Listas

June 22, 2018

1 Resumen IIC-1103

Por: Nicolás Quiroz, Daniel Leal, Alfonso Irarrázaval.

---

2 Listas

Las listas son una forma de guardar un conjunto de valores python.
Ejemplo:

```
In [30]: lista_supermercado = ['Huevos', 'Leche', 'Cereal', 'Harina']
```

Lo anterior corresponde a un conjunto de `strings` en una lista de supermercado.

£ Puedo guardar un *conjunto* de listas? Si ! Y es bastante facil. Supongamos tienes una generacion de un colegio, con distintos alumnos, por curso:

```
In [31]: generacion_2017 = [['Nicolas Quiroz', 'Benil Wakas', 'Jorge Muñoz'],
                            ['Ignacio Sanchez','Claudio Rivera', 'Ruben Alvarado'],
                           ['Cristian Ruz', 'Marcos Sepúlveda' , 'Daniel Leal']]
```

Como ven, para guardar listas dentro de listas, solo basta escribirlas separadas por una coma. Una forma que la gente suele ver las listas de listas, es como una matriz donde cada fila es una lista y cada columna es un elemento de cada lista.

Tip : **Las listas pueden guardar muchos tipos de cosas a a la vez! (strings, otras listas, ints en una misma lista)**

£Qué puedo hacer con las listas (y, por ende, listas de listas)? Muchas cosas, asi que resumiremos las mas esenciales.

- Agregar elementos

```
In [32]: lista_supermercado.append('manjar')
         print(lista_supermercado)
```

```
['Huevos', 'Leche', 'Cereal', 'Harina', 'manjar']
```

- Borrar elementos

```
In [33]: lista_supermercado.remove('Huevos')
         print(lista_supermercado)
```

```
['Leche', 'Cereal', 'Harina', 'manjar']
```

- Sacar y retornar elementos

```
In [34]: curso_a = generacion_2017.pop(1) # Si no pongo un numero, por defecto es el ultimo el
         print('Extraido:', curso_a, '\nGeneracion:', generacion_2017)
```

```
Extraido: ['Ignacio Sanchez', 'Claudio Rivera', 'Daniel Vidal']
Generacion: [['Nicolas Quiroz', 'Benil Wakas', 'Jorge Muñoz'], ['Cristian Ruz', 'Marcos Sepúlv
```

- Sumar listas

```
In [35]: abuelos_paternos = ['Jose Manuel', 'Antonia']
         abuelos_maternos = ['Alfonso', 'Florencia']
         abuelos = abuelos_paternos + abuelos_maternos
         print(abuelos)
```

```
['Jose Manuel', 'Antonia', 'Alfonso', 'Florencia']
```

- Recorrerlas

```
In [43]: def comprar(producto, lista_super):
             print(producto.capitalize(), "comprado!")
             print('Te quedan en la lista:', lista_super)

         print('Lista inicial', lista_supermercado)
         # Ciclos while
         print(' ===== While ====')
         copia_while = lista_supermercado.copy()
         while copia_while: # Equivalente a decir while len(copia_while) > 0
             comprar(copia_while.pop(), copia_while) # Sacando los ultimos de la lista primero

         print('===== For each ====')
         # Version 1: Por elemento en la lista
         copia_for1 = lista_supermercado.copy()
         for producto in copia_for1:
             # Acá el producto es un string, un elemento de la lista
             copia_for1.remove(producto)
             comprar(producto, copia_for1)

         # Version 2: Por indice en la lista
```

```
        print('==== For i ===')
        copia_for2 = lista_supermercado.copy()
        for i in range(len(copia_for2)):
            # Acá el i es un entero, indica el indice de la lista, partiendo de 0.
            print('Indice', i)
            # Aqui no los saco de la lista, por lo que sigue igual.
```

```
Lista inicial ['Leche', 'Cereal', 'Harina']
 ===== While ====
Harina comprado!
Te quedan en la lista: ['Leche', 'Cereal']
Cereal comprado!
Te quedan en la lista: ['Leche']
Leche comprado!
Te quedan en la lista: []
===== For each ====
Leche comprado!
Te quedan en la lista: ['Cereal', 'Harina']
Harina comprado!
Te quedan en la lista: ['Cereal']
==== For i ===
Indice 0
Indice 1
Indice 2
```

Oh no! Quedó un producto en la lista en el `For Each`!! Alguna idea de por qué sucede esto? Veamos que ocurre si saco productos de la lista con el `For i`

```
In [40]: print('==== For i ===')
         copia_for2 = lista_supermercado.copy()
         for i in range(len(copia_for2)):
             # Acá el i es un entero, indica el indice de la lista, partiendo de 0.
             comprar(copia_for2.pop(i), copia_for2)
             # Aqui no los saco de la lista, por lo que sigue igual.
```

```
==== For i ===
Leche comprado!
Te quedan en la lista: ['Cereal', 'Harina']
Harina comprado!
Te quedan en la lista: ['Cereal']
```

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

IndexError                                Traceback (most recent call last)

<ipython-input-40-a7c4572466ed> in <module>()
```

```
      3 for i in range(len(copia_for2)):
      4     # Acá el i es un entero, indica el indice de la lista, partiendo de 0.
----> 5     comprar(copia_for2.pop(i), copia_for2)
      6     # Aqui no los saco de la lista, por lo que sigue igual.


IndexError: pop index out of range
```

Ha ocurrido un error! Por qué?

Si bien en ambos for (`For each`, `For i`) ocurren problemas distintos, la razon es la misma. Cuando yo saco elementos de la lista, su tamaño va cambiando!

- `For Each`: Aqui lo ocurre es que el `for` saca el elemento en la posicion que sigue de la lista **en su estado actual** o sea, lo que ocurre en el for:
  1. Saco el elemento 0 -> 'Leche', la lista ahora es ['Cereal', 'Harina'], próximo índice: 1
  2. Saco el elemento 1 **pero** de la lista ['Cereal', 'Harina'], o sea -> 'Harina'
- `For i`: Aqui lo ocurre es que el `for` trata de sacar el elemento en una posicion que ya no exist, o sea, lo que ocurre en el for:
  1. Saco el elemento 0 -> 'Leche', la lista ahora es ['Cereal', 'Harina'], proximo indice: 1
  2. Saco el elemento 1 **pero** de la lista ['Cereal', 'Harina'], o sea -> 'Harina' , próximo índice: 2
  3. Trato de sacar el elemento 2 de la lista ['Cereal'] ERROR.

Cuál es la lección de esto entonces? Los `while, foreach, y fori` tienen distintos funcionamientos, por lo que sirven para distintos casos. Aquí hay un resumen de cuando usar cada uno: - `while` con lista: Cuando quieres terminar cuando la lista este vacia. - `foreach` con lista: Cuando quieres recorrer los elementos **sin** alterarla - `fori` con lista: Cuando quieres recorrer varias listas de mismo largo, al mismo tiempo, o si me interesa el índice de cada elemento.

Obviamente, estas son sugerencias, dado siempre se puede hacer lo mismo con cualquiera de los tres.

2.0.1 Otras cosas que puedes hacer con listas

- Insertar elementos

```
In [45]: posicion = 3
         lista_supermercado.insert(posicion, "queso")
         print(lista_supermercado)
```

```
['Leche', 'Cereal', 'Harina', 'queso', 'queso']
```

- Slicing!

```
In [47]: a_y_b = generacion_2017[0:1]
```

```
        """
        Otros utiles:
        [a:-1] -> todos menos el ultimo
        [:5] -> desde el comienzo hasta el 5
        [6:] -> desde el 6 hasta el final
        [:] -> todo
        [::-1] -> todo invertido.
        """

        print(a_y_b)
```

```
[['Nicolas Quiroz', 'Benil Wakas', 'Jorge Muñoz']]
```