Programas:	Python 2	Python 3
Diferencias • **Imprimir (sentencia print)** En Python 3 la sentencia print() es una función y por tanto hay que encerrar entre paréntesis lo que se quiere imprimir, mientras que con Python 2 los paréntesis no son necesarios. • **División de números enteros** En Python 2 la división entre números enteros es otro número entero, y en Python 3 para obtener un resultado con decimales el numerador o el denominador tiene que tener también al menos un decimal. El mismo comportamiento truncado puede obtenerse en Python 3 usando //. • **Módulo __future__** Lo podemos usar para adaptar Python 2 a la sintaxis de Python 3. Por ejemplo, en relación con los puntos anteriores nos permite usar print como una función en Python 2. Otra posibilidad es tener el comportamiento de la división de Python 3 en la versión 2. • **Funciones xrange y range** Las funciones xrange() y range() ayudan a crear listas de enteros que podemos utilizar para realizar bucles for. Las diferencias entre estas dos funciones están fuera del propósito de este post, pero residen en el modo en cómo se generan las listas cosa que tiene un impacto en la memoria utilizada y el tiempo necesario para recorrerlas. Lo que debemos saber en este punto es que xrange() no forma parte de Python 3, y si la intentamos usar en esta versión nos salta una excepción del tipo	• Python 2 es una versión del lenguaje de programación Python que obtendrá soporte mínimo y funciones adicionales en el futuro. • En Python 2, no es obligatorio usar paréntesis. por ejemplo, imprimir "Hola mundo" • En Python 2, la división de enteros devuelve un número entero. 7/2 da 3. Para obtener la respuesta exacta, el programador debe usar 7.0 / 2. 0. • Para hacer una cadena Unicode en Python 2, debe usar un carácter 'u'. por ejemplo, "Hola" • En Python 2, la función raw_input() se usa para obtener información del usuario. Esta función lee una cadena. • En Python 2, la función input () se puede usar para leer como cadenas si están dentro de comillas, de lo contrario, se leen como números. • En Python 2, el generador next() toma el siguiente valor del generador. • Dado que Python 2 estuvo allí durante más tiempo, tiene más compatibilidad con módulos de terceros. Algunos marcos todavía usan Python 2.	• Python 3 es una versión del lenguaje de programación Python que continuamente agrega nuevas funciones y corrige errores. • En Python 3, es obligatorio usar paréntesis. por ejemplo, imprimir ("Hola mundo") • En Python 3, la división de enteros puede dar una respuesta flotante. 7/2 dará 3,5. • En Python 3, la cadena es Unicode de forma predeterminada. • En Python 3, la función raw_input() no está disponible. • En Python 3, la función input() lee la entrada como una cadena. • En Python 3, está escrito como next (generator). • Python 3 tiene soporte limitado para módulos de terceros.

NameError. ● **Iterar un diccionario** Para iterar los elementos clave-valor de un diccionario podemos utilizar el método iteritems() o items() en Python 2. En Python 3 esta operación sólo puede realizarse con el método items(), ya que al usar iteritems() tenemos una excepción del tipo AttributeError. Del mismo modo, los métodos iterkeys() e itervalues() para iterar las claves y los valores de un diccionario respectivamente no existen en Python 3. En su lugar tenemos que usar los métodos keys() y values(). ● **Función next() y método .next()** Recuperar el siguiente elemento de un iterador puede hacerse tanto con la función integrada next() como con el método .next() en Python 2, mientras que en Python 3 next() sólo puede utilizarse como función. Utilizar next() como método en Python 3 nos da una excepción del tipo AttributeError. **Comparación de tipos** Python 2 permite la comparación entre objetos de tipo distinto, sin embargo Python 3 es restrictivo en este aspecto dándonos una excepción del tipo TypeError. ● **Función input()** La función input() sirve para capturar datos procedentes del teclado. En Python 2 esta función trata los datos tal cual sin realizar ninguna conversión. Por ejemplo, si entramos un número entero, la entrada será de tipo int. Si queremos que la entrada se trate como una cadena, entonces tenemos que usar la función raw_input() ya que ésta convierte los datos a tipo str. En Python 3 se ha suprimido la función raw_input(), y su comportamiento lo toma la		

función input(). Esto facilita el tratamiento de los datos entrados por teclado ya que sabemos de antemano que siempre serán strings. ● **Cadenas de texto Unicode** En Python 2 las cadenas de texto (tipo str) están codificadas en formato ASCII con lo que cada carácter sólo requiere 7 bits de información. Por otro lado, las cadenas de texto codificadas en bytes requieren 8 bits de información. Por lo tanto, sólo es posible combinar ambos tipos cuando la representación en bytes sólo contiene caracteres ASCII. En Python 3 todas las cadenas de texto son Unicode (8 bits). Éstas pueden almacenarse como texto (tipo str) o como datos binarios (tipo bytes), siendo imposible combinar ambos tipos ya que nos da una excepción TypeError.		

Correcciones. **Python 2.** El cambio más importante en Python 2.0 puede que no sea en el código en absoluto, sino en la forma de desarrollar Python: en mayo de 2000 los desarrolladores de Python comenzaron a utilizar las herramientas puestas a disposición por SourceForge para almacenar el código fuente, rastrear los informes de errores y gestionar la cola de envíos de parches. **Python 3.** Este artículo explica las nuevas características de Python 3.0, en comparación con 2.6. Python 3.0, también conocido como «Python 3000» o «Py3K», es la primera versión intentionally backwards incompatible de Python. Python 3.0 se lanzó el 3 de diciembre de 2008. Hay más cambios que en una versión típica y más que son importantes para todos los usuarios de Python. Sin embargo, después de digerir los cambios, descubrirá que Python realmente no ha cambiado tanto; en general, estamos solucionando principalmente molestias y defectos conocidos, y eliminando una gran cantidad de viejas cosas.	➢ Las divisiones de enteros pueden dar resultados inesperados. Es recomendable utilizar la división flotante utilizando from __future__ import division. ➢ El método next() se llama next en Python 2. En Python 3, se llama __next__. ➢ En Python 2, las cadenas de bytes se representan como objetos "str". En Python 3, se representan como objetos "bytes". ➢ En Python 2, la función xrange() es más eficiente que range() para iterar sobre un rango de enteros. ➢ Las comparaciones de objetos de diferentes tipos se basan en el orden de clasificación de los tipo	➢ Todas las cadenas son de Unicode por defecto en Python 3. En Python 2, había dos tipos diferentes para manejar texto: str para texto ASCII y unicode para texto Unicode. ➢ La función print en Python 3 es una función, mientras que en Python 2 es una sentencia. ➢ La división de enteros siempre produce un resultado de punto flotante en Python 3. En Python 2, la división de enteros truncaba el resultado a un entero. ➢ La función zip devuelve un objeto de zip en Python 3, mientras que en Python 2 devuelve una lista. ➢ Los nombres de variables y funciones no se pueden sobrescribir en una misma declaración en Python 3, mientras que en Python 2 sí se permite.