**INSTITUTO SUPERIOR TECNOLÓGICO DEL SUR**

**CARRERA**

**DISEÑO Y PROGRAMACIÓN WEB**

**LENGUAJE WEB 2**

**“Evolución de Python”**

**PROFESOR(A) : Amado Cerpa Juan Andrés**

**ALUMNO : Vilca Apaza Christian**

**SEMESTRE : V**

9/10/2020

Índice

[Historia de Python 3](#_Toc53100720)

[Versiones 3](#_Toc53100721)

[Cambios entre Python 2 y Python 3 4](#_Toc53100722)

[Imprimir (sentencia print) 4](#_Toc53100723)

[División de números enteros 4](#_Toc53100724)

[Módulo \_\_future\_\_ 5](#_Toc53100725)

[Funciones xrange y range 5](#_Toc53100726)

[Comparación de tipos 6](#_Toc53100727)

[Función input 7](#_Toc53100728)

[Cadenas de texto Unicode 7](#_Toc53100729)

[Motivos del cambio de versión de Python 8](#_Toc53100730)

[Mejor soporte Unicode 8](#_Toc53100731)

[Mejora la división entera 9](#_Toc53100732)

# Historia de Python

Guido van Rossum ideó el lenguaje Python a finales de los 80 y comenzó a implementarlo en diciembre de 1989. En febrero de 1991 publicó la primera versión pública, la versión 0.9.0. La versión 1.0 se publicó en enero de 1994, la versión 2.0 se publicó en octubre de 2000 y la versión 3.0 se publicó en diciembre de 2008.

Hasta 2018, el desarrollo de Python estaba dirigido personalmente por Guido van Rossum y bajo el paraguas de la fundación Python Software Foundation. En julio de 2018 Guido van Rossum anunció que dejaría de dirigir el desarrollo de Python. Desde 2019 el desarrollo de Python está dirigido por un consejo de dirección de cinco miembros elegidos entre los desarrolladores de Python y que se renovará anualmente.

# Versiones

Las versiones de Python se identifican por tres números X.Y.Z, en la que:

X corresponde a las grandes versiones de Python (1, 2 y 3), incompatibles entre sí:

Los principales cambios introducidos en Python 2 fueron las cadenas Unicode, las comprensiones de listas, las asignaciones aumentadas, los nuevos métodos de cadenas y el recolector de basura para referencias cíclicas.

Los principales cambios introducidos en Python 3 fueron la separación entre cadenas Unicode y datos binarios, la función print(), cambios en la sintaxis, tipos de datos, comparadores, etc.

Por el momento, no hay planes de crear una nueva versión Python 4, incompatible con las anteriores.

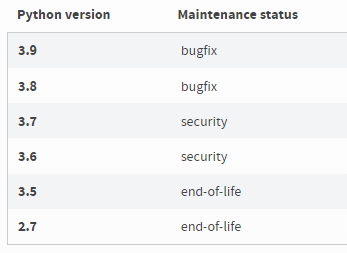
Y corresponde a versiones importantes en las que se introducen novedades en el lenguaje, pero manteniendo la compatibilidad (salvo excepciones).

Desde la versión 2.0 hasta 2019, las versiones X.Y se han publicado aproximadamente cada año y medio y se han mantenido durante cinco años, excepto la versión 2.7, que se mantuvo durante diez años, hasta el 1 de enero de 2020 (aunque se publicó una versión final en abril de 2020).

En 2019 se decidió pasar a publicar nuevas versiones X.Y anualmente, en octubre, manteniéndolas durante cinco años. Así, Python 3.9 se ha publicado en octubre de 2020 y Python 3.10 se publicará en octubre de 2021.

Z corresponde a versiones menores que se publican durante el período de mantenimiento, en las que sólo se corrigen errores durante el primer año y fallos de seguridad en los cuatro restantes.

Normalmente, se publica una última versión X.Y.Z justo antes de que una versión X.Y deje de mantenerse. Algunas empresas comerciales ofrecen el mantenimiento de versiones antiguas una vez acabado el mantenimiento oficial.



# Cambios entre Python 2 y Python 3

## Imprimir (sentencia print)

Esta sea quizás la diferencia más conocida de todas. 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.

# Python 2

>>> print 'Programa en Python'

Programa en Python

# Python 3

>>> print('Programa en Python')

Programa en Python

## División de números enteros

En Python 2 la división entre números enteros es otro número entero, y para obtener un resultado con decimales el numerador o el denominador tiene que tener también al menos un decimal.

# Python 2

>>> 1/2

0

>>> 1.0/2

0.5

# Python 3

>>> 1/2

0.5

>>> 1.0/2

0.5

El mismo comportamiento truncado puede obtenerse en Python 3 usando //.

# Python 3

>>> 1//2

0

## Módulo \_\_future\_\_

Este módulo proporciona retrocompatibilidad de comandos no compatibles entre distintas versiones. 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.

# Python 2

>>> **from** \_\_future\_\_ **import** print\_function

>>> print('Programa en Python')

Programa en Python

Otra posibilidad es tener el comportamiento de la división de Python 3 en la versión 2.

# Python 2

>>> **from** \_\_future\_\_ **import** division

>>> 1/2

0.5

## 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 NameError.

Iterar un diccionario

Para iterar los elementos clave-valor de un diccionario podemos utilizar el método iteritems() o items() en Python 2.

# Python 2

d = {'animal': 'perro', 'vehiculo': 'coche'}

**for** k,v **in** d.iteritems():

print k,':',v

vehiculo : coche

animal : perro

En Python 3 esta operación sólo puede realizarse con el método items(), ya que al usar iteritems() tenemos una excepción del tipo AttributeError.

# Python 3

d = {'animal': 'perro', 'vehiculo': 'coche'}

**for** k,v **in** d.iteritems():

print(k,':',v)

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

AttributeError: 'dict' object has no attribute 'iteritems'

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 next() como con el método .next() en Python 2, mientras que en Python 3 next sólo puede utilizarse como función.

# Python 2

>>> iterador = (letra **for** letra **in** 'python')

>>> next(iterador)

'p'

>>> iterador.next()

'y'

Utilizar next como método en Python 3 nos da una excepción del tipo AttributeError

# Python 3

>>> iterador = (letra **for** letra **in** 'python')

>>> next(iterador)

'p'

>>> iterador.next()

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

AttributeError: 'generator' object has no attribute 'next'

## 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.

# Python 2

>>> 1 < '1'

True

# Python 3

>>> 1 < '1'

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

TypeError: '<' not supported between instances of 'int' and 'str'

## 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.

# Python 2

>>> entrada = input('Introduce un número: ')

Introduce un número: 1

>>> type(entrada)

<**class** 'int'>

>>> entrada = raw\_input('Introduce un número: ')

Introduce un número: 1

>>> type(entrada)

<**class** '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.

# Python 3

>>> entrada = input('Introduce un número: ')

Introduce un número: 1

>>> type(entrada)

<**class** 'str'>

## 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.

# Python 3

>>> a = 'Hola'

>>> type(a)

<**class** 'str'>

>>> b = b'mundo'

>>> type(b)

<**class** 'bytes'>

>>> a + b

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

TypeError: can only concatenate str (not "bytes") to str

# Motivos del cambio de versión de Python

Ciertamente, Python 2 representa todo un legado de más de una década, y evidentemente sigue siendo la popular en muchas compañías gracias a su sólido software.

Sin embargo, Python 3 es el futuro y no solo las empresas comienzan a moverse hacia lo nuevo, sino que si eres un principiante que deseas aprender de este lenguaje es mucho mejor que lo hagas en la nueva versión que en una que comienza a volverse obsoleta.

## Las incompatibilidades

Aunque muchas bibliotecas de Python 2 y Python 3 son similares o compatibles, muchas otras definitivamente no lo son.

Desde luego, hay que echarle la culpa a los desarrolladores que crean estas bibliotecas para ser usadas estrictamente con Python 3. Aunque, Python 2 tiene también bibliotecas antiguas que ya están en desuso y que por tanto no funcionan para nada en Python 3.

Claro que esto no significa que no podamos usar algunos mecanismos para mudar una biblioteca de 2 al 3 pero ¿para qué complicarse? Ni un principiante ni un experto requieren de algo así. Además, las bibliotecas de Python 3 son muy completas.

## Mejor soporte Unicode

En Python 2, las cadenas se almacenan como ASCII de forma predeterminada y debíamos agregar una u’ si queríamos almacenarla como cadenas Unicode. En Python 3 esto ya no es necesario ya que las cadenas de texto se almacenan predeterminadamente en Unicode.

Esto es muy importante ya que Unicode es mucho más versátil que ASCII.  Las cadenas Unicode pueden almacenar letras en distintos idiomas, números romanos, símbolos, emojis, etc., ofreciéndonos muchísimas más opciones.

## Mejora la división entera

En Python 2, si escribes un número sin ningún dígito después del punto decimal, redondea su cálculo al número entero más cercano. Por ejemplo, si intentas dividir 5 entre 2 (5/2) el resultado será 2 debido al redondeo.

Tendrías que escribirlo como 5.0 / 2.0 para obtener la respuesta exacta de 2.5. Sin embargo, en Python 3, la expresión 5/2 devolverá el resultado esperado de 2.5 sin tener que preocuparse por agregar esos ceros adicionales.

Sin duda, este pequeño y simple ejemplo demuestra como la sintaxis de Python 3 termina siendo mucho más intuitiva, algo que le dará mucha más comodidad a los novatos que intentan aprender el lenguaje de programación Python.

# Conclusión

La gran evolución que Python sufrió en estos años desde su creación ha dado grandes beneficios especialmente en la velocidad ya que Python al ser un lenguaje interpretando tiende a ser más lento que c++ o c lenguajes copilados de bajo nivel, creo q su principal motivo de cambio fue eso la velocidad de ejecución, reparación de bugs y adaptarse a los tiempos modernos.

# Bibliografía

<https://www.python.org/downloads/>

<https://docs.python.org/3/>

<https://es.wikipedia.org/wiki/Python>

<https://www.crehana.com/pe/blog/web/que-es-python/>

<https://www.mclibre.org/consultar/python/otros/historia.html#:~:text=La%20versi%C3%B3n%201.0%20se%20public%C3%B3,la%20fundaci%C3%B3n%20Python%20Software%20Foundation.>

<https://www.redeszone.net/2018/02/11/novedades-python-3-7/>

<https://www.programaenpython.com/miscelanea/diferencias-entre-python-2-y-3/>

<https://www.inteldig.com/2018/11/python-2-vs-python-3-deberias-aprender/>