#1

UNDERCODE

TALLER DE PYTHON 3

TEMAS

INTRODUCCIÓN
HOLA MUNDO EN PYTHON 3
CÓDIGO FUENTE Y BYTECODE
HERRAMIENTAS DE DESARROLLO
EDITORES
DEPURADORES
NOVEDADES
Y MUCHO MÁS..!

TUTOR

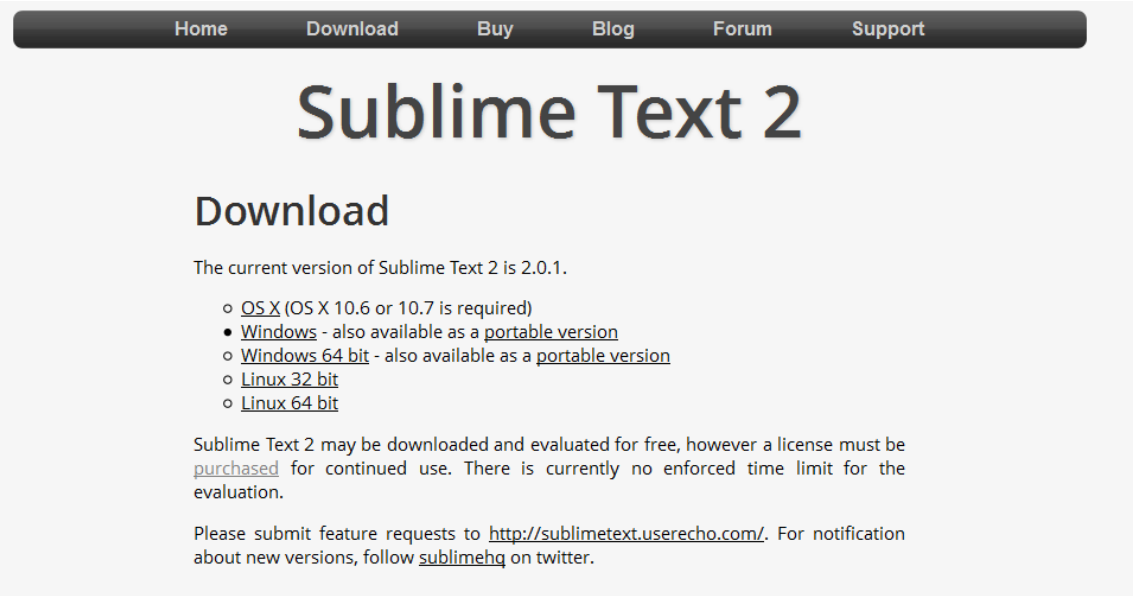
ANTRAX

Introducción

En este taller veremos las novedades sobre Python 3, que cosas nuevas e interesantes trae, que lo diferencia con Python 2, a demás conoceremos los depuradores que trae. Miraremos por arriba lo que es profiling, y de que manera usarlo para medir el rendimiento de la ejecución de algún script programado en Python

Este taller estará orientado a aquellas personas que no han visto jamás Python y también a aquellos que quieran pasar de Python 2 al 3

A demás de esto, usaremos un editor de texto, cada uno puede utilizar el que le resulte más cómodo, yo usare Sublime Text 2 http://www.sublimetext.com/2



Al intérprete lo descargamos desde acá: http://www.python.org/download/

Start with one of these versions for learning Python or if you want the most stability; they're both considered stable production release

If you don't know which version to use, try Python 3.3. Some existing third-party software is not yet compatible with Python 3; if you ne such software, you can download Python 2.7.x instead.

For the MD5 checksums and OpenPGP signatures, look at the detailed Python 3.3.0 page:

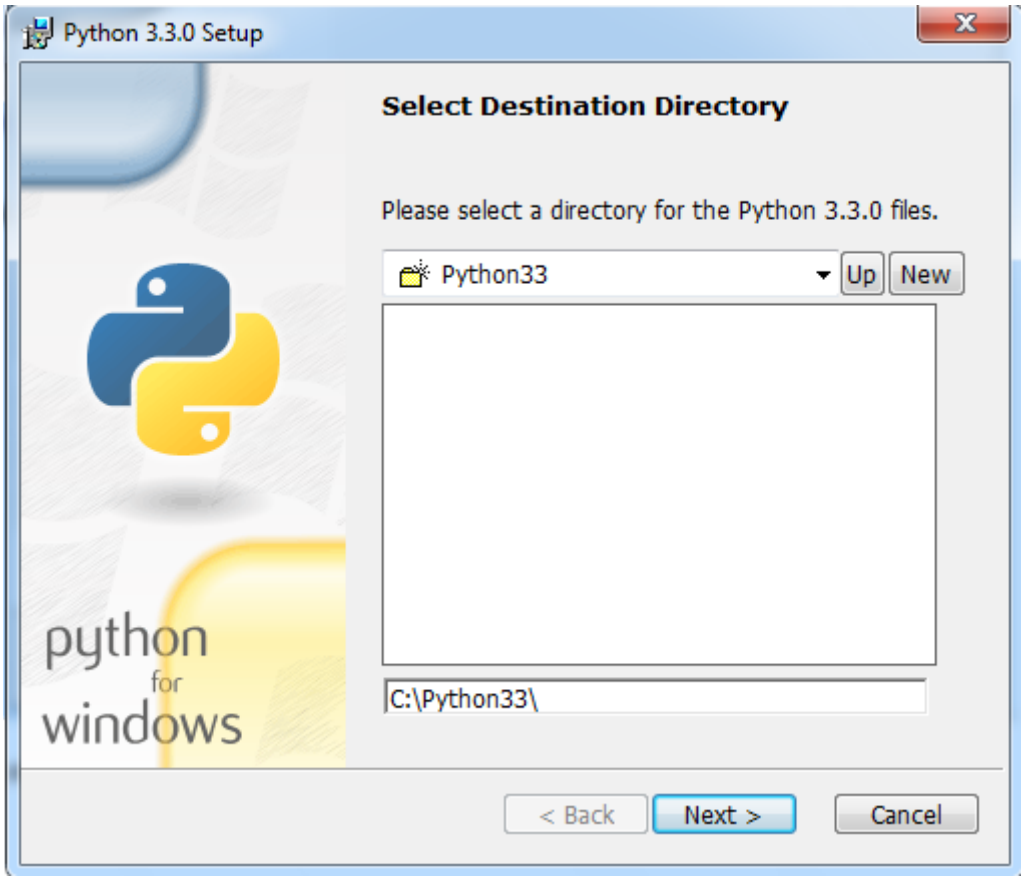
- Python 3.3.0 Windows x86 MSI Installer (Windows binary -- does not include source)
- Python 3.3.0 Windows X86-64 MSI Installer (Windows AMD64 / Intel 64 / X86-64 binary [1] -- does not include source)
- Python 3.3.0 Mac OS X 64-bit/32-bit x86-64/i386 Installer (for Mac OS X 10.6 and later [2])
- Python 3.3.0 Mac OS X 32-bit i386/PPC Installer (for Mac OS X 10.5 and later [2])
- Python 3.3.0 compressed source tarball (for Linux, Unix or Mac OS X)
- Python 3.3.0 bzipped source tarball (for Linux, Unix or Mac OS X, more compressed)

For the MD5 checksums and OpenPGP signatures, look at the detailed Python 2.7.3 page:

- Python 2.7.3 Windows Installer (Windows binary -- does not include source)
- Python 2.7.3 Windows X86-64 Installer (Windows AMD64 / Intel 64 / X86-64 binary [1] -- does not include source)
- Python 2.7.3 Mac OS X 64-bit/32-bit x86-64/i386 Installer (for Mac OS X 10.6 and later [2])
- Python 2.7.3 Mac OS X 32-bit i386/PPC Installer (for Mac OS X 10.3 through 10.6 [2])
- Python 2.7.3 compressed source tarball (for Linux, Unix or Mac OS X)
- Python 2.7.3 bzipped source tarball (for Linux, Unix or Mac OS X, more compressed)

Una vez descargado, lo instalamos

Damos click en **Next** para comenzar con la instalación

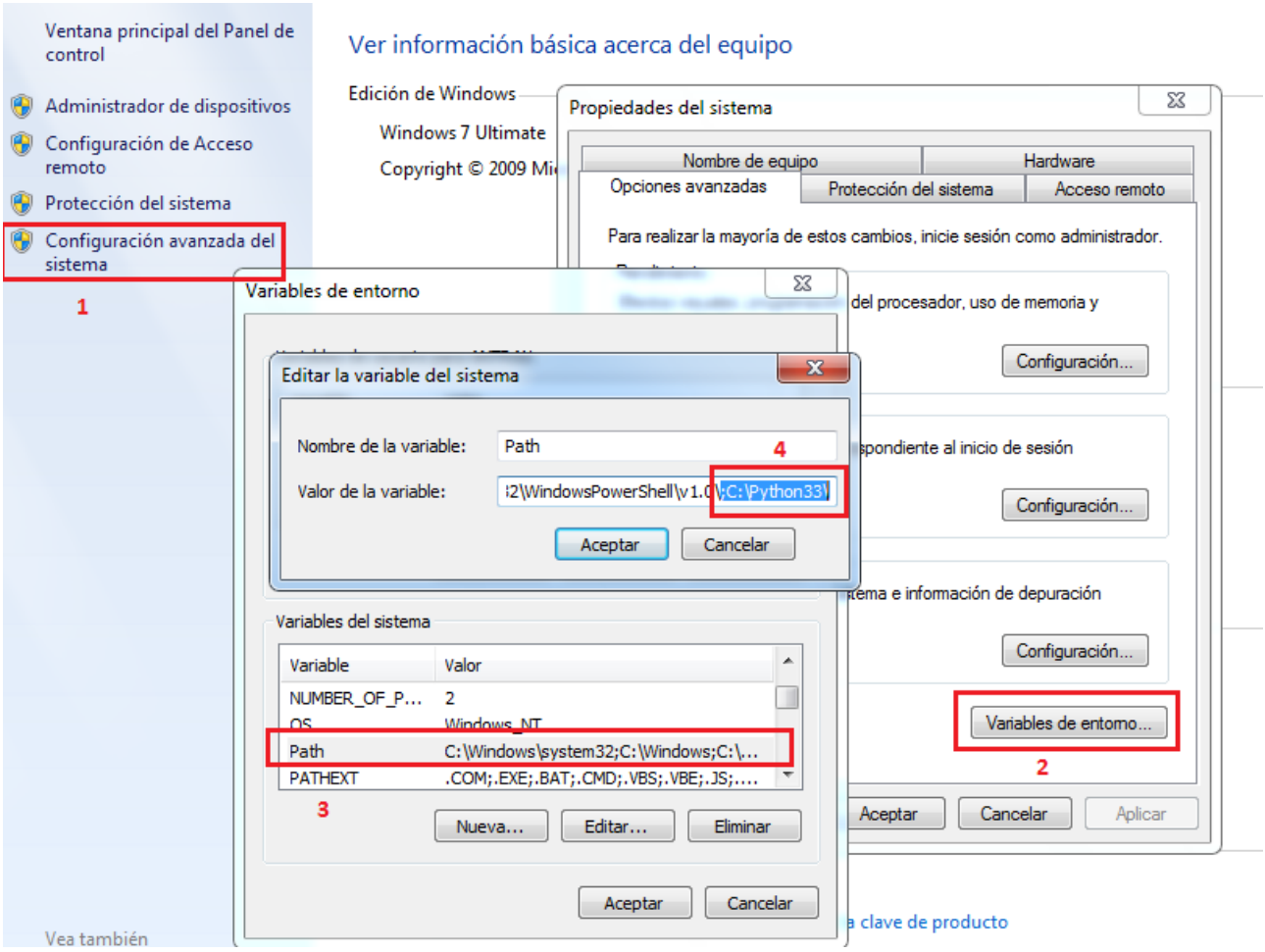


Este paso es muy importante para saber en que ruta se instalara python. En este caso es **C:\Python33**

Continuamos dando click en Next hasta que se complete la instalación.

Ahora añadiremos a Python en las variables de entorno del sistema para poder empezar a trabajar con el.

Nos dirigimos al Panel de Control y a continuación en sistema.



Los pasos de la imagen serian los siguientes:

1.- Click en la configuración avanzada del sistema

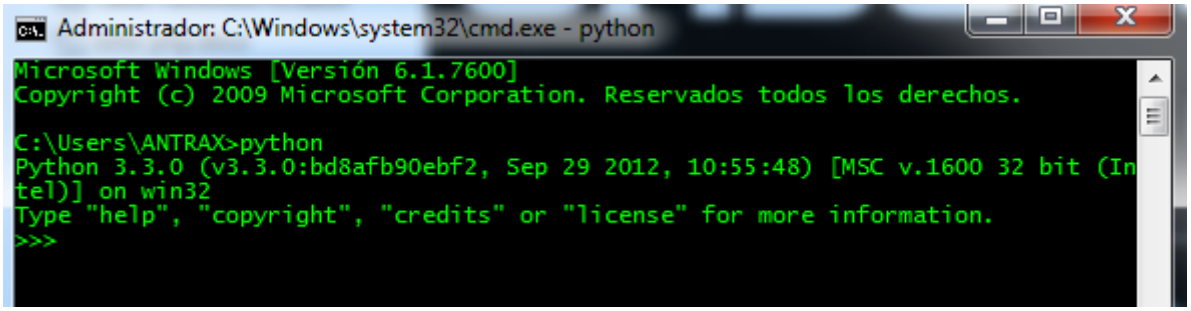
2.- Click en Variables de entorno

3.- Buscamos entre las variables del sistema a la que se llama"Path" y la editamos

4.- Agregamos la dirección en donde está instalado Python 3.

Nota: Recuerden añadir punto y coma después de la ultima ruta antes de añadir la nueva

Para verificar que todo haya quedado bien, abrimos una consola y tipiamos la palabra **python**



Veremos en la primera línea información sobre la plataforma en la que estamos trabajando, luego nos coloca comandos que nos permiten acceder a la información de Python como lo es la licencia, copyright y créditos. La última y más importante es la que comienza con **>>>** y nos muestra el cursor parpadeando, esta es el prompt del intérprete y nos permite interactuar directamente con él, es decir si tecleamos license y presionamos la tecla enter, podremos ver la licencia de python y al finalizar volverá a mostrar el prompt invitándonos a introducir otro comando o sentencia.

Hola Mundo en Python 3

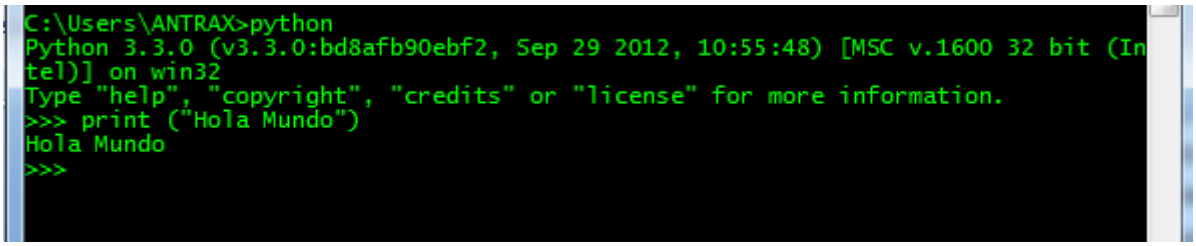
Como es tradicional cada vez que se ve un lenguaje de programación, lo primero que se hace es imprimir un hola mundo. Y acá es donde veremos la primera diferencia con Python 2

En Python 2, para imprimir un texto en pantalla, lo hacíamos de la siguiente manera:

>>> print “Hola Mundo”

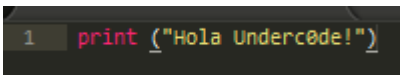
Y en Python 3 lo hacemos de esta forma:

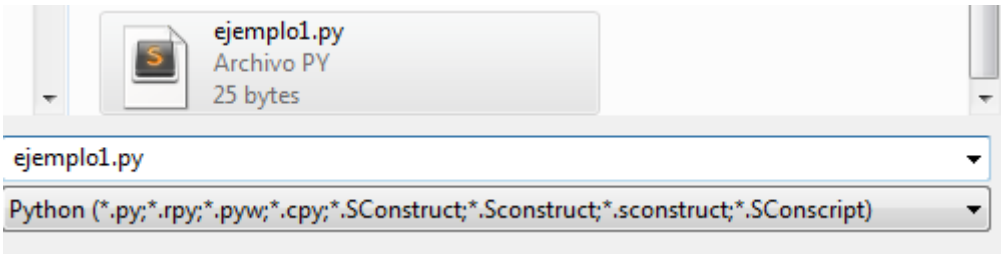
>>> print (“Hola Mundo”)



Obviamente no hace falta teclear los caracteres **>>>** ya que estos aparecen solos y nos indica que el interprete esta a la espera de que ingresemos alguna sentencia nueva.

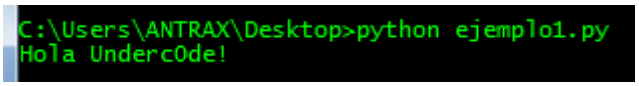
Ahora escribiremos la misma línea en nuestro editor de texto favorito y lo guardaremos con el nombre que cada uno quiera con la extensión .py que es la extensión que utiliza Python





En mi caso, lo guarde en el Escritorio con el nombre **ejemplo1.py**. Ahora es necesario abrir la consola, y ubicarnos en el directorio en donde está guardado el archivo. Una vez ubicados en el directorio del archivo, tecleamos en la consola:

python ejemplo.py



Código fuente y Bytecode

Hasta ahora solo hemos hablado de los ficheros de código Python que son los que utilizan la extensión ***.py***. También sabemos que este lenguaje es interpretado y no compilado. Sin embargo, internamente el interprete Python se encarga de generar una especie de ficheros binarios que son los que luego son ejecutados. Esto se realiza de forma oculta a partir del código fuente. Este código generado se le llama **Bytecode** y utiliza la extensión ***.pyc***. De esta forma, el intérprete de Python genera Bytecode a partir del código fuente y lo ejecuta. Este proceso se realiza por cuestiones de eficiencia.

Una vez que el fichero .pyc este generado, Python no vuelve a leer el código fuente, sino que lo ejecuta directamente y de esta forma ahorra tiempo.

El intérprete de Python es lo suficientemente inteligente como para saber cuando generar el nuevo Bytecode, y esto es cuando el código fuente es modificado. Pero de esto el programador no debe preocuparse ya que el intérprete lo hace de forma automática.

En ocasiones cuando se tiene una PC sin el intérprete, se pude generar un binario a partir de un programa en Python. Esto se realiza con programas como py2exe entre otros, cuya función es convertir el código fuente en un binario.

Herramientas de desarrollo

Editores

Podemos decir que los editores de texto son las herramientas básicas para desarrollar software, ya que nos permiten escribir el código fuente y crear un programa a partir del mismo.

Existe una gran variedad de editores de texto… Desde el clásico bloc de notas, hasta Notepad++, Sublime Text. Cualquier editor sirve para programar este lenguaje, solo que algunos son más completos, ya que resaltan sintaxis, autoidentación, navegabilidad del código, entre otras cosas que mejoran la productividad del desarrollador.

Entornos Integrados de Desarrollo (IDE)

Los IDEs amplían la funcionalidad de los editores de texto, ya que estos permiten depurar código, crear proyectos, el autocompletado, búsqueda de referencias en la documentación o marcado de sintaxis errónea. Dos de los más conocidos son NetBeans y Eclipse, que aunque últimamente se usan para el desarrollo en Java, también soportan Python.

Depuradores

La depuración de código es casi fundamental a la hora de resolver bugs en una aplicación.

La depuración consiste en seguir la ejecución de un programa o parte de él, por suerte para Python existe **pdb** que lo hace de forma automática y soporta la fijación de breakpoints. Esta puede ser invocada desde la interfaz del intérprete o del ejecutable Python.

Este depurador funciona de una forma muy sencilla y podemos llamarlo con dos simples líneas

Import pdb

pdb.set_trace()

Al lanzar **pdb** y llegar a un punto predefinido, entrara en marcha el depurador, parando la ejecución del programa y esperando, a través del prompt, para que introduzcamos un comando que nos permita por ejemplo evaluar una variable o continuar la ejecución del programa paso a paso

La sintaxis para lanzar el depurador con nuestro script, seria de la siguiente forma:

python –m pdb ejemplo1.py

Para tener más información sobre este depurador, es recomendable visitar su página oficial.

Profiling

Un Profiler es un programa que mide el rendimiento de la ejecución de otro programa y este ofrece una serie de estadísticas sobre dicho rendimiento.

Este tipo de herramientas es muy útil para mejorar un determinado programa.

Dentro de las librerías de Python, contamos con tres profilers:

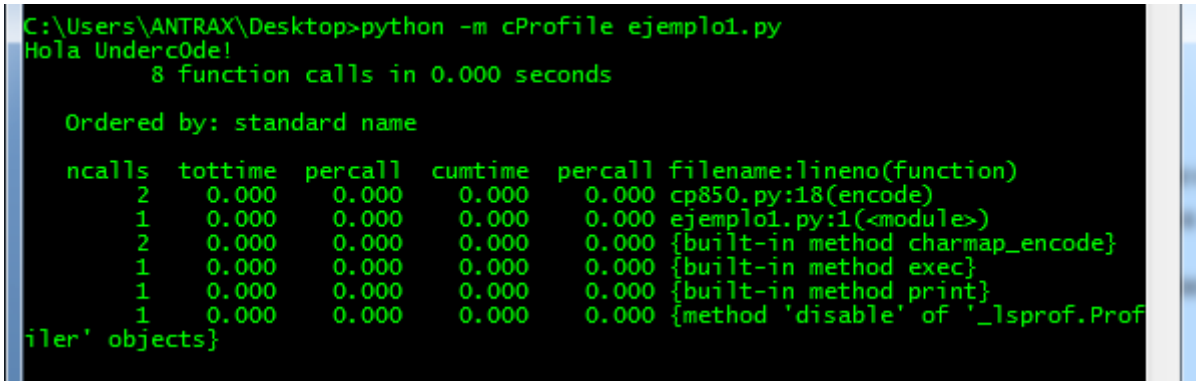
cProfiler, profile y hotshot

Entre los tres, el más recomendable es el primero (cProfiler) ya que este viene desde Python 2.5 y es bueno no solo por su fácil manejo, sino por la información que nos proporciona

Para poder utilizar cProfile y medir el rendimiento de un programa, debemos introducir lo siguiente:

python –m cProfile ejemplo1.py

Como salida de la ejecución tendremos lo siguiente:



Dado que nuestro programa de ejemplo es simplemente un Hola mundo, no tendremos información valiosa, pero si servirá a la hora de tener scripts más complejos.

Novedades de Python 3

Python 3 trae consigo una serie de novedades a diferencia de Python 2.x. Aquellos programadores que quieran pasar sus aplicaciones a Python 3, deberán tener en cuenta lo que detallaremos a continuación.

Veremos las diferencias más significativas que trae esta nueva versión.

El primer cambio se puede ver reflejado a lo que son los ***strings***. En esta versión todos son ***Unicode***. Es por ello que la función ***Unicode()*** se ha suprimido.

El operador **%**, que era utilizado para concatenar ***strings***, ha sido reemplazado por la nueva función ***format().***

Ejemplo en Python2:

>>> cadena = "%s %s" % (cadena1, cadena2)

Ejemplo en Python3:

>>> cadena = "{0} {1}" .format(cadena1, cadena2)

Otro cambio notorio, es el del ***print()***. Como vimos en el primer programa que ejecutamos del **"hola mundo"** en Python 3 el mensaje a demás de ir entre comillas, también va entre paréntesis. Lo mismo ocurre con la función ***exec()***, utilizada para ejecutar código a través de un objeto. Relacionada con esta funcionalidad, en Python 3 ha sido eliminada ***execfile()***. Para simular su funcionamiento, deberemos leer un fichero línea a línea y ejecutar ***exec()*** para cada una de ellas.

En Python 2.x la operación aritmética para realizar la división exacta debe hacerse entre dos números reales, utilizando el operador **/.** En cambio, en la versión 3 de Python se puede realizar la división entre dos números enteros. Para la división entera se utiliza el operador **//**.

Ejemplo en Python2:

>>> 7.0 / 2.0

Ejemplo en Python3:

>>> 7 / 2

Por otro lado, para la división entera seria:

>>> 7 // 2

Como resultado obtendremos 3, ya que solo mostrará la parte entera.

Otro de los cambios, fue en la representación de números en octal, que son los que tienen base 8. Ahora debe ponerse la letra **o** entre el **0** y el número que va a ser representado.

Ejemplo en Python2:

>>> x = 0777

Ejemplo en Python3:

>>> x = 0o77

Con respecto a los diccionarios, la forma de iterar entre sus claves y valores ha cambiado. Ahora las funciones ***iterkeys(), iteritems(), itervalues()*** no son necesarias, en su lugar emplearemos las funciones ***keys(), ítems(), y values()*** respectivamente.

Para comprobar si una clave se encuentra en un diccionario, en lugar de usar la función ***has_key()***, usaremos un if.

>>> if miclave in midiccionario: print ("La clave es")

Para trabajar con compresión de listas y dentro de ellas usamos tuplas, estas deberán ir entre paréntesis. A demás de esto, la función ***sorted()*** devuelve directamente una lista, sin necesidad de convertir su argumento a este tipo de dato.

Algunos nombres de la librería estándar han cambiado los nombres de algunos nombres, como por ejemplo la librería ***httplib*** que ahora se encuentra dentro de ***http*** y para invocarla seria de la siguiente manera:

>>> import http.httplib

Otro ejemplo es el modulo ***cookie*** que también se encuentra dentro de ***http*** y para invocarlo es de la siguiente forma:

>>> import http.Cookies

Estas fueron algunas de las novedades más vistosas en lo que respecta Python 3. Si desean conocer más, pueden visitar el sitio oficial del lenguaje, en donde encontraran un artículo que hace referencia a estos nuevos cambios