3.49780965e+00,
 3.52378029e+00,   3.57858904e+00,   3.62841758e+00,
 3.64909226e+00,   3.71261441e+00,   3.72691082e+00,
 3.74926353e+00,   3.84580203e+00,   3.85601772e+00,
 3.89309328e+00,   3.94784162e+00,   4.00272968e+00,
 4.04961730e+00,   4.07923654e+00,   4.10036089e+00,
 4.15831753e+00,   4.17470980e+00,   4.21001066e+00,
 4.25990805e+00,   4.31839377e+00,   4.39824800e+00,
 4.40926529e+00,   4.48280499e+00,   4.50813265e+00,
 4.55633565e+00,   4.58315561e+00,   4.59329086e+00,
 4.63298785e+00,   4.68478131e+00,   4.72719015e+00,
 4.74138291e+00,   4.79184382e+00,   4.84468916e+00,
 4.85679455e+00,   4.90929593e+00,   4.94929164e+00,
 4.98811406e+00,   5.06325758e+00,   5.06765207e+00,
 5.12783997e+00,   5.18807173e+00,   5.19435559e+00,
 5.22589557e+00,   5.26168783e+00,   5.27999497e+00,
 5.35536457e+00,   5.37821208e+00,   5.45327138e+00,
 5.50014127e+00,   5.51424913e+00,   5.56364684e+00,
 5.58603680e+00,   5.62293715e+00,   5.67449482e+00,
```

```
5.72932027e+00,   5.77039630e+00,   5.85105602e+00,
5.87259415e+00,   5.92547657e+00,   5.95222106e+00,
5.99581299e+00,   6.01909075e+00,   6.07490662e+00,
6.11835833e+00,   6.13720020e+00,   6.21228797e+00,
6.21792062e+00,   6.27217560e+00,   6.27764857e+00,
6.39122924e+00,   6.43218338e+00,   6.48589993e+00,
6.51586274e+00,   6.54892394e+00,   6.63608487e+00,
6.65377895e+00,   6.67835718e+00,   6.73982154e+00,
6.77744578e+00,   6.79577640e+00,   6.82153756e+00,
6.89237403e+00,   6.94806361e+00,   6.99463520e+00,
7.02973434e+00,   7.07669313e+00,   7.12081307e+00,
7.13711455e+00,   7.17527368e+00,   7.24032955e+00,
7.30071078e+00,   7.35426420e+00,   7.37225172e+00,
7.44281078e+00,   7.47463563e+00,   7.49061830e+00,
7.56465746e+00,   7.63573289e+00,   7.66789954e+00,
7.68741811e+00,   7.74460838e+00,   7.77226898e+00,
7.79989028e+00,   7.84354563e+00,   7.90095579e+00,
7.91370655e+00,   7.95162550e+00,   8.01313479e+00,
8.06741772e+00,   8.08997792e+00,   8.12270989e+00,
8.14868834e+00,   8.20504489e+00,   8.23635226e+00,
8.26036286e+00,   8.30395625e+00,   8.38742526e+00,
8.42866899e+00,   8.50794924e+00,   8.53237054e+00,
8.54714555e+00,   8.58225240e+00,   8.61787242e+00,
8.69295531e+00,   8.72400211e+00,   8.74721404e+00,
8.80191126e+00,   8.83489296e+00,   8.85332713e+00,
8.94785444e+00,   8.96921758e+00,   9.00519149e+00,
9.05641040e+00,   9.09292067e+00,   9.14749315e+00,
9.20696763e+00,   9.27296438e+00,   9.32627807e+00,
9.33188646e+00,   9.42911971e+00,   9.48734500e+00,
9.50460269e+00,   9.54717372e+00,   9.60925254e+00,
9.66955623e+00,   9.69056205e+00,   9.72230574e+00,
9.74541228e+00,   9.79042706e+00,   9.81068030e+00,
9.86856277e+00,   9.91797546e+00,   9.93228628e+00,
1.00470453e+01,   1.00612764e+01,   1.01087053e+01,
1.01517606e+01,   1.01669409e+01,   1.01934368e+01,
1.02403089e+01,   1.02763061e+01,   1.03381751e+01,
1.03905756e+01,   1.04287446e+01,   1.04377188e+01,
1.04814629e+01,   1.04999380e+01,   1.05701909e+01,
1.05872365e+01,   1.06277290e+01,   1.07012084e+01,
1.07137403e+01,   1.08106834e+01,   1.08715917e+01,
1.09059326e+01,   1.09767678e+01,   1.09831858e+01,
1.10356155e+01,   1.10875209e+01,   1.10969117e+01,
1.11525898e+01,   1.11774950e+01,   1.12461803e+01,
1.13217135e+01,   1.13895731e+01,   1.14075358e+01,
1.15058572e+01,   1.15076664e+01,   1.15473701e+01,
1.15819866e+01,   1.16329249e+01,   1.16749398e+01,
1.16988744e+01,   1.17461933e+01,   1.18165288e+01,
1.18846878e+01,   1.19129564e+01,   1.19724912e+01,
1.19941173e+01,   1.20610854e+01,   1.20692278e+01,
1.21207416e+01,   1.21615294e+01,   1.21938752e+01,
1.22673190e+01,   1.22739465e+01,   1.23054596e+01,
1.23460592e+01,   1.23737639e+01,   1.24663071e+01,
1.24852693e+01,   1.25051061e+01,   1.25643746e+01,
1.26340252e+01,   1.27219582e+01,   1.27680111e+01,
```

```
1.27948732e+01,   1.28313710e+01,   1.29485678e+01,
1.29596972e+01,   1.29923583e+01,   1.30638745e+01,
1.31097243e+01,   1.31846187e+01,   1.32205757e+01,
1.32470865e+01,   1.32777157e+01,   1.33068672e+01,
1.34123480e+01,   1.34556112e+01,   1.35155006e+01,
1.35276104e+01,   1.35897864e+01,   1.36046251e+01,
1.36712246e+01,   1.36885948e+01,   1.37418402e+01,
1.37579918e+01,   1.38506924e+01,   1.38645081e+01,
1.39520783e+01,   1.39764471e+01,   1.40442917e+01,
1.40939176e+01,   1.41777248e+01,   1.42116577e+01,
1.42592571e+01,   1.42831642e+01,   1.43435034e+01,
1.43813057e+01,   1.43876006e+01,   1.44796941e+01,
1.44924928e+01,   1.45473515e+01,   1.45660220e+01,
1.46982614e+01,   1.47052542e+01,   1.47392727e+01,
1.47887356e+01,   1.48526179e+01,   1.49350202e+01,
1.49952931e+01,   1.50729770e+01,   1.50945165e+01,
1.51297157e+01,   1.51909344e+01,   1.52331138e+01,
1.53154092e+01,   1.53489707e+01,   1.53730661e+01,
1.53943830e+01,   1.54380875e+01,   1.55101892e+01,
1.56183853e+01,   1.56504328e+01,   1.57130049e+01,
1.57317315e+01,   1.57829538e+01,   1.58575707e+01,
1.58944434e+01,   1.59049676e+01,   1.60208869e+01,
1.60640820e+01,   1.61073973e+01,   1.61626820e+01,
1.62437236e+01,   1.62822795e+01,   1.63244770e+01,
1.63701671e+01,   1.64103694e+01,   1.64744134e+01,
1.65231661e+01,   1.66072644e+01,   1.66328827e+01,
1.66603419e+01,   1.67446410e+01,   1.67582375e+01,
1.67825370e+01,   1.68349404e+01,   1.69204067e+01,
1.69605922e+01,   1.70195597e+01,   1.70342188e+01,
1.70969383e+01,   1.71258156e+01,   1.71875344e+01,
1.73269815e+01,   1.73960459e+01,   1.74478020e+01,
1.75356446e+01,   1.75715607e+01,   1.76325979e+01,
1.76676557e+01,   1.76963939e+01,   1.78073386e+01,
1.78598647e+01,   1.79547806e+01,   1.80073486e+01,
1.80710610e+01,   1.81296663e+01,   1.81567177e+01,
1.82276053e+01,   1.82679908e+01,   1.83341918e+01,
1.84010044e+01,   1.84270903e+01,   1.84798089e+01,
1.85522020e+01,   1.86135950e+01,   1.87402028e+01,
1.87830406e+01,   1.88061407e+01,   1.88793946e+01,
1.89196922e+01,   1.89456183e+01,   1.89946467e+01,
1.90657477e+01,   1.91193000e+01,   1.91764042e+01,
1.92266728e+01,   1.92458789e+01,   1.93390226e+01,
1.94000167e+01,   1.94961702e+01,   1.95302863e+01,
1.95765489e+01,   1.96731228e+01,   1.97574584e+01,
1.97771944e+01,   1.97990654e+01,   1.99072335e+01,
1.99625777e+01,   2.00040848e+01,   2.01035679e+01,
2.01892104e+01,   2.02134497e+01,   2.02651435e+01,
2.03538343e+01,   2.04758013e+01,   2.05123709e+01,
2.05702969e+01,   2.06396645e+01,   2.06879401e+01,
2.07493466e+01,   2.08532166e+01,   2.09245653e+01,
2.09929384e+01,   2.10668321e+01,   2.10826681e+01,
2.11225779e+01,   2.12289418e+01,   2.13066334e+01,
2.13867166e+01,   2.15091833e+01,   2.15234533e+01,
2.15797594e+01,   2.17294087e+01,   2.18113567e+01,
```

```
                2.18636895e+01,   2.19642491e+01,   2.20002768e+01,
                2.20518529e+01,   2.21089959e+01,   2.22828218e+01,
                2.23244636e+01,   2.24295550e+01,   2.24854301e+01,
                2.25656223e+01,   2.26827262e+01,   2.27531006e+01,
                2.28970742e+01,   2.29598259e+01,   2.30413797e+01,
                2.31260885e+01,   2.32207064e+01,   2.32732181e+01,
                2.33718843e+01,   2.34205880e+01,   2.36303324e+01,
                2.36669432e+01,   2.37529847e+01,   2.38455753e+01,
                2.39453285e+01,   2.40423190e+01,   2.41505027e+01,
                2.42205346e+01,   2.44647644e+01,   2.45482493e+01,
                2.46440908e+01,   2.47100467e+01,   2.48141223e+01,
                2.49162433e+01,   2.51373548e+01,   2.54662104e+01,
                1.00105564e+03])
```

```
In [15]: M = np.zeros( (2, 2, 2) )
         M
```

```
Out[15]: array([[[ 0.,  0.],
                 [ 0.,  0.]],

                [[ 0.,  0.],
                 [ 0.,  0.]]])
```

```
In [17]: M = np.zeros( (2, 2, 2, 2) )
         M
```

```
Out[17]: array([[[[ 0.,  0.],
                  [ 0.,  0.]],

                 [[ 0.,  0.],
                  [ 0.,  0.]]],


                [[[ 0.,  0.],
                  [ 0.,  0.]],

                 [[ 0.,  0.],
                  [ 0.,  0.]]]])
```

```
In [18]: def f(i, j):
             return i+j
```

```
In [21]: M = np.fromfunction(f, (5, 5))
         M
```

```
Out[21]: array([[ 0.,  1.,  2.,  3.,  4.],
                [ 1.,  2.,  3.,  4.,  5.],
                [ 2.,  3.,  4.,  5.,  6.],
                [ 3.,  4.,  5.,  6.,  7.],
                [ 4.,  5.,  6.,  7.,  8.]])
```

```
In [23]: def g(i, j, k):
             return i+j+k
```

```
In [25]: M = np.fromfunction(g, (5, 5, 5))
         M
```

```
Out[25]: array([[[ 0.,  1.,  2.,  3.,  4.],
                 [ 1.,  2.,  3.,  4.,  5.],
                 [ 2.,  3.,  4.,  5.,  6.],
                 [ 3.,  4.,  5.,  6.,  7.],
                 [ 4.,  5.,  6.,  7.,  8.]],

                [[ 1.,  2.,  3.,  4.,  5.],
                 [ 2.,  3.,  4.,  5.,  6.],
                 [ 3.,  4.,  5.,  6.,  7.],
                 [ 4.,  5.,  6.,  7.,  8.],
                 [ 5.,  6.,  7.,  8.,  9.]],

                [[ 2.,  3.,  4.,  5.,  6.],
                 [ 3.,  4.,  5.,  6.,  7.],
                 [ 4.,  5.,  6.,  7.,  8.],
                 [ 5.,  6.,  7.,  8.,  9.],
                 [ 6.,  7.,  8.,  9., 10.]],

                [[ 3.,  4.,  5.,  6.,  7.],
                 [ 4.,  5.,  6.,  7.,  8.],
                 [ 5.,  6.,  7.,  8.,  9.],
                 [ 6.,  7.,  8.,  9., 10.],
                 [ 7.,  8.,  9., 10., 11.]],

                [[ 4.,  5.,  6.,  7.,  8.],
                 [ 5.,  6.,  7.,  8.,  9.],
                 [ 6.,  7.,  8.,  9., 10.],
                 [ 7.,  8.,  9., 10., 11.],
                 [ 8.,  9., 10., 11., 12.]]])
```

2 Escribir y leer datos

```
In [44]: M = np.array([1., 2, 3, 40]).reshape(2,2)

In [45]: M

Out[45]: array([[ 1.,  2.],
                [ 3., 40.]])

In [50]: np.savetxt("M.dat", M, fmt="%10.5f", header="Una matriz interesante")

In [51]: %less M.dat

In [53]: N = np.loadtxt("M.dat")

In [54]: N

Out[54]: array([[ 1.,  2.],
                [ 3., 40.]])

In [57]: np.loadtxt?

In [59]: M = np.fromfunction(f, (3, 6) )

In [60]: M
```

```
Out[60]: array([[ 0.,  1.,  2.,  3.,  4.,  5.],
                [ 1.,  2.,  3.,  4.,  5.,  6.],
                [ 2.,  3.,  4.,  5.,  6.,  7.]])

In [61]: np.savetxt("M.dat", M, fmt="%10.5f", header="Una matriz interesante")

In [64]: N = np.loadtxt("M.dat", usecols=(2, 4, 5))
         N

Out[64]: array([[ 2.,  4.,  5.],
                [ 3.,  5.,  6.],
                [ 4.,  6.,  7.]])

In [65]: x, y, z = N.T

In [66]: x

Out[66]: array([ 2.,  3.,  4.])

In [69]: x, y, z = np.loadtxt("M.dat", usecols=(2, 4, 5), unpack=True)

In [70]: y

Out[70]: array([ 4.,  5.,  6.])
```

2.1 Sustitución en cadenas

```
In [71]: dx = 0.1
         nombre_archivo = "ecn_calor_dx.dat"
         nombre_archivo

Out[71]: 'ecn_calor_dx.dat'

In [76]: dx = 0.1
         nombre_archivo = "ecn_calor_%s.dat"
         nombre_archivo

Out[76]: 'ecn_calor_%s.dat'

In [77]: nombre_archivo % dx

Out[77]: 'ecn_calor_0.1.dat'

In [83]: nombre_archivo = "ecn_calor_{0}_dx{0}.dat".format(dx)
         nombre_archivo

Out[83]: 'ecn_calor_0.1_dx0.1.dat'

In [85]: !./ecn_calor $dx

/bin/sh: ./ecn_calor: No such file or directory

In [86]: %%bash
         ls
         echo "Hola"
```

```
01 Introducción a Python.ipynb
02 El paquete numpy para vectores y matrices.ipynb
03 Matplotlib, sympy, mpmath.ipynb
04 Aritmética con intervalos.ipynb
05a Intervalos como objetos.ipynb
05b Intervalos como objetos II.ipynb
06a_AritmeticaIntervalos-Cont.ipynb
06b Más operaciones con intervalos.ipynb
07 Aún más operaciones interválicas.ipynb
08 Excepciones.ipynb
09 Manejando git.ipynb
1. Sintaxis básica de Python.html
1. Sintaxis básica de Python.ipynb
1. Sintaxis básica de Python.tex
1. Sintaxis básica de Python_files
12 Diferenciación automática.ipynb
12 Redondeo.ipynb
12a_AritmeticaPuntoFlotante.ipynb
12b_PuntoFlotanteRedondeo.ipynb
2. Estructuras de control.html
2. Estructuras de control.ipynb
2. Estructuras de control.pdf
2. Estructuras de control.tex
2. Estructuras de control_files
3. Librerías.ipynb
4. numpy.ipynb
M.dat
Método de potencias.ipynb
Tests_ipynb
Untitled0.ipynb
clase-12-agosto-2013.ipynb
dif_auto.py
dif_auto.pyc
intervalo.py
intervalo.pyc
nuevo
raices.py
raices.pyc
test_dif_auto.py
test_dif_auto.pyc
test_intervalo.py
test_intervalo.pyc
Hola
```

```
In [87]: a = _
```

```
In [88]: a
```

```
Out[88]: './ecn_calor 0.1'
```

```
In [89]: a = 1
```

```
In [90]: s = "a = a y b = b"
```

```
In [94]: s1 = "Hola"
         s2 = "David"
```

```
          s1 + ", " + s2
```

```
Out[94]: 'Hola, David'
```

```
In [99]: s1 + ", " + s2 + ", el valor de a es " + str(a)
```

```
Out[99]: 'Hola, David, el valor de a es 1'
```

```
In [100]: s = "3"
```

```
In [103]: int(s)
```

```
Out[103]: 3
```

```
In [104]: float(s)
```

```
Out[104]: 3.0
```

```
In [107]: dx = 0.1
          dy = 0.1
          s = "ecn_calor_dx" + str(dx) + "_dy" + str(dy)
          s
```

```
Out[107]: 'ecn_calor_dx0.1_dy0.1'
```

```
In [118]: s = "ecn_calor_%s"
```

```
In [119]: s
```

```
Out[119]: 'ecn_calor_%s'
```

Un nuevo operador %:

```
In [120]: s % dx
```

```
Out[120]: 'ecn_calor_0.1'
```

```
In [121]: dx = 0.1
          dy = 0.2
```

```
In [125]: s = "ecn_calor_dx%s_dy%s" % (dx, dy)
          s
```

```
Out[125]: 'ecn_calor_dx0.1_dy0.2'
```

Otra sintaxis (más nueva):

```
In [127]: s = "ecn_calor_dx{}_dy{}".format(dx, dy)
          s
```

```
Out[127]: 'ecn_calor_dx0.1_dy0.2'
```

```
In []:
```