Anexo: Diferencias python2 y 3

Anexo de diferencias Python2 y 3

Vamos a hacer un breve listado de algunas de las diferencias evidentes que un programador de python 2.x se encuentra cuando comienza a enfrentarse con python 3.x

- Autocompletado de código con tabulador.

  La consola **REPL** del lenguaje python3, me permite autocompletar la sintaxis con la tecla TABULADOR o mostrar opciones de completado.

  Ej: **micadena = 'Las ruedas del autobús girando van'**

  micadena.startw...[tab]

- Cadenas Unicode en python 3:

  print("más ñandú")

Anexo de diferencias Python2 y 3

- **print()**

  En python 3 es una función con () y parámetros opcionales

  ```
  print("Hola mundo,", end="")
  ```

- Division de enteros ***//*** y punto flotante ***/***

  ```
  print (3 / 2)
  print (3 // 2)
  ```

- El tipo de datos **long** ya no existe.

  Todos los enteros son **int**.

  Y la representación con **L** desaparece. EJ: 4L

Anexo de diferencias Python2 y 3

- input() de python 3 equivale al raw_input() de python 2

```
>>> mi_input = input("ingresar un numero: ")
ingresar un numero: 123
>>> type(mi_input)
<class 'str‘>
```

- El método .next() de python 2 ahora es función en python 3

```
>>> mi_generator = (letra for letra in "abcdef")
>>> next(mi_generator)
'a’
```

Anexo de diferencias Python2 y 3

- La comparación de tipos de datos diferentes dan errores en python 3

En Python 2.x lo siguiente no daba error:

```
[1, 2] > 'foo' =  False
(1, 2) > 'foo' =  True
[1, 2] > (1, 2) =  False
lista = [1,3,'a']
min(lista)
```

- Lanzar excepciones con raise() debe ser con paréntesis de función

raise IOError("error de archivo“)

Anexo de diferencias Python2 y 3

- Palabras reservadas

En python 3 **nonlocal** es una palabra reservada y **exec** ya no lo es.

- Los tipos devueltos difieren para algunas secuencias:

La función **range()** ahora no devuelve una lista sino un objeto de tipo range, que se puede iterar sin problema e incluso convertir a lista con **list()**.

El método **.keys()** de los diccionarios devuelve el tipo **dict_keys**. Pasa igual que con **range**.

El método **.values()** de los diccionarios devuelve el tipo **dict_values**.

```
>>> mi_diccionario.keys()
dict_keys([‘Mario', ‘Trini'])
```