Introducción a las listas

Python nos permite definir secuencias de valores de cualquier tipo. Por ejemplo, secuencias de números o de símbolos. A esto se le llama *lista*. La lista viene determinada por una serie de valores encerrados entre corchetes y separados por comas.

```
In [2]: a =  [2, 3, 4]
```

Los elementos de una lista pueden ser cadenas de caracteres o incluso expresiones que se calculan en tiempo real

```
In [3]: b = ['hola', 'adios']
```

```
In [4]: c = [1, 1+3, 6/2]
```

```
In [5]: c
```

```
Out[5]: [1, 4, 3]
```

Una lista puede no tener elementos: *la lista vacía*

```
In [6]: void = []
```

```
In [7]: void
```

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

Operaciones básicas

Longitud: podemos calcular la longitud de una cadena con la orden *len*

```
In [8]: len(a)
```

```
Out[8]: 3
```

```
In [9]: len(void)
```

```
Out[9]: 0
```

Concatenación: podemos concatenar cadenas con el operador **+**

```
In [10]: a = a + b
```

```
In [11]: a
```

```
Out[11]: [2, 3, 4, 'hola', 'adios']
```

Repetición: podemos repetir cadenas con el operador *

```
In [12]: e = a*2
```

In [13]:
```
e
```

Out[13]: [2, 3, 4, 'hola', 'adios', 2, 3, 4, 'hola', 'adios']

Podemos acceder a cada elemento de la lista a través de su índice. *Recuerda que los índices siempre empiezan a contar en 0*

In [14]:
```
a
```

Out[14]: [2, 3, 4, 'hola', 'adios']

In [15]:
```
a[0]
```

Out[15]: 2

In [16]:
```
a[4]
```

Out[16]: 'adios'

In [17]:
```
f = [a[0], a[4]]
```

In [18]:
```
f
```

Out[18]: [2, 'adios']

Los índices pueden ser *negativos*.

In [25]:
```
z = range(4,10)
z
```

Out[25]: [4, 5, 6, 7, 8, 9]

In [26]:
```
z[-1]
```

Out[26]: 9

In [27]:
```
z[-6]
```

Out[27]: 4

Operador de corte: Podemos quedarnos con un trozo de una lista

In [31]:
```
a = [1, 2, 3, 45, 66]
```

In [32]:
```
b = a[1:4]
b
```

Out[32]: [2, 3, 45]

In [33]:
```
b = a[1:]
b
```

Out[33]: [2, 3, 45, 66]

In [34]:
```
b = a[:4]
b
```

Out[34]: [1, 2, 3, 45]

In [35]:
```
b = a[:]
b
```

Out[35]: [1, 2, 3, 45, 66]

Podemos utilizar *índices negativos*

In [36]:
```
c = a[-5:-1]
c
```

Out[36]: [1, 2, 3, 45]

In [37]:
```
#qué significa esto
b == a
```

Out[37]: True

In [38]:
```
b is a
```

Out[38]: False

El bucle **for-in** recorre los elementos de una lista.

In [39]:
```
s = ''
for n in a:
    s = s + str(n)
s
```

Out[39]: '1234566'

range es un caso particular de lista

In [40]:
```
range(20)
```

Out[40]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

In [41]:
```
a = a + range(10)
```

In [42]:
```
a
```

Out[42]: [1, 2, 3, 45, 66, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [43]:
```
s = ''
for n in a:
    s = s + str(n)
s
```

Out[43]: '12345660123456789'

Comparación de listas Las listas se pueden comparar con los operaciones habituales: ==, !=, <=, ...

In [44]:
```
a = [1, 2, 3]
b = [2, 4]
a == b
```

Out[44]: False

In [45]:
```
a != b
```

Out[45]: True

In [46]:
```
a < b
```

Out[46]: True

In [47]:
```
c = a
c
```

Out[47]: [1, 2, 3]

In [48]:
```
a == c
```

Out[48]: True

In [49]:
```
d = a[:]
```

In [50]:
```
c == d
```

Out[50]: True

In [51]:
```
c is d
```

Out[51]: False

Modificación de elementos de una lista. Mutabilidad. El operador is

In [52]:
```
#Dada una lista, podemos modificar sus componentes
a = range(4,10)
a
```

Out[52]: [4, 5, 6, 7, 8, 9]

In [53]:
```
a[2] = 99
a
```

Out[53]: [4, 5, 99, 7, 8, 9]

In [54]:
```
'''
EJEMPLO: Elimina los elementos negativos de una lista,
sustituyéndolos por 0
'''
b = [1, 3, -4, 5, -2, 3, -9]
for i in range(len(b)):
    if b[i] < 0:
        b[i] = 0
b
```

Out[54]: [1, 3, 0, 5, 0, 3, 0]

Mutabilidad, inmutabilidad --> de qué me estás hablando?

In [55]:
```
a = range(4,10)
a
```

Out[55]: [4, 5, 6, 7, 8, 9]

In [57]:
```
b = a
b
```

Out[57]: [4, 5, 6, 7, 8, 9]

In [58]:
```
a[3] = 99
a
```

Out[58]: [4, 5, 6, 99, 8, 9]

In [59]:
```
b
```

Out[59]: [4, 5, 6, 99, 8, 9]

Esto es un poco raro, no?

In [60]:
```
c = a[:]
c
```

Out[60]: [4, 5, 6, 99, 8, 9]

In [61]:
```
c[3] = 44
c
```

Out[61]: [4, 5, 6, 44, 8, 9]

In [62]:
```
a
```

Out[62]: [4, 5, 6, 99, 8, 9]

In [63]:
```
b
```

Out[63]: [4, 5, 6, 99, 8, 9]

¿Por qué pasa esto?

In [64]:
```
c = a
b = a[:]
```

In [65]:
```
print a == c, a == b
```

```
True True
```

In [66]:
```
print a is c, a is b
```

```
True False
```

Las cadenas de caracteres se comportan de forma parecida, pero no igual

In [67]:
```
s = 'hola'
s[0]
```

Out[67]:
```
'h'
```

In [68]:
```
s[0] = 'j'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/home/jesus/Dropbox/docencia13-14/ipython/<ipython-input-68-a225757de94d> in <module>()
----> 1 s[0] = 'j'

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

Añadir elementos a una lista

Podemos añadir elementos a una lista de dos formas diferentes. Una manera es utilizando el operador de concatenación. Esto crea una lista nueva con el nuevo elemento.

In [69]:
```
a = range(4,10)
b = a
```

In [70]:
```
a = a + [99]
print a, b
```

```
[4, 5, 6, 7, 8, 9, 99] [4, 5, 6, 7, 8, 9]
```

In [71]:
```
print a == b, a is b
```

```
False False
```

También podemos añadir elementos con el método *append*. De esta forma se modifica la lista original, sin crear una lista nueva.

In [72]:
```
c = b
c
```

Out[72]:
```
[4, 5, 6, 7, 8, 9]
```

In [73]:
```
c.append(99)
print c, b
```

```
[4, 5, 6, 7, 8, 9, 99] [4, 5, 6, 7, 8, 9, 99]
```

In [74]:
```
print c == b, c is b
```

```
True True
```

EJEMPLO: Crea una lista con los cuadrados de los 10 primeros números.

In [76]:
```
L = []
i = 1
while i <= 10:
    L.append(i*i)
    i+=1
L
```

Out[76]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Ejemplos

Recorridos de listas. Utilización en funciones

In [91]:
```
'''
Si a es una lista de números, escribe un programa
que modifique la lista de forma que cada
componente sea igual al cuadrado del componente original.
'''

def cuadrado_lista (a):
    '''
    Eleva al cuadrado los elementos de una lista
    '''
    for i in range(len(a)):
        a[i] = a[i]**2

#No hace falta poner return: modifico la propia lista
```

In [92]:
```
lista = range(10)+range(3)
lista
```

Out[92]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2]

In [93]:
```
cuadrado_lista(lista)
lista
```

Out[93]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 0, 1, 4]

In [94]:
```
#podemos representar mejor la lista
def dibuja(lista):
    # presenta por pantalla una lista
    k = len(lista)
    for i in range(k-1):
        print lista[i], ',',
    print lista[k-1]
```

In [95]:
```
dibuja(lista)
```

```
0 , 1 , 4 , 9 , 16 , 25 , 36 , 49 , 64 , 81 , 0 , 1 , 4
```

In [96]:
```
# creación de una lista aleatoria
def random_list(size):
    #Creo una lista aleatoria
    from random import randint
    # randint(a, b) devuelve un entero aleatorio N,
    #a <= N <= b
    data = []
    for i in range(size):
        data.append(randint(-10,10))
    return data
```

In [100]:
```
milista = random_list(15)
dibuja(milista)
```

```
4 , -4 , 8 , 10 , 9 , 10 , -2 , -2 , -10 , -3 , -5 , 6 , -2 , 0 , 2
```

In [101]:
```
cuadrado_lista(milista)
dibuja(milista)
```

```
16 , 16 , 64 , 100 , 81 , 100 , 4 , 4 , 100 , 9 , 25 , 36 , 4 , 0 , 4
```

In [ ]: