



UNIVERSIDAD
DE MÁLAGA



Listas en Python

Unidad 8

Python para principiantes
dgm@uma.es



Programación

- **Algoritmos**

- Un conjunto de reglas o pasos para resolver un problema

+

- **Estructura de Datos**

- Una forma particular de organizar datos en un ordenador

<https://es.wikipedia.org/wiki/Algoritmo>

https://es.wikipedia.org/wiki/Estructura_de_datos

¿Qué no es una "Colección"?

La mayor parte de nuestras variables tienen un único valor en ellas
– cuando ponemos un nuevo valor en la variable, el antiguo valor es sobrescrito

```
$ python  
>>> x = 2  
>>> x = 4  
>>> print(x)  
4
```

Una lista es una clase de colección



- Una colección nos permite poner **muchos valores** en una sola “variable”
- Usar colecciones es interesante, porque podemos tener muchos valores en un solo "paquete".

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]
```

```
carryon = [ 'socks', 'shirt', 'perfume' ]
```

Constantes de Listas

- Las constantes de Listas se indican **entre corchetes []** y los elementos de la lista se separan por comas
- Un elemento de una lista puede ser cualquier objeto Python – incluso otra lista
- Una lista puede estar vacía

```
>>> print([1, 24, 76])  
[1, 24, 76]  
>>> print(['red', 'yellow',  
'blue'])  
['red', 'yellow', 'blue']  
>>> print(['red', 24, 98.6])  
['red', 24, 98.6]  
>>> print([ 1, [5, 6], 7])  
[1, [5, 6], 7]  
>>> print([])  
[]
```

¡Ya usábamos listas!

```
for i in [5, 4, 3, 2, 1] :  
    print(i)  
print('Despegue!')
```

5

4

3

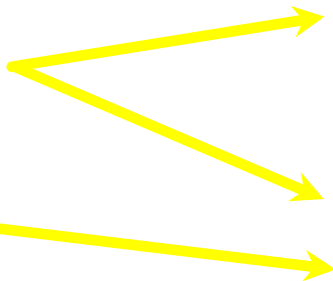
2

1

Despegue!

Listas y Bucles for – Los mejores amigos

```
friends = ['Joseph', 'Glenn', 'Sally']  
for friend in friends :  
    print('Feliz Año Nuevo:', friend)  
print('Hecho!')
```



Feliz Año Nuevo: Joseph
Feliz Año Nuevo : Glenn
Feliz Año Nuevo : Sally
Hecho!



Mirando dentro de las listas

Como en los strings, podemos acceder a cualquier elemento de una lista usando su **índice** expresado con corchetes

Joseph	Glenn	Sally
0	1	2

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]  
>>> print(friends[1])  
Glenn  
>>>
```


Las listas son mutables

- Los strings son “inmutables” – no podemos cambiar los contenidos de un string – debemos crear un nuevo string para hacer cualquier cambio
- Las listas son “mutables” – **podemos cambiar un elemento de una lista** usando su índice

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> x = fruit.lower()
>>> print(x)
banana
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

¿Cuántos elementos tiene una lista?

- La función **len()** toma una lista como parámetro, y retorna el **número de elementos** en la lista
- En realidad, len() nos dice el número de elementos de cualquier conjunto o secuencia (como los strings...)

```
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>>
```

Usando la función range

- La función **range** retorna una **lista de números** que varían desde 0 a uno menos de lo indicado en **range**
- Usando **list** y **range** podemos construir una lista de números
- Podemos construir un bucle usando **for** y **range**

```
>>> print(list(range(4)))  
[0, 1, 2, 3]  
>>> friends = ['Joseph', 'Glenn', 'Sally']  
>>> print(len(friends))  
3  
>>> print(list(range(len(friends))))  
[0, 1, 2]  
>>>
```

Una historia de 2 bucles...

```
friends = ['Joseph', 'Glenn', 'Sally']

for friend in friends :
    print('Feliz Año Nuevo:', friend)

for i in range(len(friends)) :
    friend = friends[i]
    print('Feliz Año Nuevo:', friend)
```

```
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(list(range(len(friends))))
[0, 1, 2]
>>>
```

Feliz Año Nuevo: Joseph
Feliz Año Nuevo: Glenn
Feliz Año Nuevo: Sally

Concatenando Listas con +

Podemos crear una nueva lista uniendo dos existentes

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Las listas pueden ser troceadas usando :

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41, 12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Recuerda: Como en los strings, el segundo número indica el final, **pero sin incluirlo**

Métodos de Listas

```
>>> x = list()
>>> type(x)
<class 'list'>
>>> dir(x)
['append', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
>>>
```

<http://docs.python.org/tutorial/datastructures.html>

Construyendo una Lista

- Podemos crear una lista vacía e ir sumando elementos con el método **append**
- La lista permanece en orden y los nuevos elementos son añadidos **al final de la lista**

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print(stuff)
['book', 99]
>>> stuff.append('cookie')
>>> print(stuff)
['book', 99, 'cookie']
```


¿Está algo en una lista?

- Python proporciona dos operadores para comprobar si un elemento está en una lista (**in** y **not in**)
- Estos operadores lógicos retornan True o False
- No modifican la lista

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```

Ordenando Listas

- Una lista puede contener muchos elementos y los mantiene en el orden en el que se han añadido hasta que hagamos algo que cambie este orden
- Una lista puede ser **ordenada (sort)** (es decir, cambiamos el orden)

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print(friends)
['Glenn', 'Joseph', 'Sally']
>>> print(friends[1])
Joseph
>>>
```

Borrando elementos de una lista

- Si sabemos la posición a borrar, podemos usar **pop**
- Si no necesitamos el elemento, podemos utilizar el operador **del**
- E incluso podemos indicar qué queremos borrar si no sabemos la posición usando **remove**

```
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> print(t)
['a', 'c']
>>> print(x)
b
```

```
>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> print(t)
['a', 'c']
```

```
>>> t = ['a', 'b', 'c']
>>> t.remove('b')
>>> print(t)
['a', 'c']
```

Funciones internas y Listas

- Hay varias funciones en Python que toman listas como parámetros
- ¿Recuerdas los bucles que construimos? ¡Así es mucho más simple!

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```

```
total = 0
count = 0
while True :
    inp = input('Escribe un número: ')
    if inp == 'fin' : break
    value = float(inp)
    total = total + value
    count = count + 1
```

```
average = total / count
print('Media:', average)
```

Escribe un número: 3
Escribe un número : 9
Escribe un número : 5
Escribe un número : fin
Media: 5.666666666667

```
numlist = list()
while True :
    inp = input('Escribe un número: ')
    if inp == 'fin' : break
    value = float(inp)
    numlist.append(value)
```

```
average = sum(numlist) / len(numlist)
print('Media:', average)
```

Mejores amigos: Strings y Listas

```
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print(stuff)
['With', 'three', 'words']
>>> print(len(stuff))
3
>>> print(stuff[0])
With
```

```
>>> print(stuff)
['With', 'three', 'words']
>>> for w in stuff :
...     print(w)
...
With
Three
Words
>>>
```

Split divide un string en partes y produce una **lista de strings**. Después, podemos acceder a una palabra en particular o iterar a través de todas las palabras.

```
>>> line = 'A lot of spaces'
>>> etc = line.split()
>>> print(etc)
['A', 'lot', 'of', 'spaces']
>>>
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print(thing)
['first;second;third']
>>> print(len(thing))
1
>>> thing = line.split(';')
>>> print(thing)
['first', 'second', 'third']
>>> print(len(thing))
3
>>>
```

Cuando no indicas un separador, se utilizan los espacios como separador, y múltiples espacios se tratan como uno solo

Se puede indicar **qué carácter** se quiere utilizar como **separador/delimitador** para usar en `split`

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From ') : continue
    words = line.split()
    print(words[2])
```

Sat
Fri
Fri
Fri
...

```
>>> line = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> words = line.split()
>>> print(words)
['From', 'stephen.marquard@uct.ac.za', 'Sat', 'Jan', '5', '09:14:16', '2008']
>>>
```


El patrón doble split

A veces usamos split, y después volvemos a aplicar split sobre uno de los “trozos”

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
words = line.split()  
email = words[1]  
print(email[1])
```

El patrón doble split

From **stephen.marquard@uct.ac.za** Sat Jan 5 09:14:16 2008

```
words = line.split()
```

```
email = words[1]
```

```
print(email[1])
```

```
stephen.marquard@uct.ac.za
```

El patrón doble split

From **stephen.marquard@uct.ac.za** Sat Jan 5 09:14:16 2008

```
words = line.split()  
email = words[1]  
pieces = email.split('@')  
print(pieces[1])
```

```
stephen.marquard@uct.ac.za  
['stephen.marquard', 'uct.ac.za']
```

Resumen

- Concepto de colección
- Listas y bucles for
- Indexando y buscando
- Mutabilidad de las listas
- Eliminando elementos
- Funciones: len, min, max, sum
- Troceando listas
- Métodos de listas: append, remove
- Ordenando listas
- Dividiendo strings en palabras
- Usando split para interpretar strings

Ejercicio 1

Abre el archivo romeo.txt y léelo línea a línea. Para cada línea, divide la línea en palabras usando el método `split()`. El programa debería construir una lista de palabras. Para cada palabra de cada línea, comprobar si ya estaba en la lista, y si no, añádela. Cuando el programa termine, ordenar la lista y mostrar las palabras resultantes en orden alfabético.

El archivo romeo.txt puede descargarse desde el Campus Virtual

Ejercicio 2 Abre el archivo mbox-short.txt y léelo línea a línea. Cuando encuentres una línea que comience con 'From ' como la siguiente:

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

trocea la línea usando split() e imprime la segunda palabra de la línea (es decir, la dirección de la persona que envió el mensaje). Al final, indica cuántas personas hubo (no es necesario tener en cuenta repeticiones).

Nota: asegúrate de no incluir las líneas que comiencen con 'From:'.

El archivo mbox-short.txt puede descargarse desde el Campus Virtual

Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Continue...

Initial Development: Charles Severance, University of Michigan School of Information

... Insert new Contributors and Translators here
Spanish Version: Daniel Garrido (dgm@uma.es)