

PyConES 2013

Python + Ciencia = ♥

Juan Luis Cano, @Pybonacci
Madrid, 2013-11-23

Y este ¿quién es?

- ✓ *Casi* ingeniero aeronáutico
- ✓ **Fortran 90** (¿77?) y **Excel** (¡!)
- ✓ Herramientas privativas
- ✓ ...conseguidas de manera ilegítima

Respuesta: ¡Python!

El inicio de una gran amistad



- ✓ **Python** por ~~frustración~~ cuenta propia
- ✓ Invierno 2011:
Python Madrid
- ✓ Resultado: **Pybonacci**

Comienzos: *«Dividing & merging»*



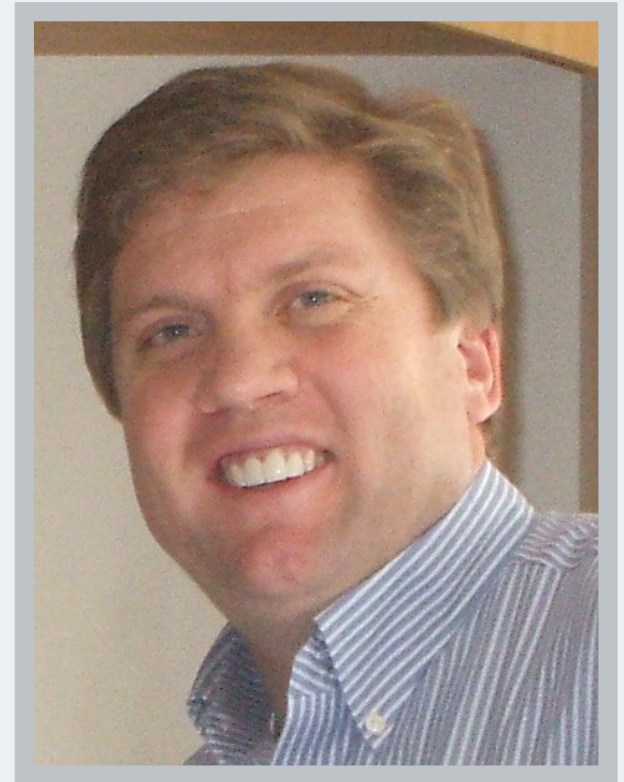
1995: Numeric



- ✓ Jim Hugunin (MIT) et al
- ✓ Objeto **array** básico
- ✓ Python como herramienta para **cálculo científico**

1997: *Travis meets Python*

- ✓ Travis E. Oliphant (Mayo Clinic, Minnesota)
- ✓ Se enamora de Python y abandona MATLAB
- ✓ Usa Numeric para crear lo que será **SciPy** en 2001



2001: Una odisea pythonica



- ✓ T. Oliphant, Pearu Peterson y Eric Jones liberan **SciPy**
- ✓ Fernando Pérez inicia **IPython**
- ✓ John Hunter † crea **matplotlib**



2003: El cisma: numarray

- ✓ **Limitaciones** de Numeric
- ✓ Perry Greenfield y otros crean **numarray**
- ✓ Mejoras... y defectos
- ✓ Confusión: ¿cuál usar?




2006

Numeric + numarray + Travis Oliphant* =
NumPy

«Dividing & merging»

*Y muchos más

A person with curly hair is playing a guitar, looking down at the instrument. The background is a solid blue color. The text is overlaid on the right side of the image.

Presente: *NumPy,* *SciPy* *y más allá*

Foto: Marcingietorigie (CC-BY-SA)

NumPy

- ✓ **Arrays multidimensionales**
- ✓ Funciones rápidas y eficientes para operar con ellos
- ✓ Otros: álgebra lineal, FFTs, números aleatorios, funciones financieras
- ✓ Motivación: «*make [Python] equivalent to a basic scientific calculator*»

Arrays

```
>>> import numpy as np
```

```
>>> np.array( [  
...   [1,  2,  3],  
...   [4,  5,  6]  
...   ] )  
array([[1,  2,  3],  
       [4,  5,  6]])
```

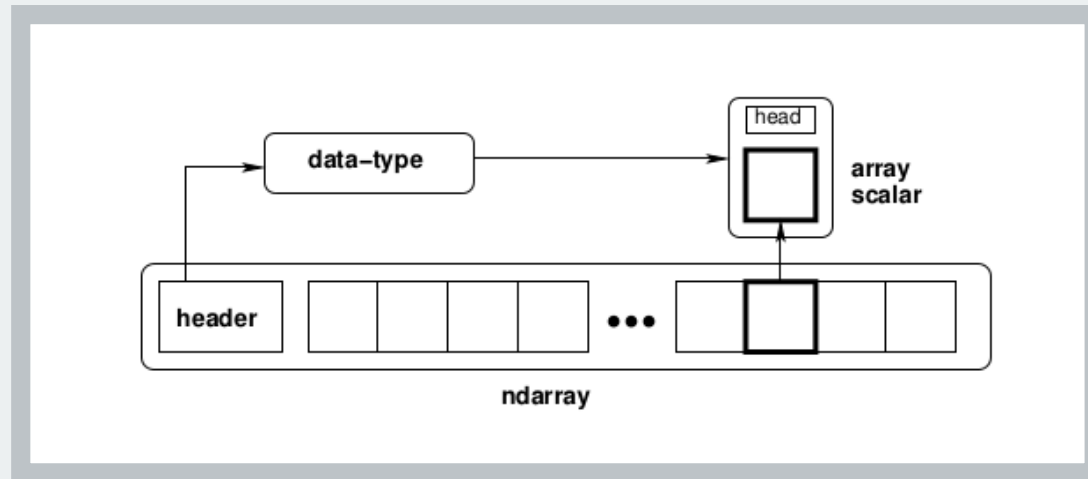

Universal functions

```
>>> a = np.arange(6).reshape(2, 3)
```

```
>>> np.sin(a)
```

```
array([[ 0.          ,  0.8415,  0.9093],  
       [ 0.1411, -0.7568, -0.9589]])
```


Ventajas



- ✓ Datos homogéneos y dimensiones fijas: almacenamiento en memoria **eficiente**
- ✓ Los bucles en Python son lentos: **vectorización**
- ✓ Operaciones sobre los datos en bloque: **expansión** (*broadcasting*)

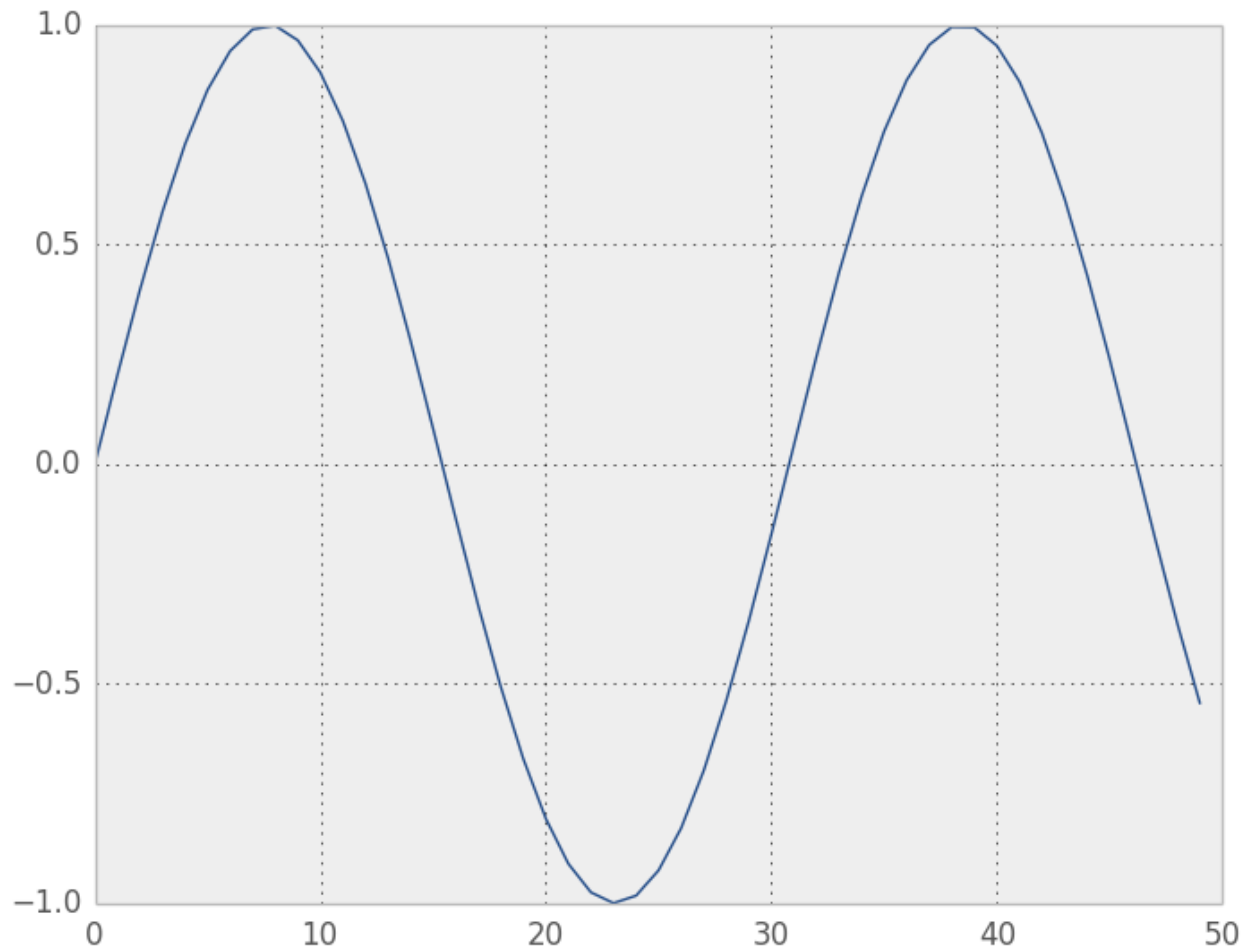
matplotlib

- ✓ El estándar *de facto* para **visualización** con Python
- ✓ Basado en la API de MATLAB
- ✓ Gráficas de alta calidad (*publication-quality*)
- ✓ Fundamentalmente para 2D

Visualización rápida

```
>>> import matplotlib.pyplot as plt
>>> x = np.linspace(0, 10)
>>> plt.plot(np.sin(x))
[<matplotlib.lines.Line2D object at
0x7f2107caa9d0>]
>>> plt.show()
```


Visualización rápida



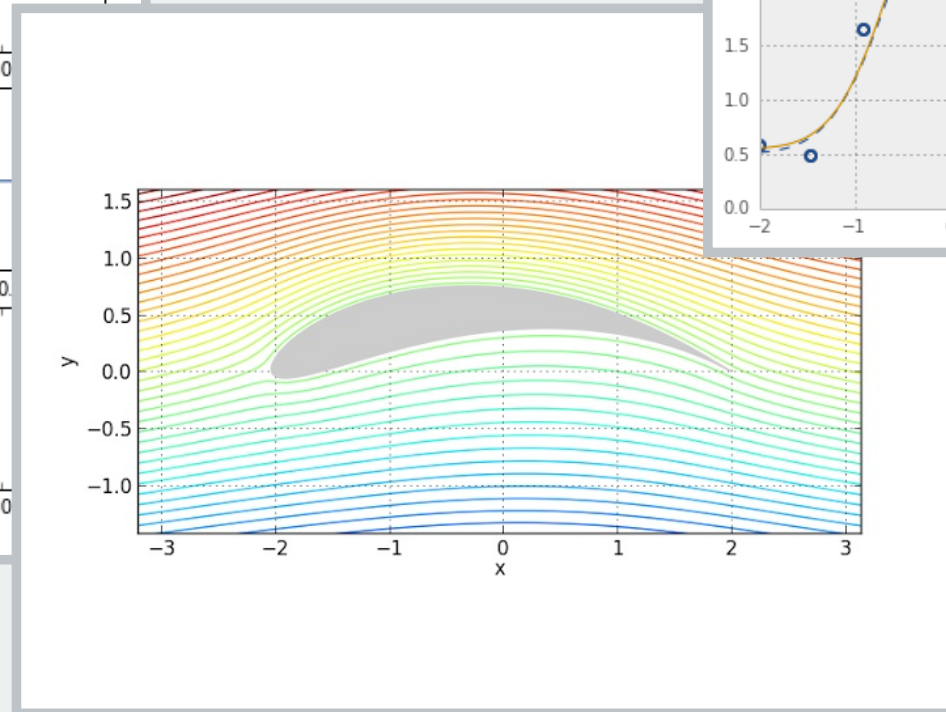
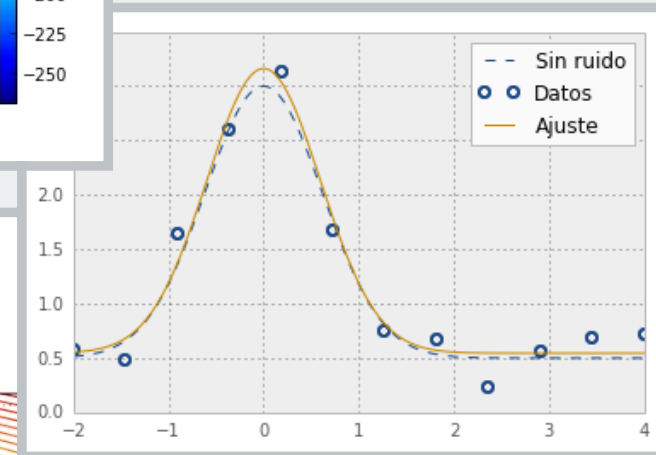
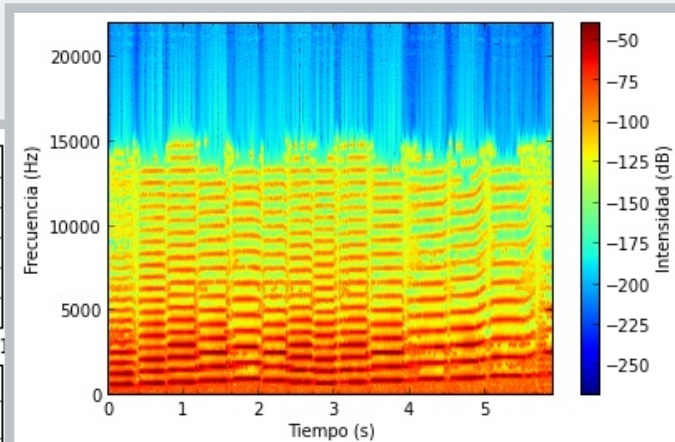
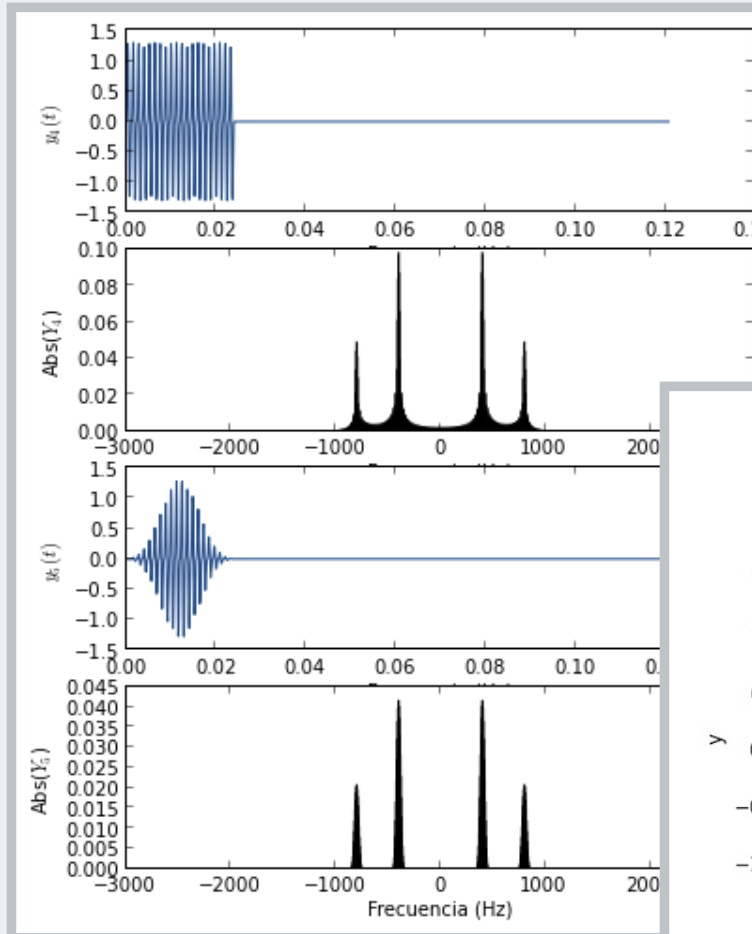
¿Por qué (no) matplotlib?

- ✓ Familiar para los usuarios de MATLAB
- ✓ ...horrible para los usuarios de R
- ✓ Potente: todo se puede personalizar
- ✓ ...pero a veces es un poco *low-level*
- ✓ Suficiente para el 95 % de los casos
- ✓ Para el otro 5 %: Mayavi, Bokeh, ggplot, Vincent...

SciPy

- ✓ Colección de algoritmos para tareas comunes
 - Integración y EDOs (`scipy.integrate`)
 - Procesamiento de señales (`scipy.signal`)
 - Funciones especiales (`scipy.special`)
 - Optimización (`scipy.optimize`)
 - Interpolación (`scipy.interpolate`)
 - ...¡y más!

Para tareas básicas...

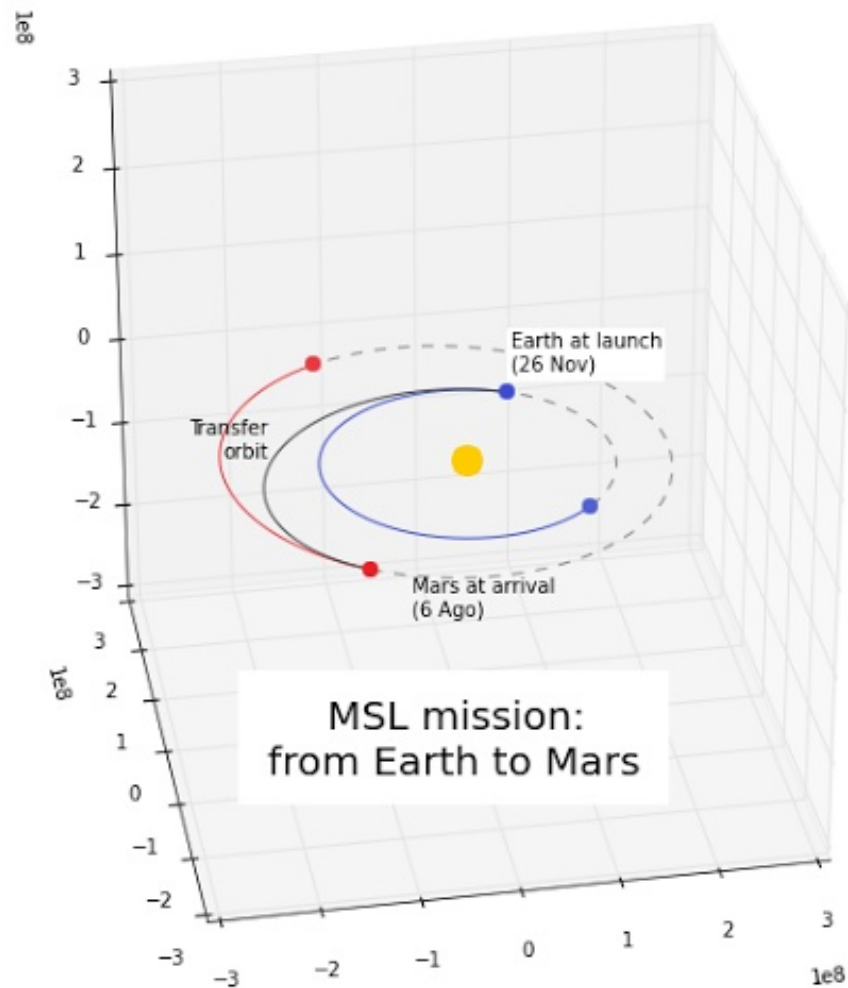


...y no tan básicas

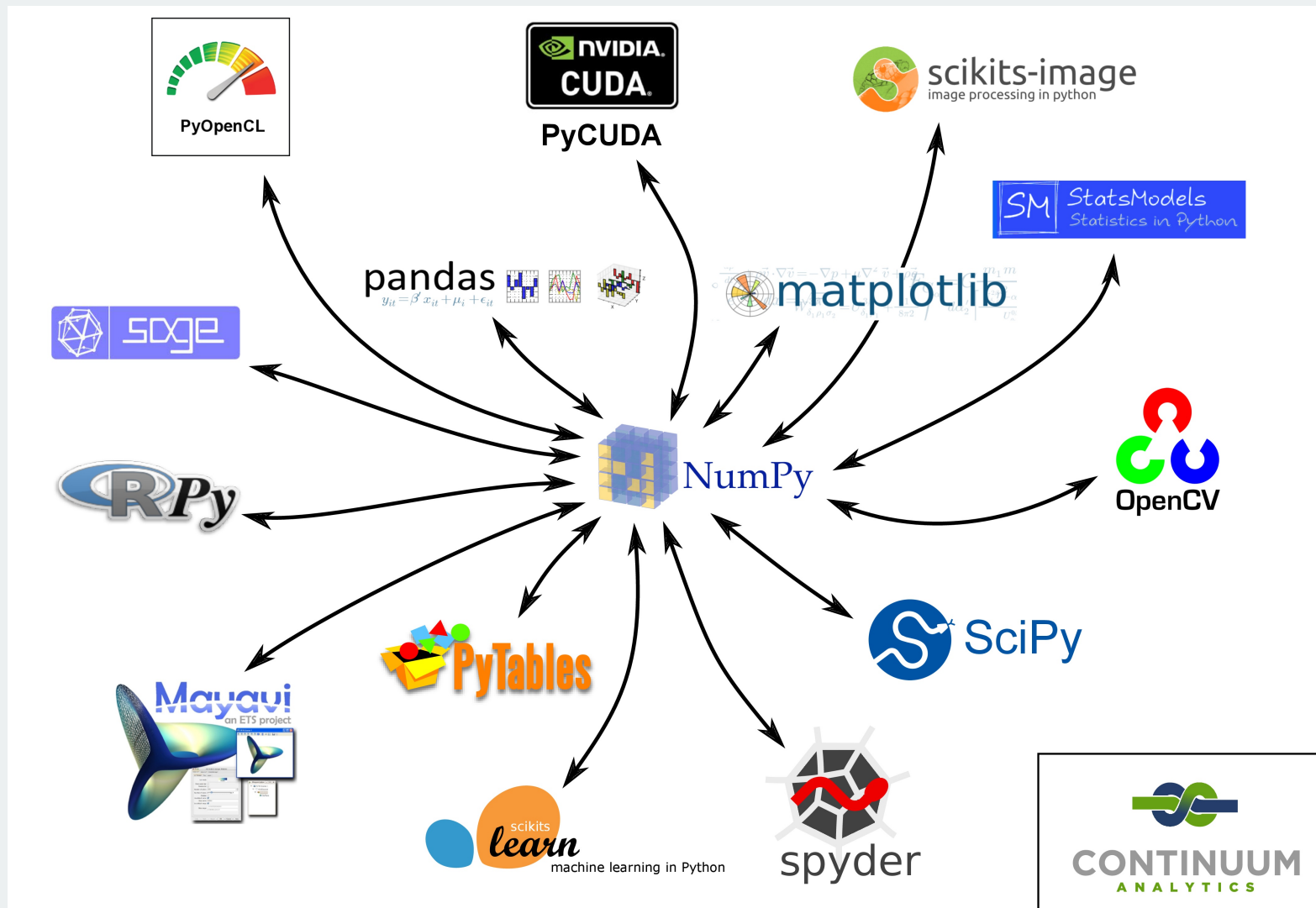
¡Dentro vídeo!

<http://youtu.be/8K4NgNVKdtM>

¡Hasta el infinito...



...y más allá!



Bonus: SymPy

- ✓ NumPy: cálculo numérico

```
>>> np.sqrt(8)
```

```
2.8284271247461903
```

- ✓ **SymPy: cálculo simbólico**

```
>>> from sympy import sqrt
```

```
>>> sqrt(8)
```

```
2*sqrt(2)
```


Un CAS en Python

- ✓ Sistema de álgebra computacional (CAS) estilo Maple, Mathematica o Maxima
- ✓ Escrito en Python puro
- ✓ Intérprete online: <http://live.sympy.org/>
- ✓ Soporte para LaTeX

```
>>> Integral(1 / x, x)
```

$$\int \frac{1}{x} dx$$

IPython:

La revolución



IPython

- ✓ Originalmente *Interactive Python*: **intérprete de Python mejorado**
- ✓ Iniciado por Fernando Pérez en 2001 inspirado en Mathematica
- ✓ Diciembre 2011: IPython 0.11, notebook con interfaz web

¡Dentro demo!

...y llegó el dinero

- ✓ Diciembre 2012: **\$1.15M** de la fundación Alfred P. Sloan
- ✓ Expansión significativa de IPython y su interfaz notebook
- ✓ Agosto 2013: **IPython 1.0** y **\$100k** de Microsoft

¿Por qué es tan increíble?

- ✓ Comunicación de **ideas** mediante **código**
- ✓ **Ciencia abierta**
- ✓ Entorno interactivo ideal para el aprendizaje

Python vs MATLAB:

¿David contra Goliat?



El statu quo

«The most dangerous enemy of a better solution is an existing codebase that is just good enough.»

—Eric S. Raymond.

El statu quo

Python



«The most dangerous enemy of a **better solution** is an **existing codebase** that is just good enough.»

MATLAB



—Eric S. Raymond.

El *statu quo*

- ✓ En la industria y en el mundo académico hay **inercias**
- ✓ *¿Desde dónde tiene que empezar el cambio?*
- ✓ No siempre es posible o deseable: código legado, experiencia

¡Pero Python es mejor!

- ✓ Coste de licencia: **\$0.0**
- ✓ **Software libre**: puedo estudiarlo y compartirlo
- ✓ *Commercial-friendly*: no copyleft

...también *técnicamente* mejor

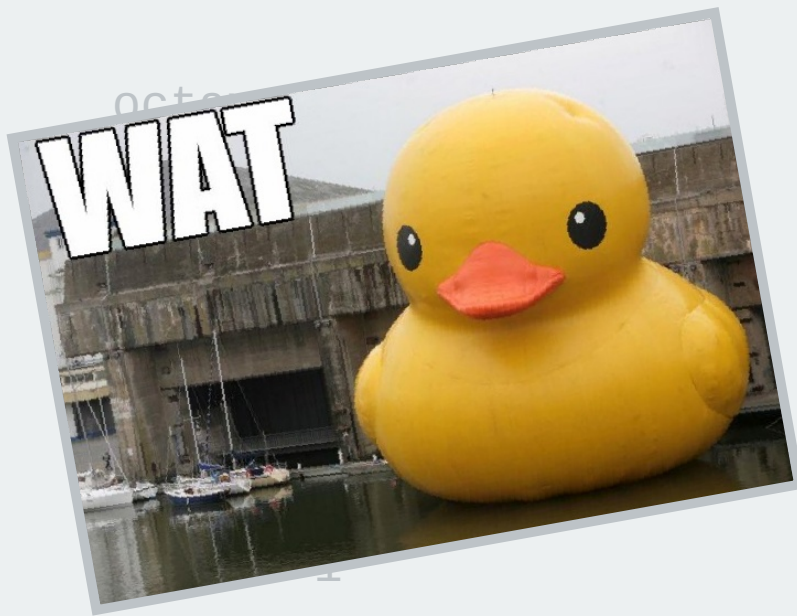
- ✓ **Software libre** (otra vez): los fallos son públicos
- ✓ Lenguaje más sólido y consistente (también conocido como: MATLAB WAT)

```
octave:1> a = [1]
a = 1
octave:2> a(1)
ans = 1
octave:3> a(1, 1, 1)
ans = 1
```

```
>>> a = np.array([1]); a
array([1])
>>> a[0]
1
>>> a[0, 0, 0]
IndexError: too many indices
```


...también *técnicamente* mejor

- ✓ **Software libre** (otra vez): los fallos son públicos
- ✓ Lenguaje más sólido y consistente (también conocido como: MATLAB WAT)



```
>>> a = np.array([1]); a  
array([1])
```

```
>>> a[0]
```

```
1
```

```
>>> a[0, 0, 0]
```

```
IndexError: too many indices
```


Python FTW!

- ✓ ¡La gente está pidiendo interfaz notebook para MATLAB!
- ✓ El desarrollo del ecosistema Python es **vertiginoso**
- ✓ Actualmente está por delante en aprendizaje automático, tratamiento de datos...

¿Nos estás ocultando algo?

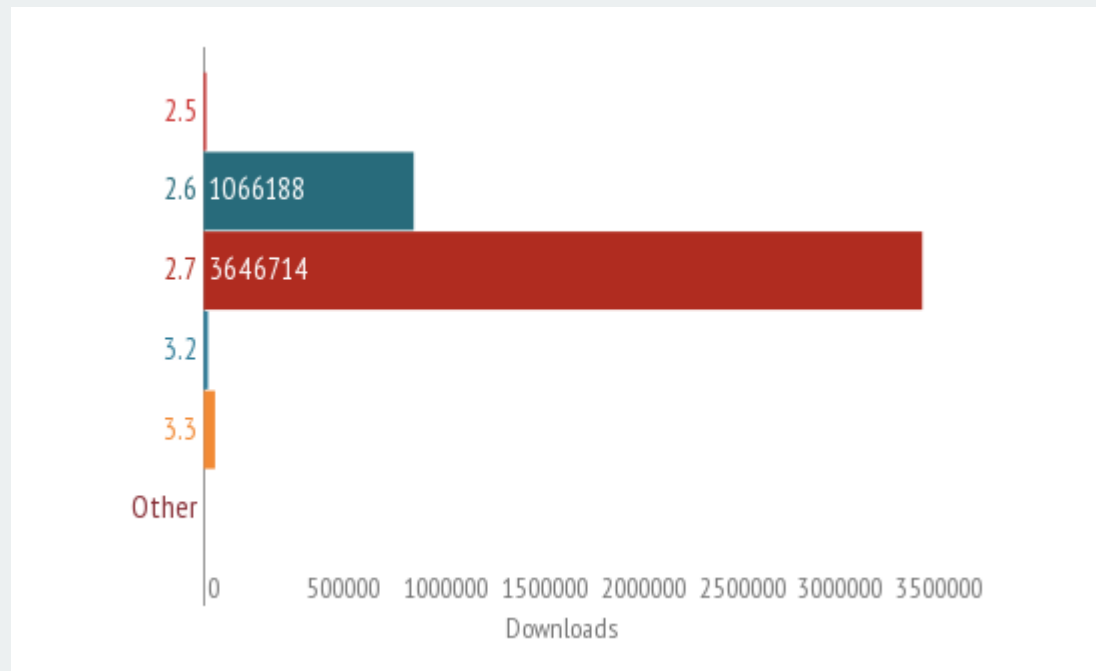
Puntos débiles:

*«Too few are lifting
too many»*



Python 2 → Python 3

- ✓ Cambio de versiones: ¡horror!
- ✓ Python 3.0 en diciembre de 2008, y aun así:



Python 2 → Python 3

- ✓ Se cometieron errores que se están solucionando ahora (2013)
- ✓ Clave: **desterrar 2to3, código único** para ambas versiones
- ✓ El ecosistema está listo: ¡migremos!

Python es más... verborreico

- ✓ El inicio de un programa Python suele ser así:

```
import numpy as np
from numpy import cos, sin, tan, [...]
import matplotlib
import matplotlib.pyplot as plt
from scipy import integrate, optimize
import os
import re
...
```

- ✓ Para sesiones interactivas es muy incómodo

Algunas con inicios de solución

- ✓ Uso de memoria: $2 * a + 3 * b$ necesita tres **arrays intermedios**

Solución: numexpr, numba, ¿otros?

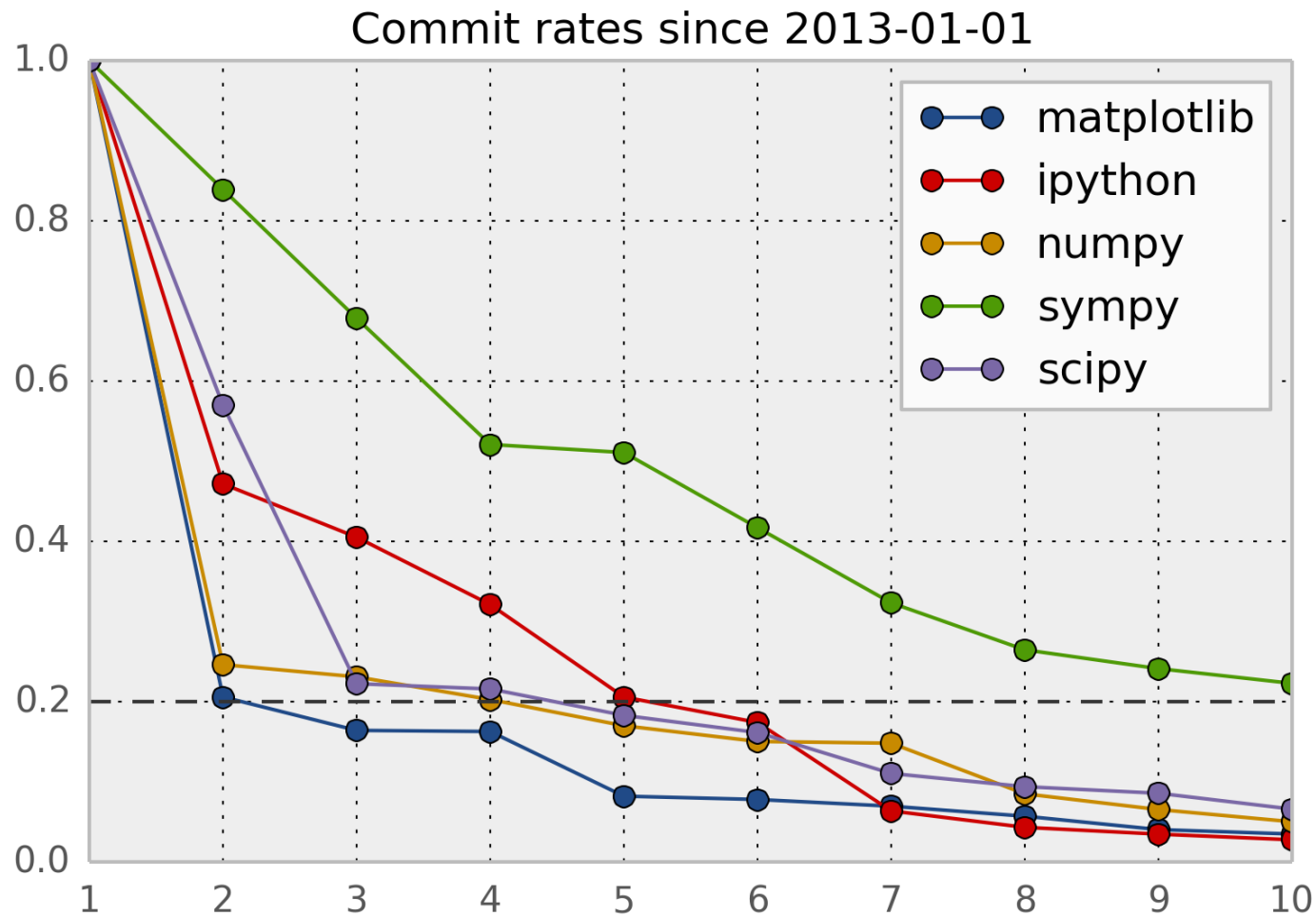
- ✓ Dificultad de **instalación** fuera de Linux (e.g. Windows)

Solución: distribuciones como Canopy o Anaconda

- ✓ Falta de **interfaces gráficas** para aplicaciones ingenieriles

Solución: ¿Simulink en Python, alguien?

«Too few are lifting too many»



Futuro: *¿Dominación mundial?*



*«These days, tools for almost **every aspect** of scientific computing are readily available in **Python**.»*

*«[...] a surprising number of Python-based tools are now **best-in-class** (or close to it) in terms of scope and ease of use—and, in virtue of C bindings, often even in terms of performance»*

The homogenization of scientific computing, or why Python is steadily eating other languages' lunch

«Nowadays **Python is probably the programming language of choice** (besides R) for data scientists for prototyping, visualization, and running data analyses on small and medium sized data sets.»

How Python became the language of choice for data science

¡Un futuro brillante!

- ✓ SciPy: hoja de ruta para 1.0
- ✓ IPython: plan de desarrollo repleto de novedades
- ✓ Nuevos scikits emergen y los existentes mejoran
- ✓ Se empieza a implantar como opción en las universidades españolas

Solo una gráfica más



Conclusiones



Python crece

- ✓ El camino ha sido arduo, pero el ecosistema está maduro
- ✓ Python se está expandiendo
- ✓ Pero hay inercias difíciles de vencer
- ✓ Debemos poner cuidado en algunas áreas:
contribuciones de código y migración a Python 3
- ✓ Podemos dominar el mundo :)

¿Preguntas?

¡Muchas gracias! :)

<http://pybonacci.wordpress.com>
@Pybonacci