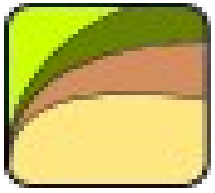


atmantree.com

Integrando mis Librerías C++ con Python

Ing. Carlos Gustavo Ruiz
http://atmantree.com
Noviembre, 2012

Temas

- El problema del PVP
- Por qué C++ con Python
- Opciones de integración
  - Python/C API
  - Cython
  - Swig
  - Boost
  - Otros
- A modo de conclusión

El problema del PVP

¿Qué piensa cuando
hablan de PVP?

El problema del PVP

El problema del PVP

El problema del PVP

Ok, si.. No dice BsF.

El problema del PVP

Como Ingeniero yo pienso en...

El problema del PVP

Productivity
vs
Performance

(Productividad contra Desempeño)

El problema del PVP

Hablar de Performance es hablar de

C – Fortran – C++

El problema del PVP

Hablar de Productividad es hablar de

Python – Ruby
Perl – PHP

El problema del PVP

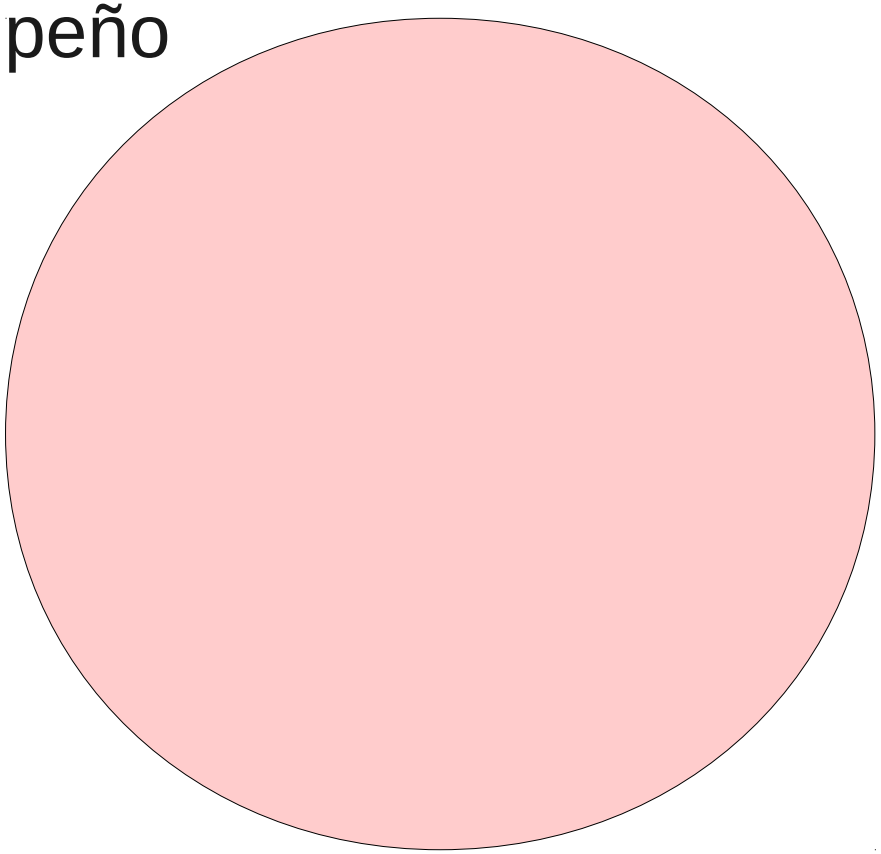
Hablar de Productividad es hablar de

Python – Ruby
Perl – PHP

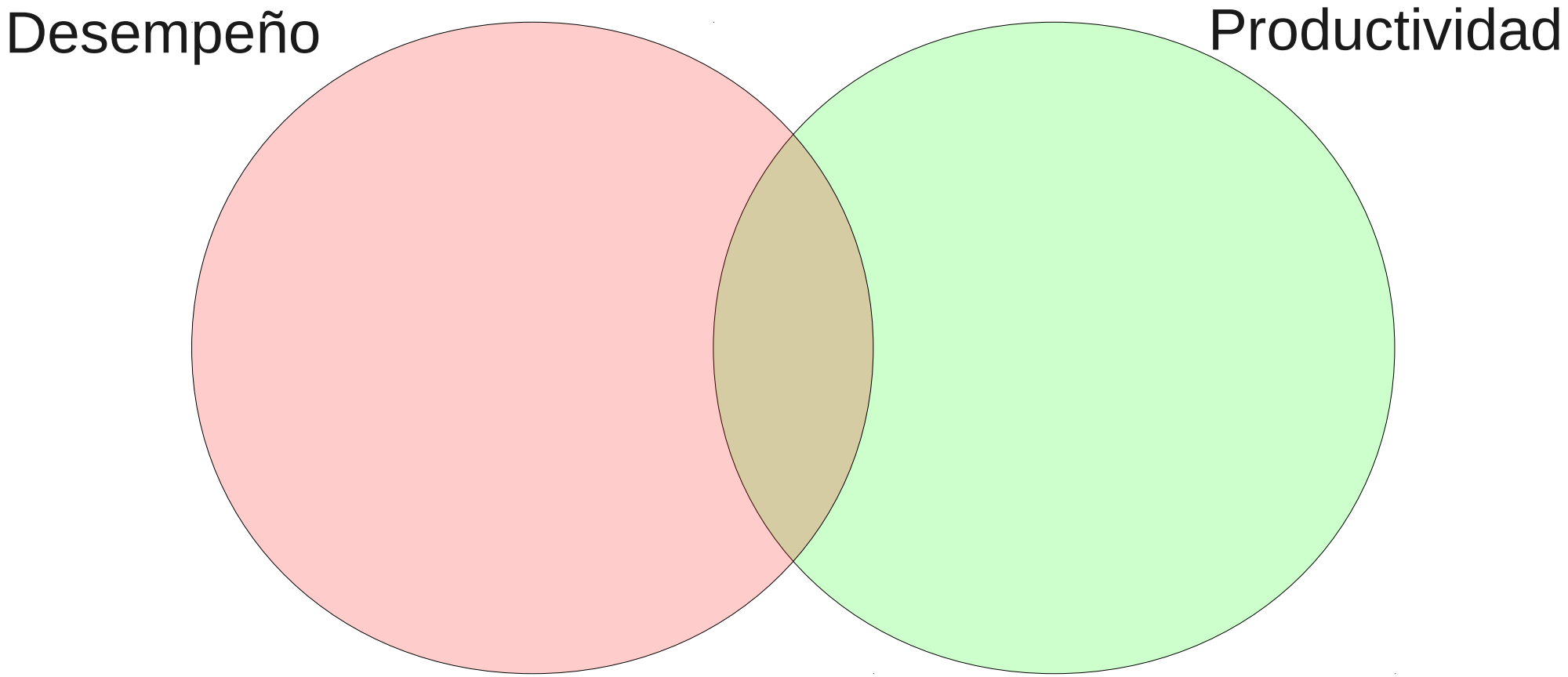
¿C# – Java?

El problema del PVP

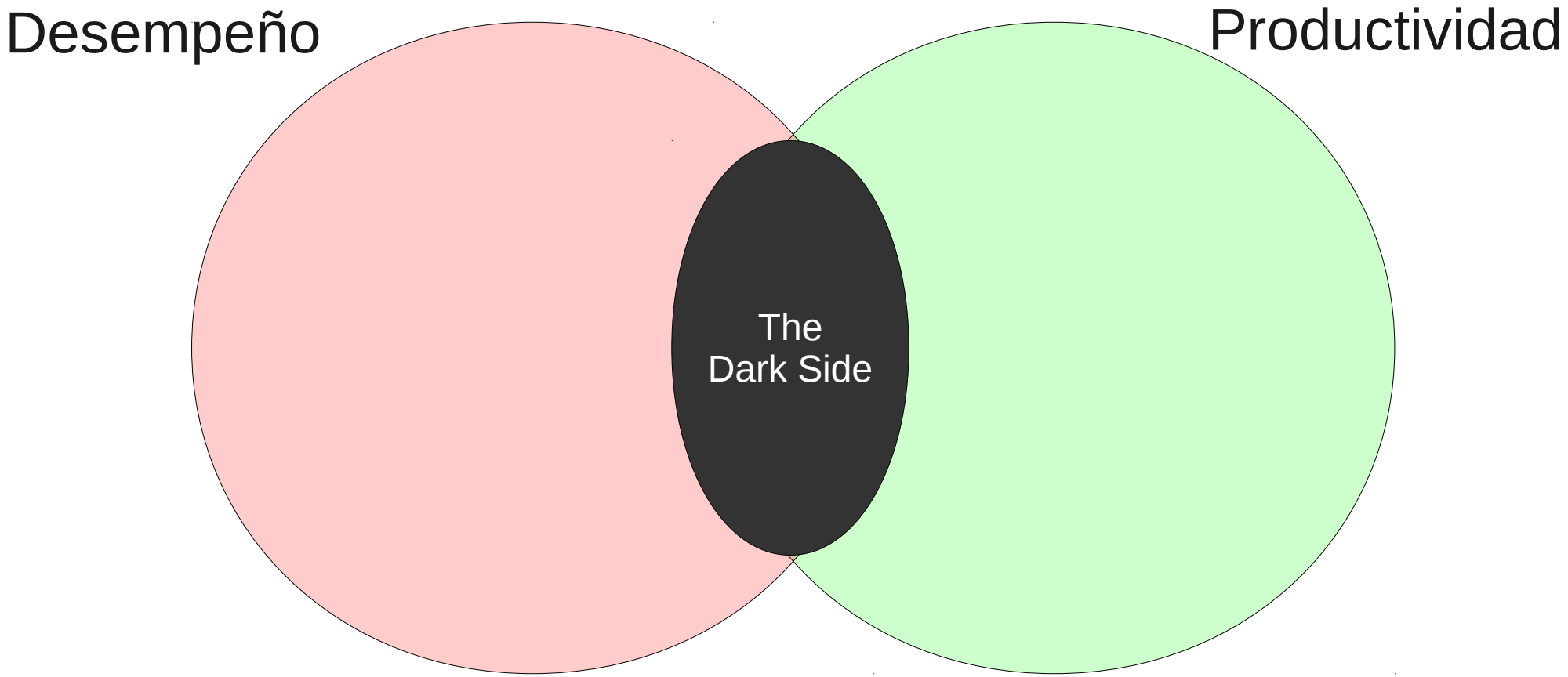
Desempeño



El problema del PVP



El problema del PVP



El problema del PVP

Por qué C++ con Python

- ¿Por qué Python?
  - Versatilidad
  - Multiplataforma
  - Respuestas para todo
  - Rapidez de desarrollo
  - Difusión
  - Ahora con PyConVE

Por qué C++ con Python

- ¿Por que C++?
  - Amplia cantidad de librerías
  - Flexibilidad
  - Poder y Control
  - Performance
  - Es de “alto” nivel
  - C++ 11 (¿C++ pyhonizado?)

Por qué C++ con Python

- ¿Por qué Python?
  - Versatilidad
  - Multiplataforma
  - Respuestas para todo
  - Rapidez de desarrollo
  - Difusión
  - Ahora con PyConVE
- ¿Por que C++?
  - Amplia cantidad de librerías
  - Flexibilidad
  - Poder y Control
  - Performance
  - Es de “alto” nivel
  - C++ 11 (¿C++ pyhonizado?)

Por qué C++ con Python

Finalmente es un hecho irrefutable que el mundo está escrito sobre C y C++.

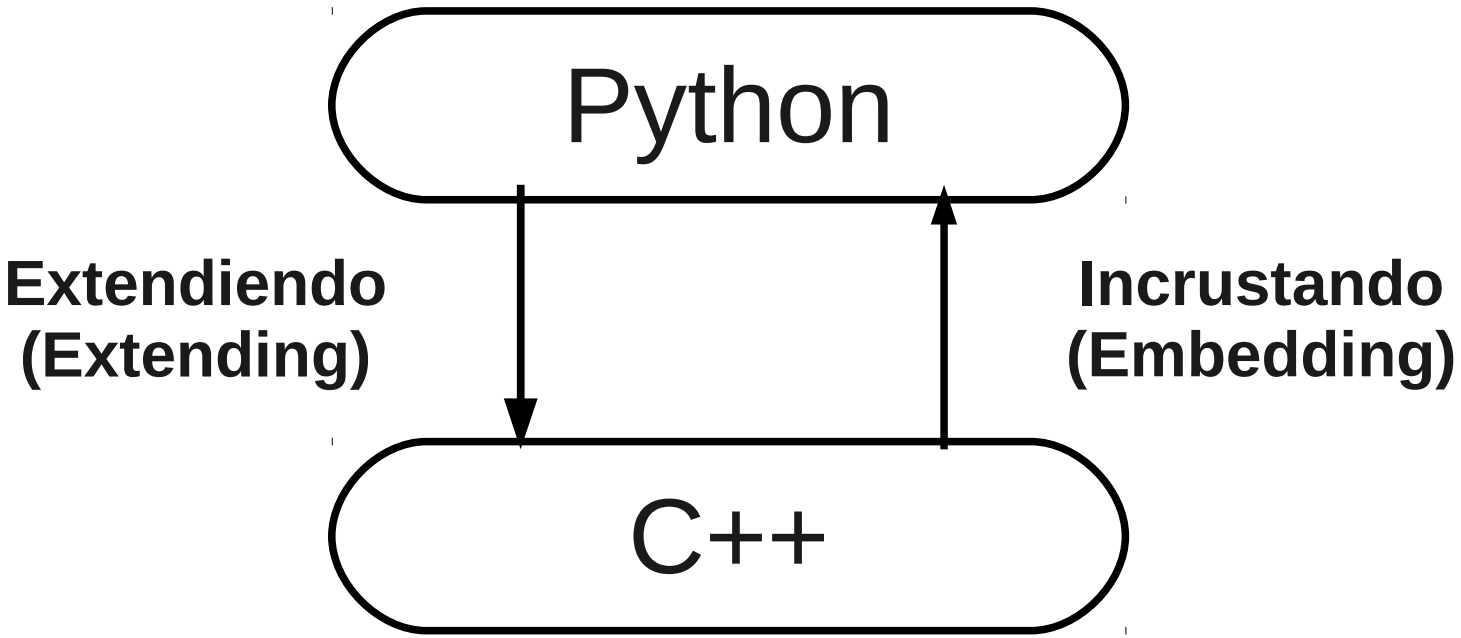
Por qué C++ con Python

Finalmente es un hecho irrefutable que el mundo está escrito sobre C y C++.

Interoperar y generar interfaces con estos lenguajes no solo es conveniente sino deseable.

Opciones de Integración

Opciones de Integración



#include “Python.h”

- Usos
  - Implementar módulos de Python
  - Agregar el intérprete de Python a su aplicación C++
- Bueno para
  - Optimizar módulos
  - Agregar Scripting Python a tu App
- Para tomar en cuenta
  - Pasarás mucho rato escribiendo Py.. y _Py.. dentro de C y C++
  - Se debe tomar en cuenta restrcciones en cuanto a el acceso a objetos static o global.

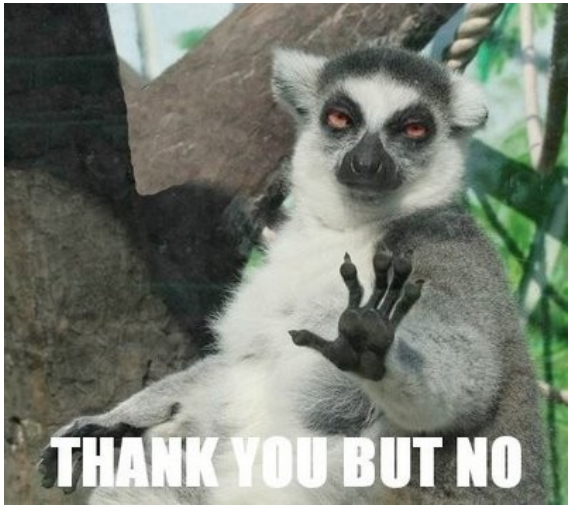
#include “Python.h”

- Más información en
  - http://docs.python.org/2/c-api/
  - http://docs.python.org/2/extending/

Cython

- Cython != Python
- Basado en PyRex
- Cython permite agregar “sabor a C” a Python
- Luego de la versión 0.13 Cython da soporte nativo a C++

- No es necesario escribir un wrappers
- Pero..



..es otro lenguaje.

Swig

- Es tal vez la forma más estándar de compartir librerías con otros lenguajes.
- Crea interfaces para para acceder a las librerías.
- Ideal para compartir librerías preexistentes
- En la mayoría de los casos no requiere modificar el código original.

Swig

- Bueno para:
  - Facilitar tareas (UI, testing, customize & reconfigure)
  - Incorporando C/C++ a un lenguaje de mas alto nivel resulta en mayor productividad, flexibilidad, menos código, etc.
  - La programación C++ se hace más deseable (o soportable)

Swig

- Usos para:
  - Más Flexibilidad
    - Reemplaza main() por una versión pythonica en esteroides.
  - Acelerar las Pruebas
    - Probando tu librería C++ con una serie de scripts
    - Usando el intérprete como debugger
  - Integrar sistemas
  - Crear módulos de alto “performance”

Swig

- Ejemplo rápido

```
bash$ cat erf.i

%module erf
#include
double erf(double);

bash$ swig -o erf_wrap.c -python erf.i
bash$ gcc -o erf_wrap.os -c -fPIC -I/usr/include/python2.4 erf_wrap.c
bash$ gcc -o _erf.so -shared erf_wrap.os
bash$ python
>>> from erf import erf
>>> erf(1)
0.8427007929497149
```

http://www.johndcook.com/blog/2009/01/20/using-swig-to-expose-c-code-to-python/

Boost

- Boost trae C++ 11 al presente.
- Similar a Swig pero más orientado a C++ y casado con Python
- A diferencia de Swig no requiere de un lenguaje IDL adicional.
- Puede ser utilizado tanto para Extender Python como para Incrustar en C++

Boost

- Bueno para:
  - .. para todo lo anterior, pero con mejoras significativas como:
    - Soporte de funciones virtuales para ser sobreescritas en Python
    - Set completo de herramientas para el manejo de ciclo de vida de bajo nivel en referencias y punteros
    - Soporte para organizar extensiones como paquetes Python.
    - Mecanismos seguros para serialización C++/Python
    - Coherencia con las reglas de los manejos de “lvalues and rvalues” en C++

Boost

- Para tener en cuenta:
  - Con Boost no requiere otra herramienta más que su compilador C++ favorito.
  - Python y C++ llaman a las mismas cosas de maneras distintas. Prepare su tabla de equivalencias
  - Python y C++ tienen formas distintas de operar sus elementos. No traduzca muy literalmente las interfaces

Boost

- Un “Hola Mundo”

```
#include <boost/python/module.hpp>
#include <boost/python/def.hpp>

char const* greet()
{
   return "hello, world";
}

BOOST_PYTHON_MODULE(hello_ext)
{
    using namespace boost::python;
    def("greet", greet);
}
```

```
$ python
Python 2.6.6 (r266:84292, Dec 27 2010,
00:02:40)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or
"license" for more information.
>>> import hello_ext
>>> print hello.greet()
hello, world
>>>
```

Otros

- “I see death projects”
  - ScyPy Weave
  - PyInline
  - Pyrex
- Estos proyectos tienen al menos 2 años de inactividad

A modo de conclusión

- Es necesario conocer la Python/C API aun cuando no haga uso de ella
- Swig es la solución más popular, por lo que tiene mejor documentación y más ejemplos
- Boost es una mejora importante para ambientes C++ con Python, pero requiere conocer mejor C++.

A modo de conclusión

- A menos que sea necesario evite agregar más elementos a su ecuación de integración.
- Nunca optimice antes de tiempo.
- Python y C++ son amigos

Este fue el qué..

..para saber el cómo participe
en el taller del sábado.

Gracias por su tiempo..