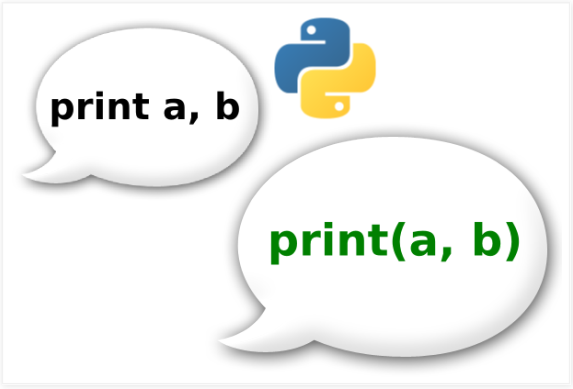
# Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)



**miércoles, 28 de enero de 2015**

## Convertir programas Python 2.x a 3.x




***2to3*** es una herramienta Python que convierte programas escritos en código Python 2.x a 3.x; manteniendo todos los comentarios y el sangrado existentes en el código original.

Dicho programa está basado en el módulo **lib2to3** que ofrece cierta flexibilidad para los programadores al permitir personalizar el modo en que se traduce el código de los programas.

El programa ***2to3*** se ejecuta como un script Python y se encuentra, generalmente, en el directorio Tools\Scripts dentro de una instalación Python.

En Linux podemos conocer la ubicación del script ***2to3*** con:

$ **which 2to3**

Si está en el sistema nos mostrará su ruta de instalación. Ejemplo:

**/usr/bin/2to3**

También, si tenemos curiosidad podemos mostrar el contenido del script ***2to3*** desde la línea de comandos:

**$ cat /usr/bin/2to3**

*2to3*:

```
#! /usr/bin/python2.7
import sys
from lib2to3.main import main

sys.exit(main("lib2to3.fixes"))
```

### Mostrar diferencias del código a convertir

El siguiente programa está escrito con Python 2.x (saludo.py):

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def saludo(nombre, ciudad):
    print "Hola {} y vives en {}".format(nombre, ciudad)

print "Escribe tu nombre:"
nombre = raw_input()
print "Escribe el nombre de tu ciudad:"
```

**Buscar**

Buscar

**Python para impacientes**

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

**Anexos**

Guía urgente de MySQL
Guía rápida de SQLite3

**Entradas + populares**

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario ( GUI ) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones

```
ciudad = raw_input()
saludo(nombre, ciudad)
```

Para mostrar las diferencias que tendría el código al convertirlo a 3.x, ejecutaremos desde la línea de comandos:

$ **2to3 saludo.py**

Resultado:

```
...

RefactoringTool: Refactored saludo.py
--- saludo.py (original)
+++ saludo.py (refactored)
@@ -2,10 +2,10 @@
 # -*- coding: utf-8 -*-

 def saludo(nombre, ciudad):
-    print "Hola {}, vives en {}".format(nombre, ciudad)
+    print("Hola {}, vives en {}".format(nombre, ciudad))

-print "Escribe tu nombre:"
-nombre = raw_input()
-print "Escribe el nombre de tu ciudad:"
-ciudad = raw_input()
+print("Escribe tu nombre:")
+nombre = input()
+print("Escribe el nombre de tu ciudad:")
+ciudad = input()
 saludo(nombre, ciudad)
RefactoringTool: Files that need to be modified:
RefactoringTool: saludo.py
```

### Traducir el código de Python 2.x a 3.x

Para hacer efectiva la conversión del código Python de 2.x a 3.x y realizar una copia de seguridad del fuente original, ejecutaremos desde la línea de comandos:

$ **2to3 -w saludo.py**

Cuando finalice la conversión tendremos dos archivos: **saludo.py** y **saludo.py.bak**. El primero de ellos contendrá el código traducido a Python 3.x y el .bak el código original. Para descartar, en un momento dado, la copia de seguridad tendremos que añadir la opción -n.

Código obtenido en Python 3.x (saludo.py):

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def saludo(nombre, ciudad):
    print("Hola {} y vives en {}".format(nombre, ciudad))

print("Escribe tu nombre:")
nombre = input()
print("Escribe el nombre de tu ciudad:")
ciudad = input()
saludo(nombre, ciudad)
```

Por defecto, se aplicarán en la traducción un conjunto de reglas que pueden mostrarse con la opción -l:

$ **2to3 -l**

### Aplicar o excluir reglas específicas de traducción

Si queremos aplicar sólo algunas reglas de las disponibles las especificaremos mediante la opción -f. En cambio, para excluirlas del rastreo utilizaremos la opción -x.

Como se puede comprobar, en el código de ejemplo anterior (**saludo.py**) se han traducido sólo los comandos **raw_input()** y **print** por **input()** y **print()**, respectivamente. El resto no ha sufrido ningún cambio. Si queremos ganar tiempo en la traducción podemos rastrear y traducir específicamente estos comandos y omitir al resto con:

$ **2to3 -w -f raw_input -f print saludo.py**

que permiten obtener de distintos modos números a...

**Archivo**

enero 2015 (2)

**python.org**

**pypi.org**

**Sitios**

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

Pero ***2to3*** no es infalible. A veces, podemos encontrarnos con código que no puede traducirse automáticamente. En este caso, ***2to3*** nos mostrará una advertencia que con posterioridad debemos solucionar editando y modificando el código manualmente para hacerlo compatible con 3.x, y que no tengamos problemas al ejecutarlo.

La opción -v permite la salida de más información relacionada con el proceso de traducción. Para consultar más opciones utilizar la opción -h.

### Conversión de módulos de terceros

El script ***2to3*** puede resultarnos de mucha utilidad para convertir módulos de terceros que fueron desarrollados en Python 2.x y que todavía no han sido convertidos a 3.x; y que, de vez en cuando, nos encontramos en repositorios como PIP.

Ir al índice del tutorial de Python

Publicado por Pherkad en 14:23

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua