El lenguaje de programación Python

Diego Martín

Universidad de Salamanca

martinarroyo@usal.es

11 de abril de 2014

Contenidos

The Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one– and preferably only one –obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.

Historia

Concebido en los años 90 por Guido Van Rossum como un sucesor de ABC, un lenguaje de propósito general pensado para la enseñanza y el prototipado. En el año 2000 aparece Python 2.0 El proyecto sigue siendo orquestado por Guido, lo que le ha hecho merecedor del título de *Benevolent Dictator for Life*.

Conceptos clave

- Python es un lenguaje multi-paradigma (oo y programación estructurada)
- Tipado dinámico y gestión de la memoria automática
- Es legible
- Es portable
- Es integrable
- Tiene que ser divertido de usar

Nobody expects the Python intermission!

¿Para qué sirve Python?

¿Para qué sirve Python?

Ahá, así que vas a trabajar con algo que nadie sabe muy bien para qué sirve.

Vas a ver que es muy sencillo de utilizar

No conocía ese lenguaje

¿Para qué sirve Python?

- Prototipado
- Interfaces Gráficas de Usuario
- Administración de sistemas
- CGI
- Scripting

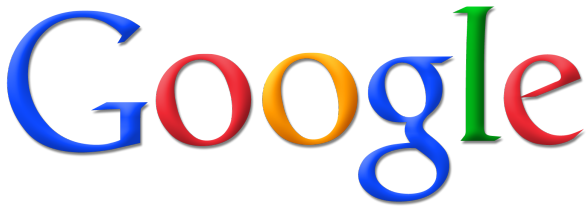
Ventajas de Python

- Es un lenguaje orientado a objetos *opcionalmente*
- Es libre
- Tiene un gran soporte
- Gran espíritu de comunidad
- Portabilidad
- Potencia
- Combinable con otros lenguajes
- Fácil de utilizar y de aprender
- Sintaxis clara

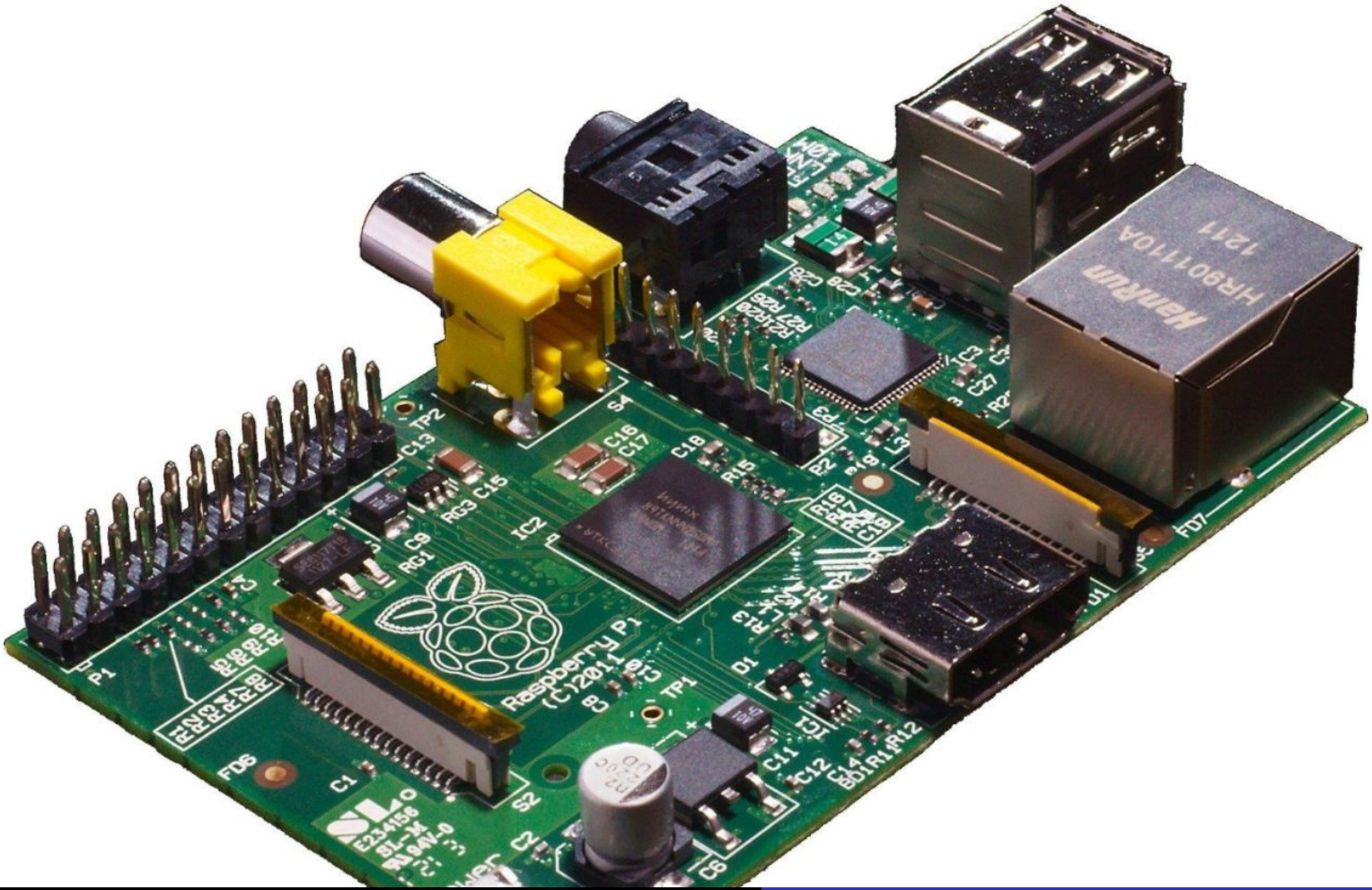
Desventajas de Python

Rendimiento. Al ser un lenguaje interpretado, su potencia es menor que la alcanzada con otros como C. Sin embargo, gracias a la traducción del código a un *bytecode* intermedio se ha mejorado notablemente.

En adición a los 500,000 usuarios del lenguaje, muchas compañías utilizan el lenguaje en servicios en red y para pruebas de hardware.



La Pi de Raspberry Pi



Números

Aporta mucha flexibilidad

- Enteros
- Reales
- ¡Complejos!
- El límite lo marca la memoria del equipo

Operaciones con números

Cualquier operación soportada en C, en adición a nuevas como la suma con truncado (//). Es capaz de trabajar con varias bases de forma natural. La conversión entre números de distinta precisión es 'hacia arriba' a menos que se especifique lo contrario.

Cadenas de texto

En Python son un tipo de secuencia. El lenguaje cuenta con un potente conjunto de utilidades para la manipulación de cadenas. Son elementos inmutables por lo que una vez creados no se pueden modificar. Trabaja con varios tipos de codificación de caracteres.

- Búsqueda por índices: cadena[i]
- Iteración: for x in cadena
- Métodos: cadena.capitalize(), cadena.center(), isalnum()...

Gestión de la memoria

Toda variable en Python almacena una referencia a un dato en la memoria. La gestión de esta se realiza mediante un contador de referencias.

Importante: Si bien las variables no tienen un tipo, los datos almacenados sí, y la conversión no es automática.

Tercera categoría: mapas..
Destacan las listas y los diccionarios.

Listas

Una lista es una colección de elementos ordenados muy flexible. No importa el tipo de objeto que almacenen (son colecciones de referencias), e implementan muchas de las operaciones que en otros lenguajes son responsabilidad del programador.
Son mutables, de longitud variable y orden definido.
Ejemplos de listas:

- L1 = [];
- L2 = ['abc', 'is', 'as', 'easy', 'as'];
- L2.append('123');

Diccionarios

Los diccionarios son el tipo más flexible integrado en el lenguaje. Son colecciones de elementos sin orden clasificados por entradas clave-valor. Son mapas mutables.

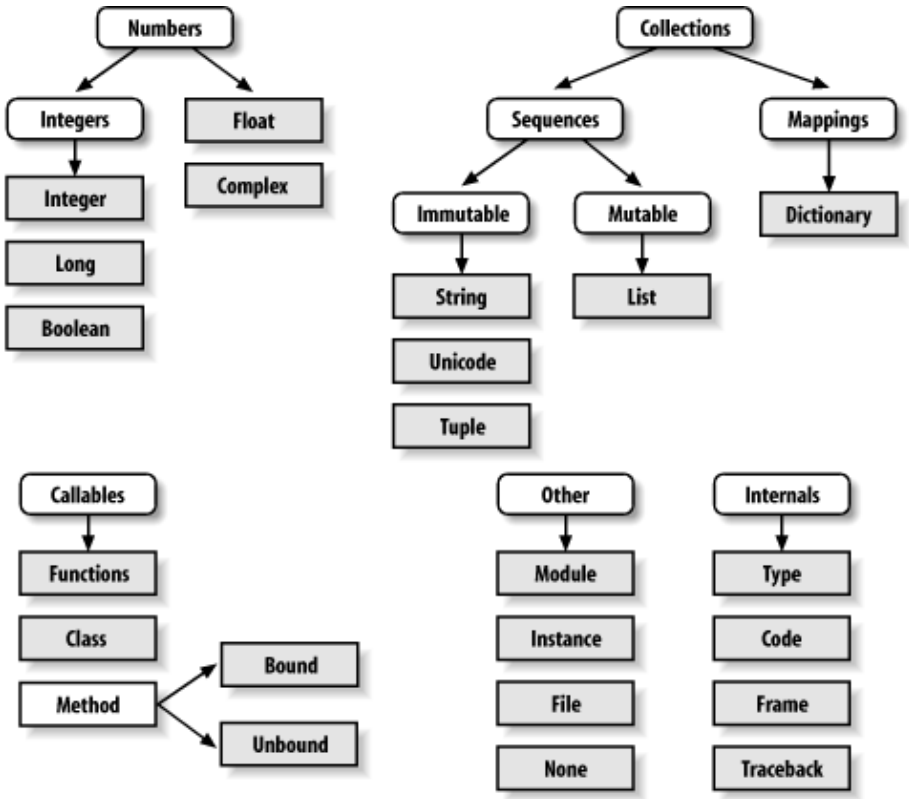
Trabajo con índices

Trabajar con índices en Python es mucho más versátil que en otros lenguajes mediante la adición de operadores nuevos:

- array[1:]
- array[:4]
- array[1:3]
- array[-1]
- array[:]

Para qué sirve tener tipos inmutables y mutables?

La principal ventaja de contar con ambos tipos de elementos es la flexibilidad de unos y la garantía de *inmovilidad* de los otros.



- if, elif, else
- while añade operadores como else

Toda función en Python genera una referencia que se almacena en un nombre. Ejemplo:

def correr(): global velocidad; print .^Estoy corriendo a %d metros/segundo" % velocidad

Módulos

Un módulo de Python es un paquete de código que permite aumentar la funcionalidad de programas mediante la reutilización de piezas de otros. A grandes rasgos constituyen paquetes de nombres que son cargados (y compilados sin es necesario) en memoria según la funcionalidad que se desee aprovechar de ellos. Fomentan la reusabilidad del código, particionan el espacio de nombres e implementan servicios compartidos o datos.
Se utilizan las sentencias **import** o **from** para añadir módulos al programa.

Al ser un lenguaje multiparadigma, la utilización de objetos o no es una decisión delegada al programador. Sin embargo es altamente recomendable utilizar la orientación a objetos debido a los beneficios que supone.

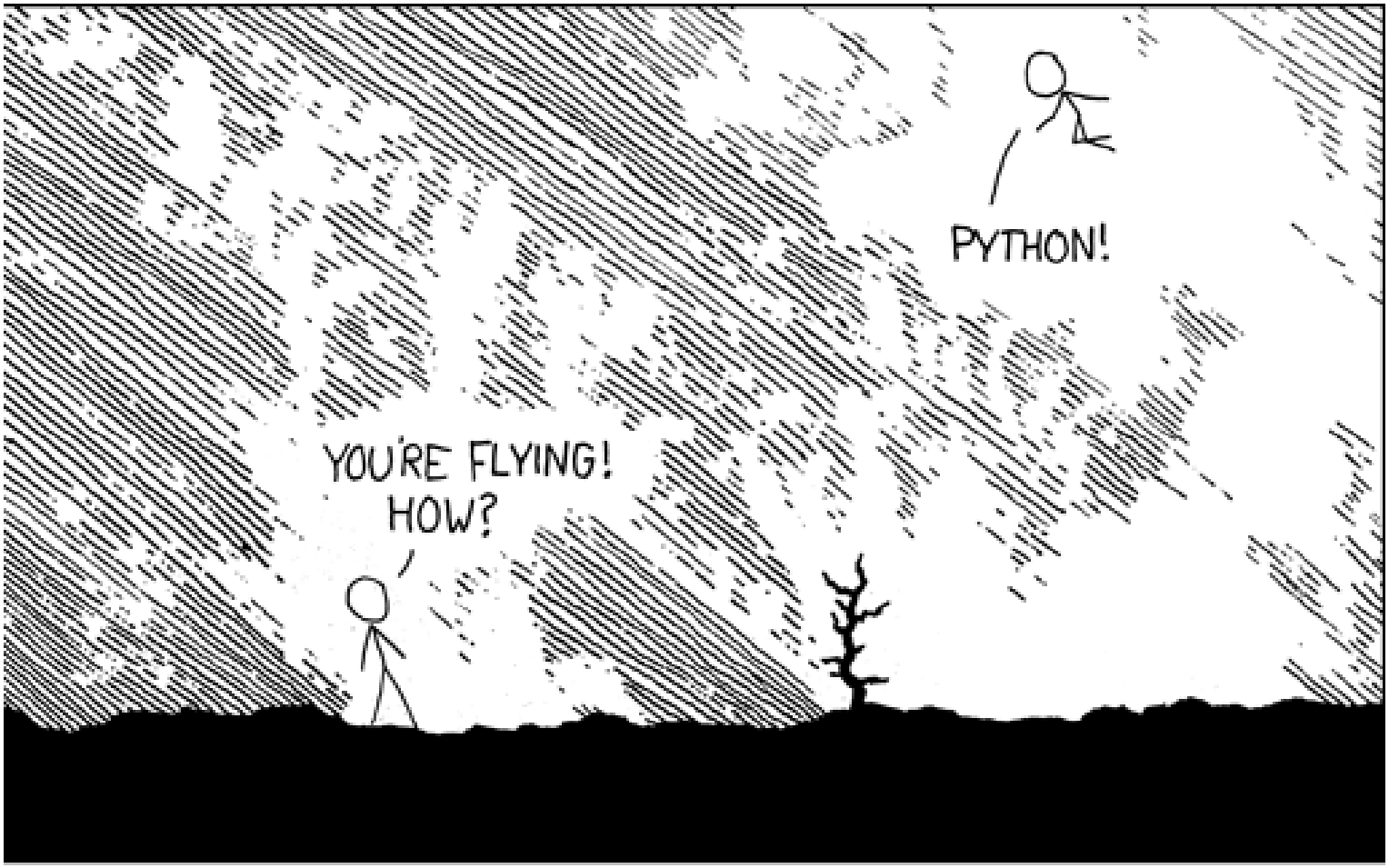
Ejemplo de una clase

```
class Perro:
def Ladrar():
print 'Arrf!'
def Correr():
print 'Estoy corriendo'
```

Las excepciones son mecanismos de control de situaciones irregulares dentro del flujo de un programa. Consisten en bloques de código rodeados por las palabras **try**, **except** y **finally**. Se capturan tanto excepciones incluidas en los módulos estándar de Python como aquellas excepciones definidas por el usuario (utilizando **raise**).

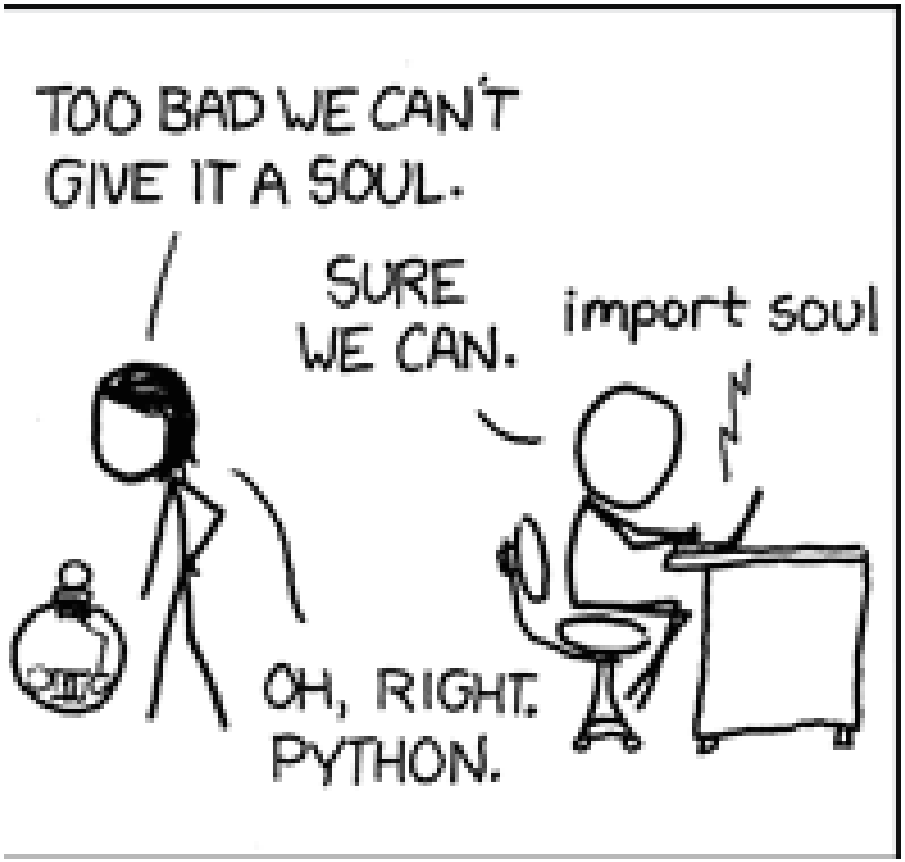
Demo time!

Impacto en el mundo del desarrollo de software.



Impacto en el mundo del desarrollo de software.

Impacto en el mundo del desarrollo de software.



The End