P3. T2. Tabla comparativa de Python 2 vs Python 3

Alumna: Daniela Mendez Ramirez

Número de Cuenta: 258331-9

Aplicaciones de Redes

Profesor: Omar Vázquez González

Fecha de Entrega: 30 de enero de 2025

Tabla Comparativa:

Característica	Python 2	Python 3
Cadena de caracteres en bytes	str es ASCII por defecto, unicode debe especificarse	bytes es separado de str
Comparación de None	None < 1 es válido	None < 1 lanza TypeError
Comparación de objetos diferentes	Se permite comparar tipos diferentes ("2" > 1 no da error)	Comparar tipos incompatibles lanza TypeError ("2" > 1 es un error)
Compatibilidad	No compatible con Python 3	No compatible con Python 2
Diccionario ordenado (OrderedDict)	No hay orden en dict (necesita OrderedDict)	Desde Python 3.7, dict mantiene el orden de inserción
División de enteros	5 / 2 = 2 (división entera por defecto)	5 / 2 = 2.5 (división flotante por defecto)

Entrada de usuario	raw_input() para cadenas, input() evalúa expresiones	input() siempre devuelve cadena (str)
Formateo de cadenas	"%s %d" % ("Texto", 5)	"{0}{1}".format("Texto", 5) o f"{texto}{num}" desde Python 3.6
Función print	Declaración: print "Hola"	Función: print("Hola")
Funciones next()	next() es un método (it.next())	next() es una función global (next(it))
Iteración sobre diccionarios	dict.keys(), dict.values(), dict.items() devuelven listas	dict.keys(), dict.values(), dict.items() devuelven iteradores
Iteradores (xrange vs range)	xrange() para iteradores, range() devuelve lista	range() funciona como xrange() (iterador)
Manejo de archivos	open() devuelve cadenas ASCII (str)	open() devuelve Unicode (str) por defecto
Manejo de excepciones	except Exception, e:	except Exception as e:

Manejo de super()	super(Clase, self).metodo()	super().metodo() sin necesidad de self ni clase
Módulo __future__	Usado para importar características de Python 3	No es necesario para compatibilidad con versiones futuras
Soporte	Fin de soporte oficial (desde 2020)	Versión activa con soporte y actualizaciones
Soporte de L en enteros largos	123456789L para enteros largos	int es de precisión arbitraria, no usa L
Tipos de cadena	str para ASCII, unicode para Unicode	str es Unicode (UTF-8 por defecto)
Unicode en print	Se necesita u"texto" explícito	Soporta Unicode por defecto

Nuevas Funcionalidades y Mejoras en Python 3

Funcionalidad	Python 2	Python 3
f-strings (cadenas formateadas)	No disponible	f"Hola {nombre}" (desde Python 3.6)
Manejo de múltiples contextos (with)	Solo se permite un contexto	Se pueden usar múltiples: with A() as a, B() as b:
Corutinas (async / await)	No soportado	Soporta programación asíncrona (desde 3.5)
Módulo pathlib para manejar rutas	No disponible	Introducido en Python 3.4
Módulo enum	No disponible	Introducido en Python 3.4
Módulo typing (tipado estático opcional)	No disponible	Introducido en Python 3.5
Operador @ para multiplicación de matrices	No disponible	Introducido en Python 3.5 (para numpy)

Argumentos de función *args, **kwargs ordenados	No garantiza orden	Garantiza orden (desde Python 3.6)
Depuración con breakpoint()	No disponible	Introducido en Python 3.7 (equivalente a import pdb; pdb.set_trace())
Manejo de objetos bytes	str y unicode mezclados	str es Unicode y bytes es separado
Iteración de elementos en dict	dict.keys(), dict.values(), dict.items() devuelven listas	Devuelven iteradores (uso eficiente de memoria)
Mejor rendimiento de set y dict	Más lento en búsquedas e inserciones	Optimizado, mayor velocidad

Librería / Módulo	Python 2	Python 3
urllib y urllib2	urllib, urllib2, urlparse	urllib.request, urllib.parse, urllib.error
ConfigParser	ConfigParser	configparser (renombrado)
StringIO y cStringIO	StringIO, cStringIO (separados)	io.StringIO, io.BytesIO
print en módulos	print como declaración	print() como función estándar
cmp() para comparación	Disponible	Eliminado en favor de key en sorted()
raw_input()	raw_input() devuelve cadenas	Eliminado (usar input())
commands	commands para ejecutar comandos shell	Reemplazado por subprocess
basestring	Disponible (str y unicode)	Eliminado (usar str)

Optimización	Python 2	Python 3
Uso de memoria	Más consumo debido a list	range(), dict.items() son iteradores, menos memoria
Mejor manejo de hilos (threading)	Global Interpreter Lock (GIL) presente	Mejoras en concurrencia y asyncio
Mejor recolección de basura	Menos eficiente	Optimización del recolector de basura
Módulo multiprocessing	Disponible	Mejorado, permite mejor paralelismo

Conclusión

Python 3 representa una evolución necesaria y acertada del lenguaje, mejorando la eficiencia, legibilidad y modernización. La transición fue desafiante, pero las mejoras, como las f-strings, la programación asíncrona y el tipado opcional, hacen que escribir código sea más intuitivo y estructurado. Aunque Python 2 fue clave en la popularización del lenguaje, su retiro permitió un desarrollo sin limitaciones heredadas. Hoy, Python 3 es más robusto y preparado para el futuro, consolidándose como una de las mejores opciones en programación.