

Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

Listas, tuplas y diccionarios

Listas

- La lista es un tipo de colección ordenada.
- Sería equivalente a lo que en otros lenguajes se conoce por arrays, o vectores.
- Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos e incluso listas

Listas

Table of Contents

- 5. Data Structures
 - 5.1. More on Lists
 - 5.1.1. Using Lists as Stacks
 - 5.1.2. Using Lists as Queues
 - 5.1.3. List Comprehensions
 - 5.1.4. Nested List Comprehensions
 - 5.2. The `del` statement
 - 5.3. Tuples and Sequences
 - 5.4. Sets
 - 5.5. Dictionaries
 - 5.6. Looping Techniques
 - 5.7. More on Conditions
 - 5.8. Comparing Sequences and Other Types

Previous topic

4. More Control Flow Tools

Next topic

6. Modules

This Page

[Report a Bug](#)
[Show Source](#)

5. Data Structures

This chapter describes some things you've learned about already in more detail, and adds some new things as well.

5.1. More on Lists

The list data type has some more methods. Here are all of the methods of list objects:

`list.append(x)`

Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.extend(iterable)`

Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

`list.insert(i, x)`

Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.remove(x)`

Remove the first item from the list whose value is equal to `x`. It raises a `ValueError` if there is no such item.

`list.pop([i])`

Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the `i` in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

`list.clear()`

Remove all items from the list. Equivalent to `del a[:]`.

`list.index(x[, start[, end]])`

Return zero-based index in the list of the first item whose value is equal to `x`. Raises a `ValueError` if there is no such item.

The optional arguments `start` and `end` are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the `start` argument.

`list.count(x)`

Listas

- Ejemplo

```
In [1]: lista = [1, 1.5, True, "texto", [1, 2]]
```

```
In [2]: lista[0]
```

```
Out[2]: 1
```

```
In [3]: lista[1]
```

```
Out[3]: 1.5
```

```
In [4]: lista[2]
```

```
Out[4]: True
```

```
In [5]: lista[3]
```

```
Out[5]: 'texto'
```

```
In [6]: lista[4]
```

```
Out[6]: [1, 2]
```

Operador []

Leer y modificar

Listas - métodos

```
In [7]: lista.append(23)
```

```
In [8]: lista
```

```
Out[8]: [1, 1.5, True, 'texto', [1, 2], 23]
```

```
In [10]: lista.insert(2,34.56)
```

```
In [11]: lista
```

```
Out[11]: [1, 1.5, 34.56, True, 'texto', [1, 2], 23]
```

```
In [12]: lista.pop(2)
```

```
Out[12]: 34.56
```

```
In [13]: lista
```

```
Out[13]: [1, 1.5, True, 'texto', [1, 2], 23]
```

```
In [14]: lista.append('nuevo texto')
```

```
In [15]: lista
```

```
Out[15]: [1, 1.5, True, 'texto', [1, 2], 23, 'nuevo texto']
```

```
In [16]: lista.remove('texto')
```

```
In [17]: lista
```

```
Out[17]: [1, 1.5, True, [1, 2], 23, 'nuevo texto']
```

```
In [18]: lista.remove(23)
```

```
In [19]: lista
```

```
Out[19]: [1, 1.5, True, [1, 2], 'nuevo texto']
```

Listas - métodos

```
In [24]: nueva_lista = [4,5,1,2,6,7,8,9,11]
```

```
In [25]: nueva_lista
```

```
Out[25]: [4, 5, 1, 2, 6, 7, 8, 9, 11]
```

```
In [26]: nueva_lista.sort()
```

```
In [27]: nueva_lista
```

```
Out[27]: [1, 2, 4, 5, 6, 7, 8, 9, 11]
```

```
In [29]: otra_lista = ['abcdef', 'abcde', 'abcd', 'abc', 'ab']
```

```
In [30]: otra_lista.sort(key=len)
```

```
In [31]: otra_lista = ['abcdef', 'abcde', 'abcd', 'abc', 'ab']
```

```
In [32]: otra_lista
```

```
Out[32]: ['abcdef', 'abcde', 'abcd', 'abc', 'ab']
```

```
In [33]: otra_lista.sort(key=len)
```

```
In [34]: otra_lista
```

```
Out[34]: ['ab', 'abc', 'abcd', 'abcde', 'abcdef']
```

Listas

```
In [38]: otra_lista[0:2]
```

```
Out[38]: ['ab', 'abc']
```

```
In [39]: otra_lista[2:4]
```

```
Out[39]: ['abcd', 'abcde']
```

```
In [40]: otra_lista
```

```
Out[40]: ['ab', 'abc', 'abcd', 'abcde', 'abcdef']
```


Listas | Ejercicios

- Ej1. Usando append
 - Guardar los resultados de los ejercicios WHILE en una lista
- Ej2. Dada una lista de strings, guarde en otra lista aquellos cuya longitud sea mayor a 3

Listas

```
import random

nuevaLista = []

for i in range(0,10):
    n = random.randint(1,20)
    nuevaLista.append(n)

print(nuevaLista)
```

Listas | Ejercicios

- Ej3. Dada una lista de N elementos aleatorios, remueva los elementos duplicados
- Ej4. Dada una lista de N elementos aleatorios, genere dos listas nuevas:
 - Impares
 - Pares

Tuplas

- Otro tipo de colección de datos
- Las tuplas son más “ligeras” que las listas
 - Colecciones básicas y ahorrar memoria

Tuplas

- Ejemplo

```
In [1]: tupla = (1,2,3)
```

```
In [2]: tupla[0]
```

```
Out[2]: 1
```

```
In [3]: tupla[0] = 5
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-3-a2f896436084>", line 1, in <module>  
    tupla[0] = 5
```

```
TypeError: 'tuple' object does not support item assignment
```

Tuplas

```
In [1]: tupla = 6, 7, 8, 15
```

```
In [2]: tupla
```

```
Out[2]: (6, 7, 8, 15)
```

```
In [3]: a, b, c, _ = tupla
```

```
In [4]: a
```

```
Out[4]: 6
```

```
In [5]: b
```

```
Out[5]: 7
```

```
In [6]: c
```

```
Out[6]: 8
```

```
In [7]:
```