

TicCis2014

June 2, 2014

Python atractivo para Científicos e Ingenieros



I Congreso de Tecnologías de la Información (CIS - U N L)
Milton Labanda ([@miltonlab](#))

0.1 Que hace generalmente un científico ?

1. **Observa**
2. Realiza experimentos o **simulaciones**
3. Obtiene **datos** de los experimentos
4. **Procesa** y **visualiza** los datos obtenidos
5. Propone **modelos**, entre otras cosas ...

0.2 Qué es Python ?

- **Python.org** dice *“Python is a programming language that lets you work quickly and integrate systems more effectively”*
- Un lenguaje de propósito general que puede funcionar como: **procedimental, funcional y orientado a objetos**.
- Un lenguaje limpio, simple y **“muy expresivo”**: con pocas palabras se dice y se hace mucho
- Su intérprete y sus herramientas tienen **licenciamiento de Software Libre** compatible con la licencia **GPL**
- A disposición miles de **librerías, paquetes y módulos** para diversos ámbitos

0.3 IPython Notebook:

- **“El paper ejecutable para los científicos”**
- Un intérprete interactivo avanzado que ha revolucionando la manera en que se utiliza Python en ámbitos científicos, conferencias y tutoriales**
- Mucho más que un shell: autocompletado, historial, ayuda en línea, salida al sistema, gestión de archivos ...

- Mucho ahorro de tiempo en escritura de código y pruebas
- Se puede ejecutar código de otros lenguajes: *R, Octave, Cython, Bash, Perl, Ruby, etc.*
- Permite escribir en formato Markdown, HTML, o texto puro (raw)
- Permite escribir en **LateX**
- Permite incluir imágenes, links a archivos locales, videos, etc
- Exporta hacia difentes formatos tales como: PDF, RST, Latex, reveal.js, etc

0.4 Numpy

- **Estructuras base para los cálculos**
- Código parecido a notación científica
- Código pythónico y sin bucles
- Muchas funciones matemáticas aplicables a matrices de datos
- Array SLICES (indexado) multidimensionales
- Funciones propias de Arreglos: diagonal, transpose, where, unique, fill, etc.
- Persistencia de arreglos en archivos de texto

0.5 NumPy práctico

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

Los arreglos, la estructura básica

```
In [2]: a = np.array([[5,2,9,4],[1,8,8,9],[6,6,4,7],[3,5,6,2]])
a
```

```
Out[2]: array([[5, 2, 9, 4],
               [1, 8, 8, 9],
               [6, 6, 4, 7],
               [3, 5, 6, 2]])
```

```
In [3]: # Seleccionar columna 2
a[:,2]
```

```
Out[3]: array([9, 8, 4, 6])
```

```
In [4]: # Seleccionar fila 2
a[2,:]
```

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

```
In [5]: # Selección de un subarreglo
a[0:2,1:3]
```

```
Out[5]: array([[2, 9],
               [8, 8]])
```

```
In [6]: # Selección condicional
np.where(a%2==0,a,0)
```

```
Out[6]: array([[0, 2, 0, 4],
               [0, 8, 8, 0],
               [6, 6, 4, 0],
               [0, 0, 6, 2]])
```

```
In [7]: a.flatten()
```

```
Out[7]: array([5, 2, 9, 4, 1, 8, 8, 9, 6, 6, 4, 7, 3, 5, 6, 2])
```

```
In [8]: c=a.copy()
        c.sort(axis=1)
        c
```

```
Out[8]: array([[2, 4, 5, 9],
               [1, 8, 8, 9],
               [4, 6, 6, 7],
               [2, 3, 5, 6]])
```

```
In [9]: np.diag((5,4,3,5))
```

```
Out[9]: array([[5, 0, 0, 0],
               [0, 4, 0, 0],
               [0, 0, 3, 0],
               [0, 0, 0, 5]])
```

```
In [10]: np.identity((4))
```

```
Out[10]: array([[ 1.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  1.]])
```

Funciones y trabajo con Arreglos

```
In [11]: a>5
```

```
Out[11]: array([[False, False,  True, False],
                [False,  True,  True,  True],
                [ True,  True, False,  True],
                [False, False,  True, False]], dtype=bool)
```

```
In [12]: a*3
```

```
Out[12]: array([[15,  6, 27, 12],
                [ 3, 24, 24, 27],
                [18, 18, 12, 21],
                [ 9, 15, 18,  6]])
```

```
In [13]: a/2
```

```
Out[13]: array([[2, 1, 4, 2],
                [0, 4, 4, 4],
                [3, 3, 2, 3],
                [1, 2, 3, 1]])
```

```
In [14]: np.sum(a), np.sum(a,0), np.sum(a,1)
```

```
Out[14]: (85, array([15, 21, 27, 22]), array([20, 26, 23, 16]))
```

```
In [15]: a.mean(), np.median(a), a.std(), a.var()
```

```
Out[15]: (5.3125, 5.5, 2.4423029603224902, 5.96484375)
```

```
In [16]: a.max(), a.max(0), a.max(1)
```

```
Out[16]: (9, array([6, 8, 9, 9]), array([9, 9, 7, 6]))
```

Algebra Lineal

```
In [17]: a
```

```
Out[17]: array([[5, 2, 9, 4],
                [1, 8, 8, 9],
                [6, 6, 4, 7],
                [3, 5, 6, 2]])
```

```
In [18]: # Matriz triangular superior e inferior
         np.tril(a)
```

```
Out[18]: array([[5, 0, 0, 0],
                [1, 8, 0, 0],
                [6, 6, 4, 0],
                [3, 5, 6, 2]])
```

```
In [19]: np.triu(a)
```

```
Out[19]: array([[5, 2, 9, 4],
                [0, 8, 8, 9],
                [0, 0, 4, 7],
                [0, 0, 0, 2]])
```

```
In [20]: np.linalg.inv(a)
```

```
Out[20]: array([[ 0.0372093 , -0.13488372,  0.14418605,  0.02790698],
                [-0.16877076,  0.00465116,  0.02458472,  0.23056478],
                [ 0.10431894,  0.03255814, -0.11362126,  0.04252492],
                [ 0.05315615,  0.09302326,  0.06312292, -0.24584718]])
```

```
In [21]: np.linalg.matrix_power(a,2)
```

```
Out[21]: array([[ 93, 100, 121, 109],
                [ 88, 159, 159, 150],
                [ 81, 119, 160, 120],
                [ 62,  92, 103, 103]])
```

```
In [22]: # El arreglo debe ser cuadrado
         np.linalg.det(a)
```

```
Out[22]: 1504.9999999999998
```

```
In [23]: a.transpose()
```

```
Out[23]: array([[5, 1, 6, 3],
                [2, 8, 6, 5],
                [9, 8, 4, 6],
                [4, 9, 7, 2]])
```

```
In [24]: np.identity(3)
```

```
Out[24]: array([[ 1.,  0.,  0.],
                [ 0.,  1.,  0.],
                [ 0.,  0.,  1.]])
```

Operaciones con matrices

```
In [25]: b = np.array([[5, 4, 2, 0], [9, 9, 6, 1], [3, 0, 5, 3], [8, 0, 6, 1]])
        b
```

```
Out[25]: array([[5, 4, 2, 0],
               [9, 9, 6, 1],
               [3, 0, 5, 3],
               [8, 0, 6, 1]])
```

```
In [26]: np.vdot(a,b)
```

```
Out[26]: 310
```

```
In [27]: a>b
```

```
Out[27]: array([[False, False,  True,  True],
               [False, False,  True,  True],
               [ True,  True, False,  True],
               [False,  True, False,  True]], dtype=bool)
```

```
In [28]: a.dot(b)
```

```
Out[28]: array([[102,  38,  91,  33],
               [173,  76, 144,  41],
               [152,  78, 110,  25],
               [ 94,  57,  78,  25]])
```

```
In [29]: a * b
```

```
Out[29]: array([[25,  8, 18,  0],
               [ 9, 72, 48,  9],
               [18,  0, 20, 21],
               [24,  0, 36,  2]])
```

```
In [30]: np.concatenate((a,b))
```

```
Out[30]: array([[5, 2, 9, 4],
               [1, 8, 8, 9],
               [6, 6, 4, 7],
               [3, 5, 6, 2],
               [5, 4, 2, 0],
               [9, 9, 6, 1],
               [3, 0, 5, 3],
               [8, 0, 6, 1]])
```

Utilitarios

```
In [31]: a.tolist()
```

```
Out[31]: [[5, 2, 9, 4], [1, 8, 8, 9], [6, 6, 4, 7], [3, 5, 6, 2]]
```

```
In [32]: a.tostring()
```

```
Out[32]: '\x05\x00\x00\x00\x02\x00\x00\x00\t\x00\x00\x00\x04\x00\x00\x00\x01\x00\x00\x00\x08\x00\x00\x00'
```

```
In [33]: a.tofile('/tmp/arreglo.txt')
```

Polinomios

```
In [34]: # Visualizando LATEX de paso en IPython Notebook !!!
```

\$ polinomio $\rightarrow x^4 - 11x^3 + 9x^2 + 11x - 10$ \$

```
In [35]: # Coeficientes de Polinomios de menor a mayor
p=np.polynomial.Polynomial([-10,11,9,-11,1])
p.roots() #raices del polinomio
```

```
Out[35]: array([ -1.          ,  0.99999998,  1.00000002, 10.          ])
```

```
In [36]: # Obtencion de los coeficientes a partir de las raices
np.poly([-1,1,1,10])
```

```
Out[36]: array([ 1, -11,  9, 11, -10])
```

```
In [37]: # Ejemplo 2 de polinomio
ecuacion2=np.polynomial.Polynomial([-8,2,1])
ecuacion2.roots()
```

```
Out[37]: array([-4.,  2.])
```

```
In [38]: # Evaluación de un polinomio en un punto particular
# las fuciones numpy.poly* reciben los coeficientes de mayor a menor
np.polyval([1,2,-8], 4)
```

```
Out[38]: 16
```

0.6 Matplotlib

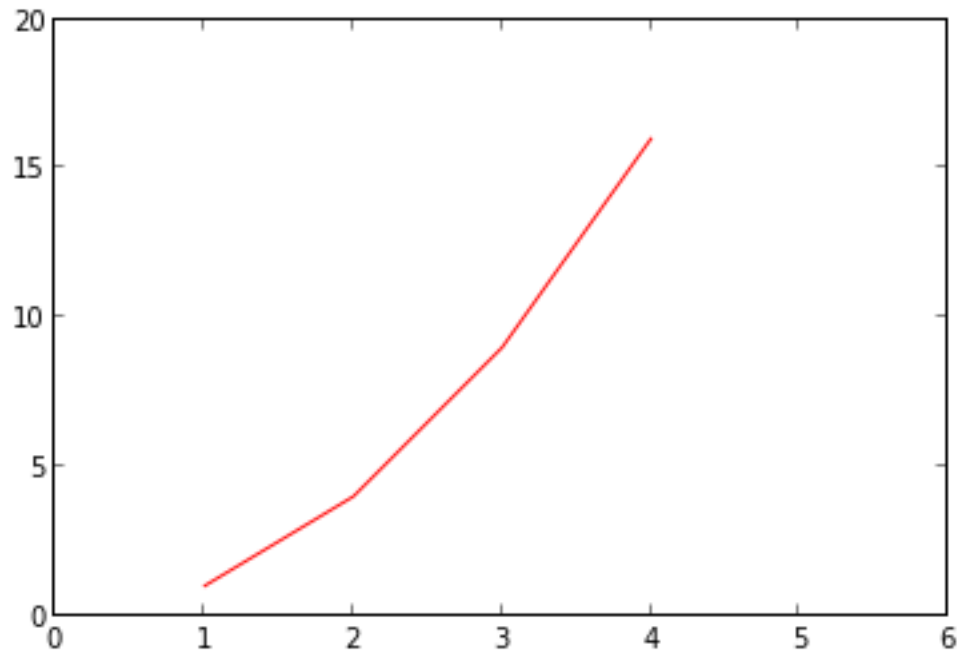
- Visualizacion de datos
- Una librería para el trazo (dibujo) orientada a objetos
- Trazado de diferentes estilos de gráficos: barras, pastel, histogramas, etc
- Personalización en detalle del aspecto visual

0.7 Matplotlib práctico

```
In [39]: import numpy as np
from matplotlib import pyplot
%matplotlib inline
```

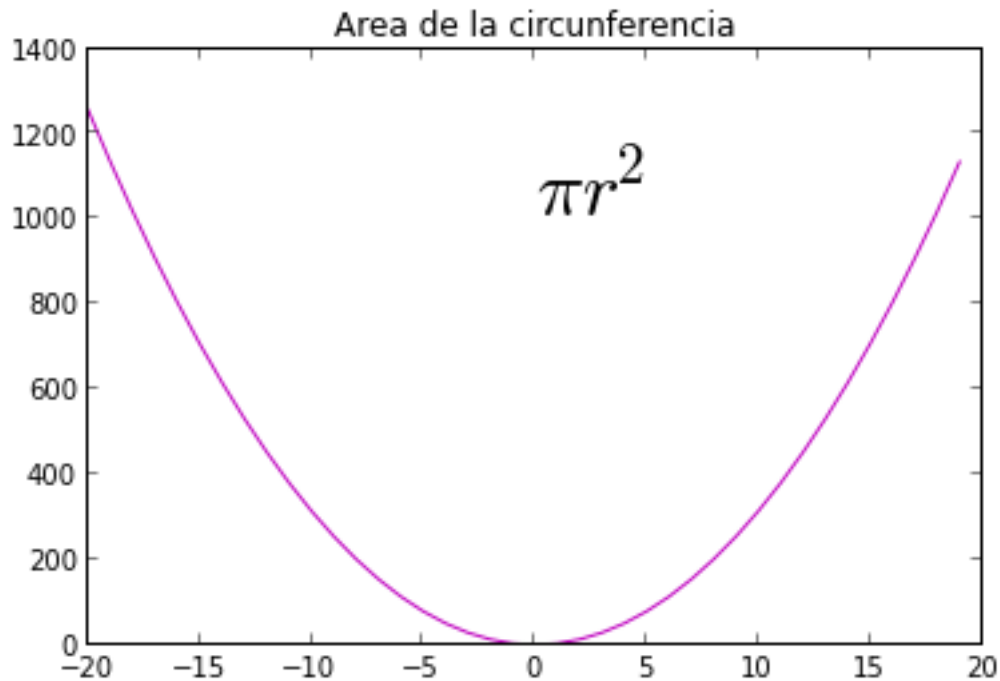
```
In [40]: pyplot.axis([0, 6, 0, 20])
pyplot.plot([1,2,3,4], [1,4,9,16], 'r-')
```

```
Out[40]: [<matplotlib.lines.Line2D at 0xb0a310c>]
```



```
In [41]: # Modificando el aspecto
x=np.arange(-20,20)
pyplot.title('Area de la circunferencia')
# Latex
pyplot.text(0,1000, r'$\pi r^2$', fontsize=30)
pyplot.plot(x, x*x*np.pi, 'm')
#pyplot.show()
```

```
Out[41]: [<matplotlib.lines.Line2D at 0xb27c5cc>]
```



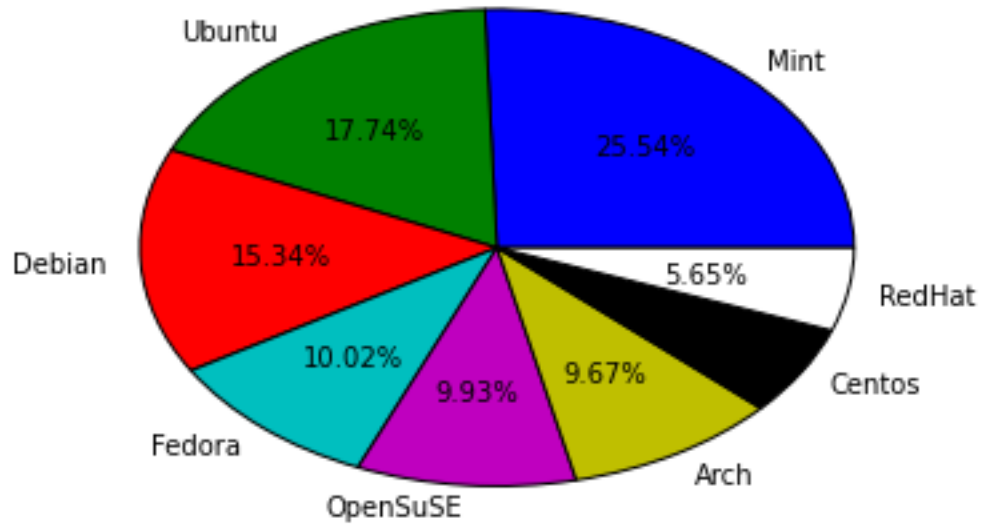
```
In [42]: # Graficos de Pastel
distros = ['Mint', 'Ubuntu', 'Debian', 'Fedora', 'OpenSuSE', 'Arch',
           'Centos', 'RedHat']
# Datos según http://distrowatch.com la tercera semana de Mayo del 2014
ranking = [3105, 2157, 1865, 1218, 1207, 1176, 743, 687]
```

```
pyplot.pie(ranking, labels=distros, autopct='%.2f%%')
#pyplot.show()
```

```
Out[42]: ([<matplotlib.patches.Wedge at 0xb54c6ec>,
<matplotlib.patches.Wedge at 0xb54cb8c>,
<matplotlib.patches.Wedge at 0xb54cfac>,
<matplotlib.patches.Wedge at 0xb55772c>,
<matplotlib.patches.Wedge at 0xb557b2c>,
<matplotlib.patches.Wedge at 0xb557f2c>,
<matplotlib.patches.Wedge at 0xb88dfec>,
<matplotlib.patches.Wedge at 0xb88d40c>],
[<matplotlib.text.Text at 0xb54ca2c>,
<matplotlib.text.Text at 0xb54cecc>,
<matplotlib.text.Text at 0xb55764c>,
<matplotlib.text.Text at 0xb557a4c>,
<matplotlib.text.Text at 0xb557e4c>,
<matplotlib.text.Text at 0xb55726c>,
<matplotlib.text.Text at 0xb88d32c>,
<matplotlib.text.Text at 0xb88d72c>],
[<matplotlib.text.Text at 0xb54cb4c>,
<matplotlib.text.Text at 0xb54ca8c>,
<matplotlib.text.Text at 0xb54cf2c>,
<matplotlib.text.Text at 0xb54cf6c>],
```

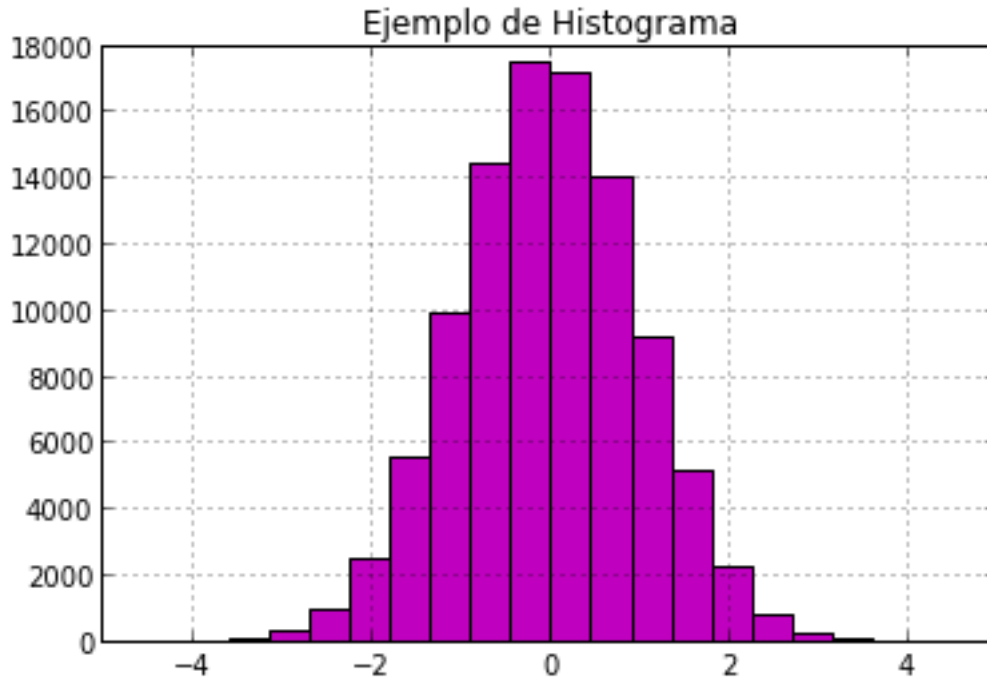


```
<matplotlib.text.Text at 0xb557aac>,  
<matplotlib.text.Text at 0xb557eac>,  
<matplotlib.text.Text at 0xb5572cc>,  
<matplotlib.text.Text at 0xb55730c>]]
```



```
In [43]: x=np.random.randn(100000)  
         pyplot.hist(x, facecolor='m', bins=20)  
         pyplot.grid(True)  
         pyplot.title('Ejemplo de Histograma')
```

```
Out[43]: <matplotlib.text.Text at 0xb8b268c>
```



0.8 SciPy

- **La librería para computación científica**
- Colección de herramientas y algoritmos matemáticos
- Resolución de problemas de ingeniería y ciencias tales como optimización, integración, procesamiento de señales e imágenes, etc

0.9 SciPy práctico

```
In [44]: import scipy
import scipy.optimize
```

Ej.: Encontrar las raíces de la ecuación no lineal:

$$x + 2\cos(x) = 0$$

```
In [45]: def f(x):
return x + 2 * scipy.cos(x)
```

```
In [46]: scipy.optimize.bisect(f, -2, 2)
```

```
Out[46]: -1.0298665293221347
```

```
In [47]: scipy.optimize.newton(f, 2)
```

```
Out[47]: -1.0298665293222757
```

Integrar una función de una variable entre dos puntos

Ej.: Resolver la ecuación

$$A = \int_0^2 x^2 dx$$

```
In [48]: import scipy.integrate
def f(x):
    y = x * x
    return y
result, error = scipy.integrate.quad(f,0,2.0)
result, error
```

```
Out[48]: (2.6666666666666665, 2.9605947323337504e-14)
```

Ej.: Resolver la ecuación lineal ordinaria:

$$\frac{dx}{dt} = -x$$

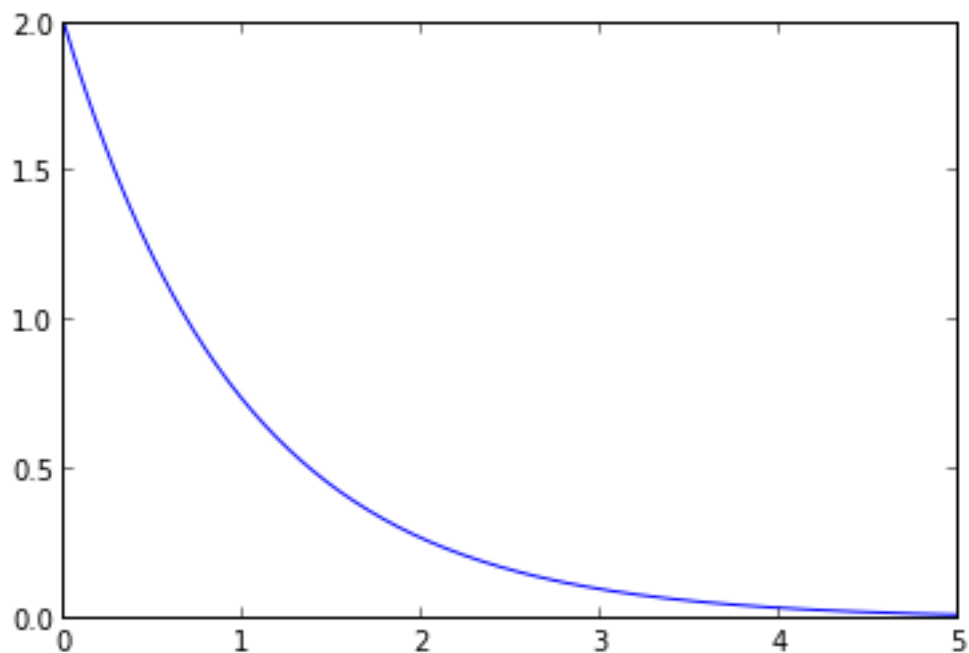
con la condición inicial

$$x(0) = 2$$

```
In [49]: def dx_dt(x, t=0):
    y = -x
    return y
t = scipy.linspace(0,5,1000)
x0 = 2
x = scipy.integrate.odeint(dx_dt, x0, t)
```

```
In [50]: import matplotlib; import matplotlib.pyplot as pyplot
pyplot.plot(t,x)
```

```
Out[50]: [<matplotlib.lines.Line2D at 0xb5b48ec>]
```



0.10 Enlaces

- Sitio Oficial del lenguaje de programación Python python.org
- IPython ipython.org
- SciPy www.scipy.org
- Conferencia anual de Python Científico conference.scipy.org/scipy2014
- Blog Pybonacci pybonacci.wordpress.com