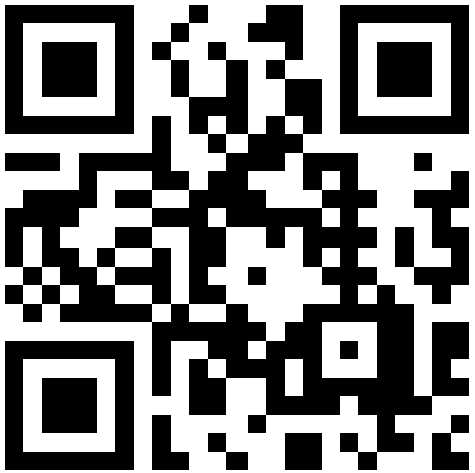
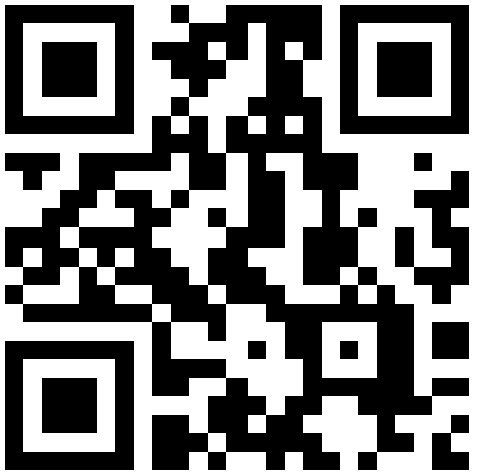
'Embedding' de Python en otras aplicaciones



Jesús Cea Avión
jcea@jcea.es
@jcea
httpS://www.jcea.es/
httpS://blog.jcea.es/



Jesús Cea Avión

- Programando Python desde 1996 (Python 1.4).
- *Core Developer* desde 2008 (Python 2.6 y 3.0).
- Fundador de Python Madrid y Python España, instrumental para la creación de Python Vigo.
  - Miembro de las dos Juntas Directivas.
  - Listas de correo, repositorio Mercurial, calendario de eventos, Twitter, OpenBadges.
- Consultor y *Freelance* a tu disposición.

Visión Global

- ¿Por qué integrar Python en otros programas?
  - Rendimiento del programador.
  - Gestión de errores.
  - Expresividad.
  - Acceso a un mundo de recursos y librerías.
- Pero ojo con:
  - Rendimiento.
  - Adaptación de impedancias entre ambos contextos.

Taxonomía

- Extender Python mediante librerías en otros lenguajes: C, Cython, bindings a librerías ajenas.
- Extender otros programas integrando en su interior un intérprete de Python:
  - El programa dispone de un sistema de plugins:
    - El propio programa incluye un intérprete Python.
    - Integramos un intérprete en un plugin.
  - NO hay sistema de plugins:
    - Inyección de código.
    - Interposición.

Plugins Python nativos

- Enlazamos con la librería Python.
- Inicializamos el intérprete.

  Py_Initialize()
- Ejecutamos código simple:

  PyRun_SimpleString(), PyRun_SimpleFile()
- Creamos adaptaciones entre el API C y Python, en ambos sentidos. Puede ser una tarea incremental.
- ¡Escribimos plugins en Python!

Ejemplo: Olimpo

- Nodo de control para ESNET e IRC-Hispano. 1997-2002.
- Inicialmente escrito 100% en C, monolítico.
- En 2000, versión modular a base de plugins en C.
- En 2001 se integró Python 2.0, plugins en Python.
- Código liberado como AGPL3 en enero de 2013.

  http://hg.jcea.es/IRC/olimpo/file/tip/
- Buena documentación.

Olimpo: Inicialización

```
void inicializa_python(void)
{
[...]
  if (!Py_IsInitialized()) {
    Py_Initialize();
    PyEval_InitThreads();
    mainThreadState = PyThreadState_Get();
  }
[...]
  olimpo = PyImport_AddModule("Olimpo");
  Py_InitModule("Olimpo", olimpo_methods);
  olimpo_dict = PyObject_GetAttrString(olimpo, "__dict__");
  olimpo_mod = PyImport_AddModule("Olimpo.privmsg");
  Py_InitModule("Olimpo.privmsg", privmsg_methods);
  PyMapping_SetItemString(olimpo_dict, "privmsg",olimpo_mod);
[...]
```

Olimpo: Definición de métodos

```
static PyMethodDef olimpo_methods[] = {
  {"especifica_fin", (PyCFunction)
olimpo_methods_especifica_fin, METH_O},
  {"comentario_modulo", olimpo_methods_comentario_modulo,
METH_VARARGS},
  {"hace_log", olimpo_methods_hace_log, METH_VARARGS},
  {"hace_log2", olimpo_methods_hace_log2, METH_VARARGS},
  {"debug", (PyCFunction) olimpo_methods_debug, METH_NOARGS},
  {NULL, NULL, 0}                 /* sentinel */
};

static PyMethodDef privmsg_methods[] = {
  {"nuevo_nick", privmsg_methods_nuevo_nick, METH_VARARGS},
  {"quit_nick", privmsg_methods_quit_nick, METH_VARARGS},
  {"envia_nick", privmsg_methods_envia_nick, METH_VARARGS},
[...]
  {NULL, NULL, 0}                 /* sentinel */
};
```

Olimpo: Un módulo de ejemplo

https://www.jcea.es/irc/modulos/saluda.htm

```
# $Id: saluda.py,v 1.4 2002/09/17 14:03:39 jcea Exp $

import Olimpo

def privmsg(nick, remitente, mensaje):
  flags = Olimpo.privmsg.lee_flags_nick(remitente)
  nick_u = Olimpo.privmsg.lee_nick(remitente)
  ip = Olimpo.privmsg.lee_ip_nick(remitente)
  host = Olimpo.privmsg.lee_host_nick(remitente)
  Olimpo.privmsg.envia_nick(nick,remitente,"Hola, %s, …
  Olimpo.privmsg.envia_nick(nick,remitente,"Tu IP es …

def inicio():
  Olimpo.comentario_modulo("Modulo PYTHON de ejemplo
$Revision: 1.4 $")
  Olimpo.privmsg.nuevo_nick("saluda","+odkirhB",privmsg)
```

Olimpo: Registro de nuevo nick (1)

```
static PyObject *privmsg_methods_nuevo_nick(PyObject * self,
PyObject * args)
{
  char *nick, *modos;
  PyObject *funcion;
  int ok;
  int result;

  ok = PyArg_ParseTuple(args,"ssO", &nick, &modos, &funcion);
  if (!ok)
    return NULL;
  if (!PyCallable_Check(funcion)) {
    PyErr_SetString(PyExc_TypeError,
 "El objeto pasado a 'nuevo_nick' no es una funcion valida");
    return NULL;
  }
  Py_XINCREF(funcion);
  result = nuevo_nick(nick, modos, (void *) funcion);
  return PyInt_FromLong(result);
}
```

Olimpo: Registro de nuevo nick (2)

```
void envia_privmsg_modulo(char *destino, char *remitente,
char *mensaje)
{
[...]
    n = modulo_actual->nicks;
    while (n) {
      if (n->privmsg && (n->handle == handle)) {
        if (modulo_actual->python) {
          modulo_python_privmsg(modulo_actual->python,
                      n->privmsg, handle, handle2, mensaje);
        }
        else {
          n->privmsg(handle, handle2, mensaje);
        }
        return;
      }
      n = n->siguiente;
    }
[...]
```

Olimpo: Registro de nuevo nick (3)

```
void modulo_python_privmsg(void *modulo_python, void
*privmsg, int handle, int handle2, char *mensaje)
{
  PyObject *result;
[...]
  ENTER_PYTHON;
  result = PyObject_CallFunction(privmsg, "iis", handle,
                                          handle2, mensaje);
  if (!result)
    modulo_excepcion();
  Py_XDECREF(result);
  EXIT_PYTHON;
[...]
}
[...]
static void modulo_excepcion(void)
{
  PyErr_Print();
}
```

Plugins en C

- Escribimos un plugin en C que lo único que hace es inicializar un intérprete de Python

  Py_Initialize()
- Ejecutamos código simple:

  PyRun_SimpleString(), PyRun_SimpleFile()
- Creamos adaptaciones entre el API C y Python, en ambos sentidos. Puede ser una tarea incremental.
- ¡Escribimos plugins en Python!

Ejemplo: mod_python, mod_wsgi

- Permite extender Apache HTTP mediante código Python.
- Permiten acceder a *hooks* a bajo nivel, como autenticación o grabación de logs.
- mod_wsgi proporciona un *framework* wsgi nativo.
- mod_python permite interactuar con Apache HTTP a muy bajo nivel, creando filtros, reescrituras de peticiones web, etc. Desarrollo poco activo.
- No son compatibles en el mismo servidor.

No hay sistema de plugins

- Podemos usar interposición o inyección.
- En interposición, escribimos una librería dinámica en C que lo único que hace es inicializar un intérprete de Python e interponerse a una ***SO***.

  Py_Initialize(), ***LD_PRELOAD***
- Ejecutamos código simple:

  PyRun_SimpleString(), PyRun_SimpleFile()
- Creamos adaptaciones entre el API C y Python, en ambos sentidos. Puede ser una tarea incremental.
- ¡Escribimos plugins en Python!

Ejemplo: interposición (1)

```
#include <Python.h>
[...]
#include <dlfcn.h>
#include <stdarg.h>

static int initialized = 0;
static int (*open_original)(const char *pathname, int
flags, ...) = NULL;

static void init(void) __attribute__((constructor));
static void init(void)
{
    open_original = dlsym(RTLD_NEXT, "open");
    Py_Initialize();
    PyRun_SimpleString("import intercept");
[... Inyectamos el acceso a 'open' con ctypes ...]
    initialized = 1;
}
```

Ejemplo: interposición (2)

```
int open(const char *pathname, int flags, ...)
{
[...]
    va_start(valist, flags);
    mode = va_arg(valist, mode_t);
    va_end(valist);
    if(!initialized) {
        return open_original(pathname, flags, mode);
    }
    /* XXX: BUFFER OVERFLOW, ESCAPING y rendimiento! */
    sprintf(buf, "intercept.do_open('%s', '%d', '%d')",
                                        pathname, flags, mode);
    result = PyRun_String(buf, Py_file_input, globals,
                                locals);
    if (!result) {
        PyErr_Print();
        return -1;
    }
    return (int)PyLong_AsLong(result);
}
```

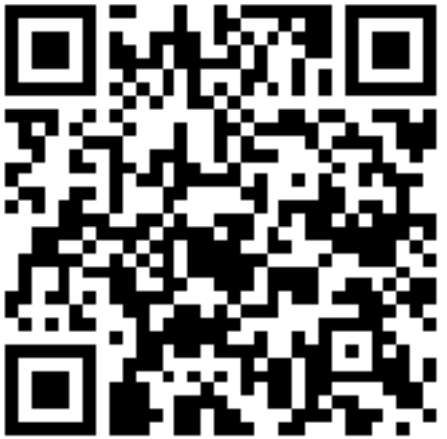
Ejemplo: interposición (3)

- Compilamos con:

```
$ gcc -I/usr/local/include/python3.5m/ -shared \
  -o interposicion.so -fPIC interposicion.c -lpython3.5m
```

- Ejecutamos con:

```
$ LD_PRELOAD=/tmp/z/interposicion.so EJECUTABLE
```



- Referencias:

https://stackoverflow.com/questions/9759880/
https://blog.jcea.es/posts/20150509-ld_reload_e_interposicion.html

Ejemplo: posibilidades...

```
def do_open(fichero, flags, modo) :
    print(fichero)
    return open_original(fichero, flags, modo)

def do_open(fichero, flags, modo) :
    if fichero not in permitidos :
      return -1    # Ojo, errno
    return open_original(fichero, flags, modo)


def do_open(fichero, flags, modo) :
    if fichero in acceso_remoto :
      return abrir_sesion_remota(fichero).fd

def do_read(fd, size) :
    if fd in sesiones_remotas :
        return sesiones_remotas[fd].read(size)
    return read_original(fd, size)
```

Ejemplos reales en programas propietarios:

- Reemplazo del algoritmo de “loudness” para la normalización de un audio (podcast).
- Reemplazo de funciones matriciales para usar PyCUDA (rendimiento x2.5 en un proceso que necesita una semana de cálculo) (machine learning).
- Análisis y emulación de un dispositivo mapeado como fichero en “/dev/”.

¿Preguntas?

- Multithreading.
- Subintérpretes.
- Múltiples intérpretes simultáneos, Python 2 y 3.
- Recarga de módulos.
- Rendimiento (uso de cython).
- Gestión de errores.
- ¡Unicode!
- Seguimiento de la evolución del API.
- ¿C++ y otros lenguajes?

Referencias adicionales

- Extending and Embedding the Python Interpreter:

  https://docs.python.org/3.5/extending/index.html

- Python/C API Reference Manual:

  https://docs.python.org/3.5/c-api/

- Proyecto OLIMPO:

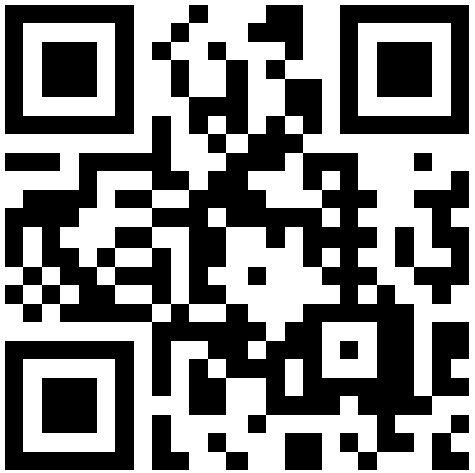
  https://www.jcea.es/irc/olimpo1.htm

  http://hg.jcea.es/IRC/olimpo/file/tip/

- mod_wsgi:

  https://modwsgi.readthedocs.org/

¡Gracias!



Jesús Cea Avión
jcea@jcea.es
@jcea
httpS://www.jcea.es/
httpS://blog.jcea.es/