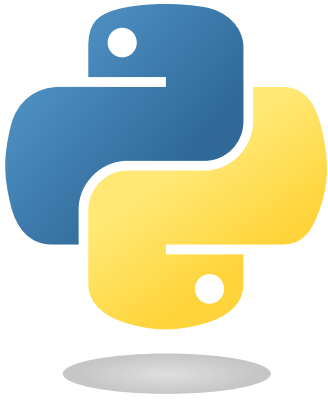


python™

Python en Supercomputación

Charla introductoria

Guillem Borrell i Nogueras

ETSIA, Octubre 2007

Preguntas...

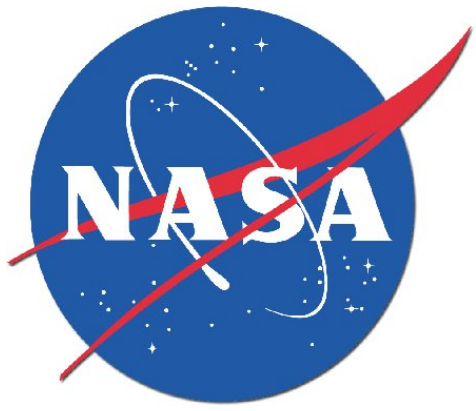
- ¿Por qué se llama Python?
- ¿Quién usa Python?
- ¿Para qué sirve Python?
  - Principales características de Python
- ¿Por qué Python se está volviendo tan popular?
- ¿Por qué Python y no otro lenguaje?
- ¿Qué puede ofrecer Python al HPC?
  - Inconvenientes de Python
- Ejemplos

¿Por qué se llama Python?

Python debe su nombre a...

Monty Python

¿Quién usa Python?



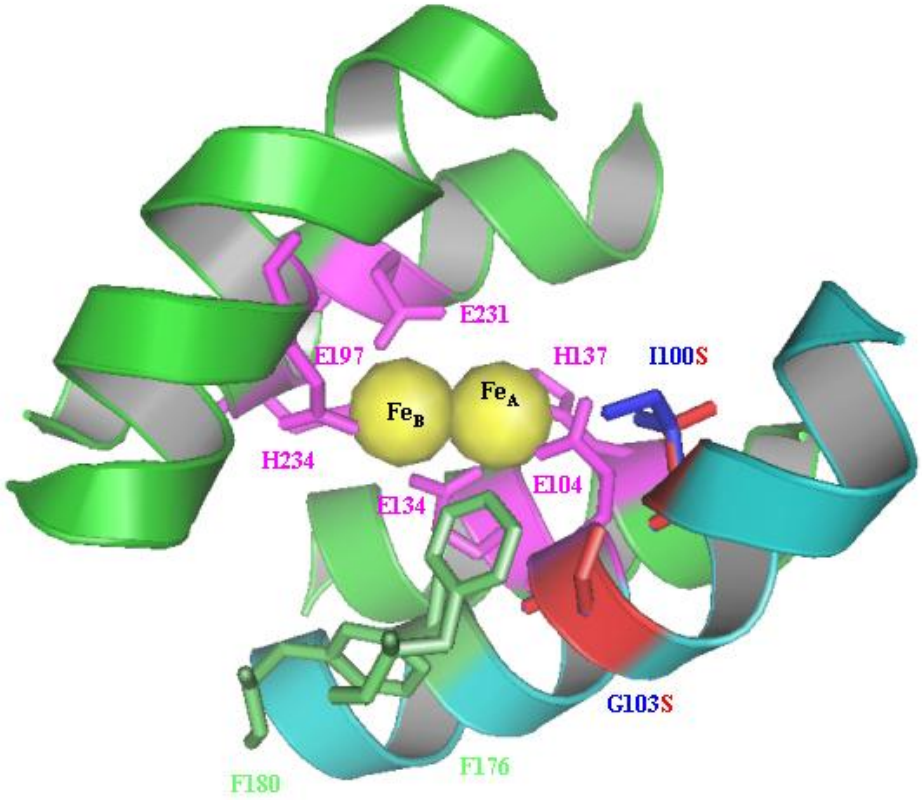
¿Para qué sirve Python?

Prácticamente para cualquier cosa que se nos pueda ocurrir

Desde páginas web...

Plone, Zope

...A utilidades para bioinformática



Pymol

Principales características de Python

- Software Libre (Licencia estilo BSD)
- Interpretado
- Interactivo
- Multiparadigma
  - Procedimental
  - Modular
  - Orientado a Objetos
- Multiplataforma
- Especificación especialmente corta (IronPython, Jython, PyPy, Stackless)
- *Incluye las pilas*
- ...

¿Por qué Python se está volviendo tan popular

- Fácil de aprender
- Fácil de ampliar
- Consistente por diseño
- Impone un buen estilo de programación
- Soporta todas las prácticas propuestas por XP, Agile.

Porque es divertido

¿Por qué Python y no otro lenguaje?

- Fácilmente extensible
  - CPython, escrito en ANSI C
- Duck Typing (VS. Java y C++)
- Software Libre
- Excelente documentación

¿Qué pinta tiene código escrito en Python?

Todo es un objeto

```
guillem@aiguaviva ~ $ python
Python 2.4.4 (#1, Sep 25 2007, 21:44:53)
[GCC 4.1.2 (Gentoo 4.1.2)] on linux2
Type "help", "copyright", "credits" or "license" ...
>>> import cmath
>>> i=cmath.sqrt(-1) #Los namespaces son objetos
>>> i.conjugate() #Los números son objetos
-1j
>>> i.imag
1.0
>>> i.real
0.0
>>> i+=3
>>> i
(3+1j)
```

Definir funciones es muy fácil

```
>>> def sumauno(numero):
...     return numero+1
...
>>> sumauno(i)
(4+1j)
```

- No hay llaves ni ends, los niveles se definen por el sangrado.

Documentación a la Matlab

```
>>> def docfunc():
...     """Esta es una función documentada, a diferencia
...     de Matlab la documentación de las funciones de
...     Python es extraíble y formateable"""
...     pass
...
>>> help(docfunc)
Help on function docfunc in module __main__:

docfunc()
    Esta es una función documentada, a diferencia de
    Matlab la documentación de las funciones de Python
    es extraíble y formateable
```

Una clase llamada pato

```
 >>> class pato:
...     cantidad = 1
...     def haz_cua(self):
...         print "cua!"
...
...     def reproducete(self):
...         cantidad += 1
...
>>> estoesunpato=pato() #instancia de pato
>>> estoesunpato.cantidad
1
>>> estoesunpato.haz_cua()
cua!
```

Duck Typing

Si algo anda como un pato y hace cua como un pato para mi va a ser un pato.

```
>>> estoesunpato=pato() #instancia de pato
>>> cuaqueador(estoesunpato)
cua!
>>> class guillem:
...     def haz_cua(self):
...             print "cua!"
...
>>> falsopato=guillem() #ese soy yo
>>> cuaqueador(falsopato)
cua!
>>> isinstance(falsopato,pato)
False
```

Para la función cuaqueador yo soy tan pato como un pato.

Y mucho más...

¿Qué puede ofrecer Python al HPC?

Python es lento, $\sim 10\times$C; Python pretende completar C y Fortran, no sustituirlos.

Sirve para...

Manejar la Complejidad

Wrappers, Interfaces, Prototipado, Scripting...

Inconvenientes de Python (para HPC)

- Diseño *Single Thread* (GIL)!
- Implementación limpia vs. optimizada
- Librería estándar escrita en python (20 %-30 % C)
- Versatilidad vs. potencia.
- No hay una implementación propia de Arrays y Buffers
- Es interpretado (¿?)

Soluciones a los inconvenientes

- Inlining (Weave)
- Extending (ctypes)
- Paralelo (PyMPI, ctypes, ParallelPython)
- PyPy
- Stackless
- Numpy, Scipy...

Numpy

Es una extensión de Python que soporta arrays n-dimensionales

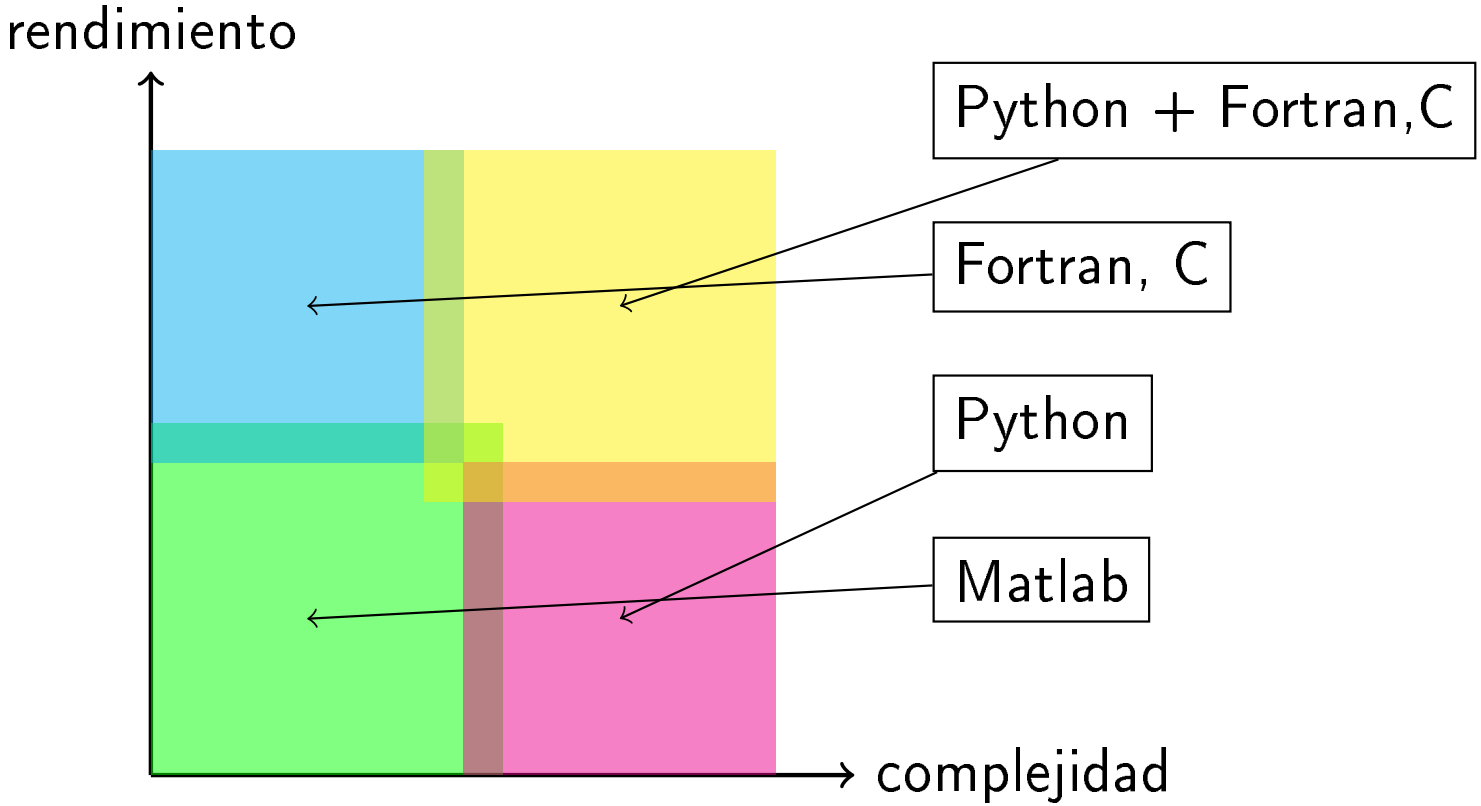
Es una maravilla, verdad verdadera. Lástima que no se pueda demostrar en una transparencia

Una pequeña introducción

```
>>> import numpy as N
>>> x=N.array([[1,2,3,2],[2,3,4,3],
               [2,3,4,3],[3,2,3,4]],'d')
>>> N.fft.rfft2(x)
array([[ 44.+0.j,  -6.+2.j,   0.+0.j],
       [ -4.+0.j,  -2.+2.j,   0.+0.j],
       [ -4.+0.j,  -2.-2.j,   0.+0.j],
       [ -4.+0.j,   2.-2.j,   0.+0.j]])
>>> x.transpose()
array([[ 1.,  2.,  2.,  3.],
       [ 2.,  3.,  3.,  2.],
       [ 3.,  4.,  4.,  3.],
       [ 2.,  3.,  3.,  4.]])
```

¿Cuál es la idea entonces?

Introducir Python en HPC $\Leftrightarrow$ Crear una aplicación multilenguaje.



¿Qué se gana añadiendo Python a C, Fortran?

- Namespaces
- Abstracción (Modularidad, OO)
- Añadir interactividad
- Autodocumentación
- Crear cajas negras
  - Si no tengo que saberlo no me lo cuentes
  - Si no tengo que verlo no me lo enseñes
  - Si algo funciona bien, recíclalo

¿Cómo se añade Python a C y Fortran?

Haciendo Wrappers

Objetivo: crear un Matlab © muy personalizado.

- Ventajas
  - Añaden interactividad, potencia, versatilidad...
  - No hay que repetirlos
- Inconvenientes
  - Más decisiones de diseño
  - Requieren más esfuerzo

¿Por qué Python?

Porque en muchos casos:

- Los wrappers ya estarán hechos (lapack, blas, fftpack, mpi)
- Hacerlos requerirá un esfuerzo mínimo

Gracias a:

- **ctypes**
- **f2py** (Fortran)
- SWIG (C, C++)
- weave
- ...

Continuará...

Charla técnica

- Python Vs. Matlab.
- Más sobre los wrappers.
- Arrays en C, Fortran y Python.
- GIL.
- Uso de F2Py.
- ctypes.
- Python en paralelo.