8 Julio

January 31, 2018

0.1 Sobre el futuro de map, filter, reduce y lambda

0.1.1 All Things Pythonic: The fate of reduce() in Python 3000

by Guido van van Rossum, March 10, 2005 http://www.artima.com/weblogs/viewpost.jsp?thread=98196

```
lambda argument_list: expression
```

```
In [5]: sum_lambda = lambda x, y : x + y
```

```
In [11]: sum_lambda(5,)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-11-2823ad53bce9> in <module>()
----> 1 sum_lambda(5,)


TypeError: <lambda>() missing 1 required positional argument: 'y'
```

```
In [7]: # def sum_def
        def sum_function(x, y):
            return x+ y
```

```
In [15]: sum_function(5, 8)
```

```
Out[15]: 13
```

```
In [9]: print(type(sum_lambda))
        print(type(sum_function))
```

```
<class 'function'>
<class 'function'>
```

0.1.2 List Comprehension

https://docs.python.org/3/tutorial/datastructures.html

```
nueva_lista = [x for x in iterable]
```

```
In [28]: letras_ciencia_set = set([letra for letra in 'Ciencia de datos ABCHJAASAS|'])
```

```
In [23]: print(letras_ciencia_set)
```

```
{'J', 'c', 'n', 'd', 'a', '|', ' ', 'C', 'i', 's', 'A', 't', 'o', 'S', 'H', 'B', 'e'}
```

```
In [31]: letras_ciencia_lc = [letra for letra in 'Ciencia de datos ABCHJAASAS|']
```

```
In [33]: print(letras_ciencia_lc)
```

```
['C', 'i', 'e', 'n', 'c', 'i', 'a', ' ', 'd', 'e', ' ', 'd', 'a', 't', 'o', 's', ' ', 'A', 'B'
```

```
In [38]: letras_ciencia = []

         for simbolo in 'Ciencia de datos ABCHJAASAS|':
             letras_ciencia.append(simbolo)
```

```
In [42]: print(letras_ciencia)

         aaa = 'asd'
```

```
['C', 'i', 'e', 'n', 'c', 'i', 'a', ' ', 'd', 'e', ' ', 'd', 'a', 't', 'o', 's', ' ', 'A', 'B'
```

0.1.3 Condicionales

```
In [44]: tupla_lugares = ('cafeteria', 'biblioteca', 'rectoria', 'estacionamiento')
```

```
In [46]: lista_lugares = [lugar for lugar in tupla_lugares if lugar != 'rectoria']
```

```
In [47]: lista_lugares
```

```
Out[47]: ['cafeteria', 'biblioteca', 'estacionamiento']
```

Operadores Matemáticos

```
In [53]: lista_numero_mas_dos = [numero  + 2 for numero in range(10) if numero % 2 == 0]
```

```
In [54]: lista_numero_mas_dos
```

```
Out[54]: [2, 4, 6, 8, 10]
```

```
In [60]: lista_numero = [numero for numero in range(11)]

         print(lista_numero)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Agregamos la condicion:

```
In [62]: lista_numero_condicion = [numero for numero in range(10) if numero % 2 == 0]

In [63]: lista_numero_condicion

Out[63]: [0, 2, 4, 6, 8]
```

Incrementando dos enteros más:

```
In [64]: lista_numero_mas_dos = [numero  + 2 for numero in range(10) if numero % 2 == 0]

         print(lista_numero_mas_dos)

[2, 4, 6, 8, 10]
```

Generar una lista de números donde obtengamos el cuadrado de cada valor, si originalmente es impar dentro del rango 0-99

```
In [68]: lista_cuadrados = [numero_impar ** 2 for numero_impar in range(100) if numero_impar %

In [66]: lista_cuadrados

Out[66]: [1,
          9,
          25,
          49,
          81,
          121,
          169,
          225,
          289,
          361,
          441,
          529,
          625,
          729,
          841,
          961,
          1089,
          1225,
          1369,
```

```
 1521,
 1681,
 1849,
 2025,
 2209,
 2401,
 2601,
 2809,
 3025,
 3249,
 3481,
 3721,
 3969,
 4225,
 4489,
 4761,
 5041,
 5329,
 5625,
 5929,
 6241,
 6561,
 6889,
 7225,
 7569,
 7921,
 8281,
 8649,
 9025,
 9409,
 9801]
```

Elementos anidados

```
In [75]: ## Generar una nueva lista con elementos divisibles por 3 y 5 en el rango 0-99

         lista_numeros_35 = [num for num in range(100) if num % 3 == 0 if num % 5 == 0 if num %
```

```
In [76]: print(lista_numeros_35)
```

```
[15, 45, 75]
```

```
In [77]: lista_numeros_35 = []

         for num in range(100):
             # if <expresion> True >>
             if num % 3 ==0:
```

```
                if num % 5 == 0:
                    if num % 2 != 0:
                        lista_numeros_35.append(num)

        print(lista_numeros_35)
```

```
[15, 45, 75]
```

nested loops

```
In [107]: lista_loops = []


          for numero_prim_dimension in [20, 30, 60, 80]:
              for numero_seg_dimension in [2, 3, 6, 8]:
                  for n in range(9):
                      lista_loops.append(numero_prim_dimension * numero_seg_dimension + n)
```

```
In [108]: print(lista_loops)
```

```
[40, 41, 42, 43, 44, 45, 46, 47, 48, 60, 61, 62, 63, 64, 65, 66, 67, 68, 120, 121, 122, 123, 1
```

```
In [112]: # num_prim_dimension = n1
          # num_seg_dimension = n2
          lista_loops = [n1 * n2 + n3 for n1 in [20, 30, 60, 80] for n2 in [2, 3, 6, 8] for n3

          print(lista_loops)
```

```
[40, 41, 42, 43, 44, 45, 46, 47, 48, 60, 61, 62, 63, 64, 65, 66, 67, 68, 120, 121, 122, 123, 1
```

```
In [113]: # Comprension de listas sobre un DataFrame de Pandas
```

```
In [111]: import pandas as pd
```

Crear un DF dummy con 3 Columnas: - Nombre - Semestre - Final
Y 5 renglones

```
In [115]: data_dict = {
              'nombre': [ 'Sergio', 'Vladimir', 'Tania', 'Elsy', 'Oktavia'],
              'semestre': [1, 1, 3, 5, 5],
              'final': [7.5, 8, 9, 8.2, 4]
          }
```

```
In [121]: df = pd.DataFrame(data_dict)
          df = df[['nombre', 'semestre', 'final']]
          df
```

Out[121]:

	nombre	semestre	final
0	Sergio	1	7.5
1	Vladimir	1	8.0
2	Tania	3	9.0
3	Elsy	5	8.2
4	Oktavia	5	4.0

In [122]:
```
# Loop
sig_sem = []

for renglon in df['semestre']:
    sig_sem.append(renglon + 1)

df['sig_semestre_loop'] = sig_sem
```

In [132]:
```
# list comprehension

df['semestre_lc'] = [renglon + 1 for renglon in df['semestre']]
```

In [135]:
```
df
```

Out[135]:

	nombre	semestre	final	sig_semestre_loop	semestre_lc
0	Sergio	1	7.5	2	2
1	Vladimir	1	8.0	2	2
2	Tania	3	9.0	4	4
3	Elsy	5	8.2	6	6
4	Oktavia	5	4.0	6	6

In [ ]:
```
!wget https://gist.githubusercontent.com/israelzuniga/bef0001668c2bab3c3686db3759f0faa/
```