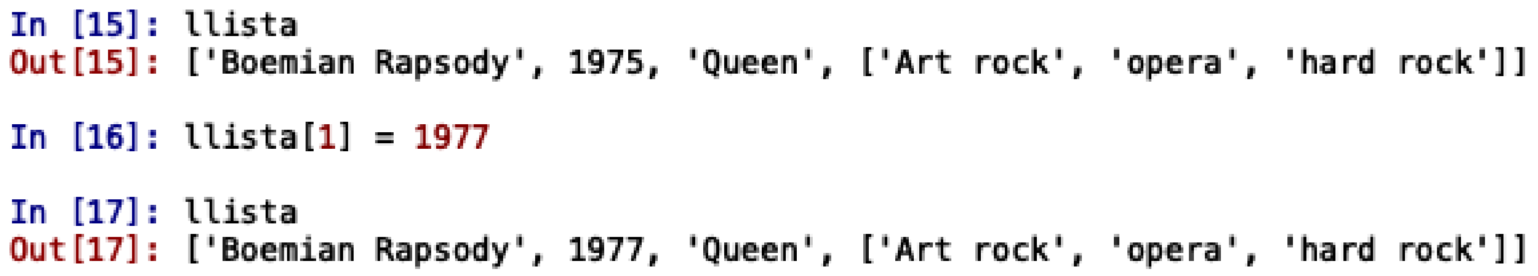
L9a: Advanced data structures

Lists

Lists: Summary

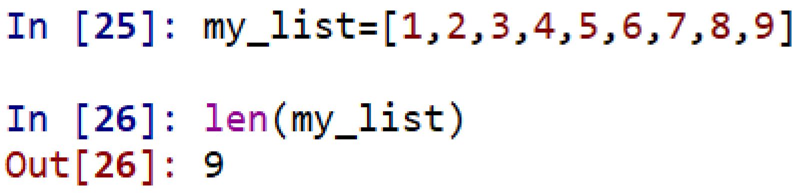
- A list is an **ordered** sequence of values, where each value is identified by an **index**.
- The elements in the list can be **heterogeneous**.
- Of variable length, and nested (lists of lists).
- Lists are **mutable.** We access to the position with operator **[ ]** and assign the new value with operator **=** :



```
In [15]: llista
Out[15]: ['Boemian Rapsody', 1975, 'Queen', ['Art rock', 'opera', 'hard rock']]

In [16]: llista[1] = 1977

In [17]: llista
Out[17]: ['Boemian Rapsody', 1977, 'Queen', ['Art rock', 'opera', 'hard rock']]
```

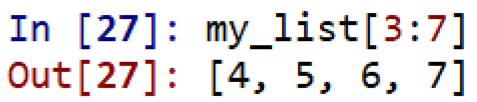
Lists: Summary (Length, cut, membership)

`len(l)` returns the length of the list:



```
In [25]: my_list=[1,2,3,4,5,6,7,8,9]

In [26]: len(my_list)
Out[26]: 9
```

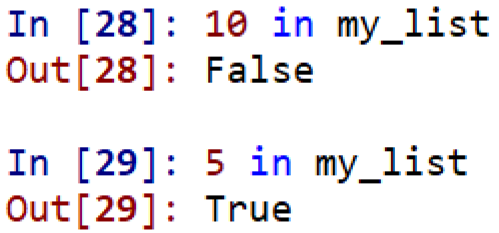
The operator `[n:m]` returns the sub-list from the n-th element (included) to the m-th (not included):



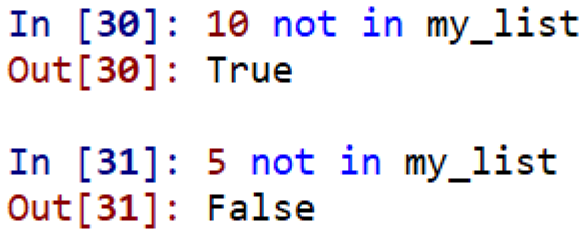
```
In [27]: my_list[3:7]
Out[27]: [4, 5, 6, 7]
```

The operator `in` checks if an item is in the list. It returns a Boolean.

It can be combined with operator `not`.



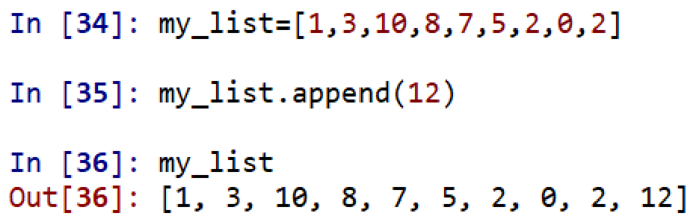
```
In [28]: 10 in my_list
Out[28]: False

In [29]: 5 in my_list
Out[29]: True
```



```
In [30]: 10 not in my_list
Out[30]: True

In [31]: 5 not in my_list
Out[31]: False
```

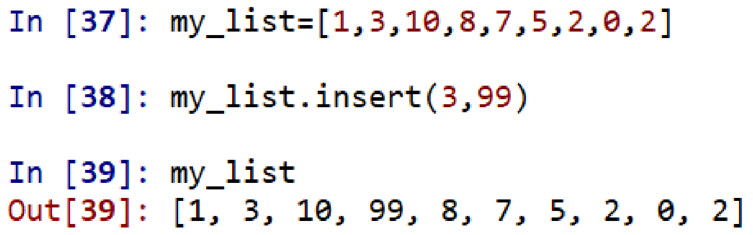
Lists: Summary (Addition)

- To add an item at the end of the list we use `append`.



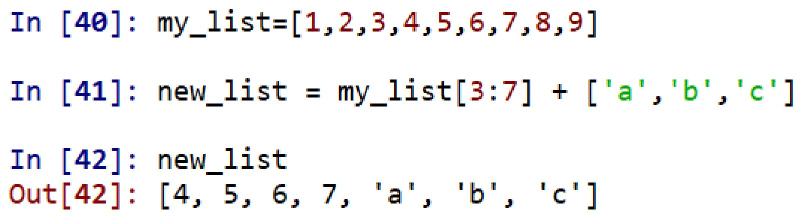
```
In [34]: my_list=[1,3,10,8,7,5,2,0,2]

In [35]: my_list.append(12)

In [36]: my_list
Out[36]: [1, 3, 10, 8, 7, 5, 2, 0, 2, 12]
```

- To add an element x at a certain position i, we use `insert(i,x)`



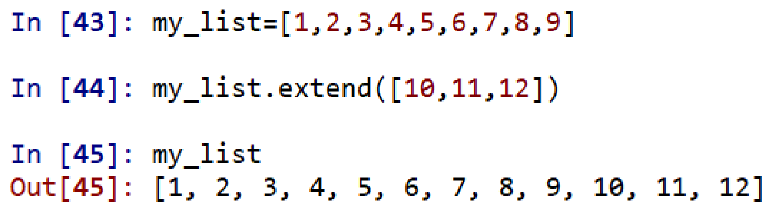
```
In [37]: my_list=[1,3,10,8,7,5,2,0,2]

In [38]: my_list.insert(3,99)

In [39]: my_list
Out[39]: [1, 3, 10, 99, 8, 7, 5, 2, 0, 2]
```

- We can concatenate two or more lists with the operator +, returning a new list.



```
In [40]: my_list=[1,2,3,4,5,6,7,8,9]

In [41]: new_list = my_list[3:7] + ['a','b','c']

In [42]: new_list
Out[42]: [4, 5, 6, 7, 'a', 'b', 'c']
```

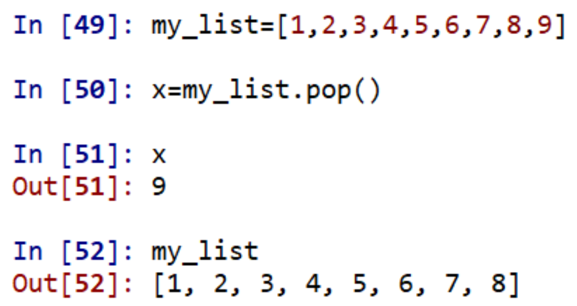
- To concatenate we also have `extend`.



```
In [43]: my_list=[1,2,3,4,5,6,7,8,9]

In [44]: my_list.extend([10,11,12])

In [45]: my_list
Out[45]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

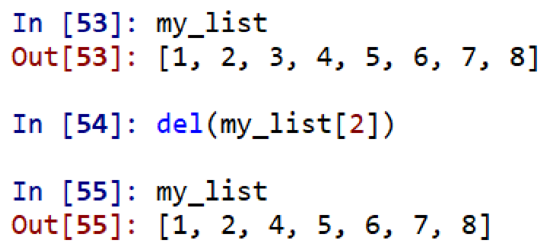
Lists: Summary (Delete)

- To remove an item at the end of the list we have `pop()`. It returns the deleted item and changes the list



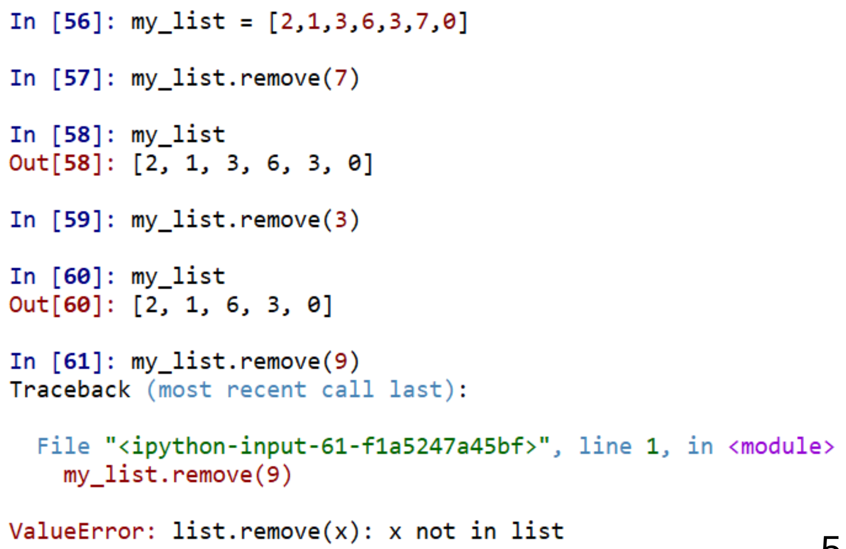
```
In [49]: my_list=[1,2,3,4,5,6,7,8,9]

In [50]: x=my_list.pop()

In [51]: x
Out[51]: 9

In [52]: my_list
Out[52]: [1, 2, 3, 4, 5, 6, 7, 8]
```

- To remove an item from a certain position in the list we function `del`



```
In [53]: my_list
Out[53]: [1, 2, 3, 4, 5, 6, 7, 8]

In [54]: del(my_list[2])

In [55]: my_list
Out[55]: [1, 2, 4, 5, 6, 7, 8]
```

- To remove an item with a certain value we have remove:
  - it searches the item and removes it
  - if the item appears multiple times, it only removes the first occurrence
  - if the item is not in the list, it generates an error



```
In [56]: my_list = [2,1,3,6,3,7,0]

In [57]: my_list.remove(7)

In [58]: my_list
Out[58]: [2, 1, 3, 6, 3, 0]

In [59]: my_list.remove(3)

In [60]: my_list
Out[60]: [2, 1, 6, 3, 0]

In [61]: my_list.remove(9)
Traceback (most recent call last):

  File "<ipython-input-61-f1a5247a45bf>", line 1, in <module>
    my_list.remove(9)

ValueError: list.remove(x): x not in list
```

List: index

The `index()` returns the index of the specified element in the list

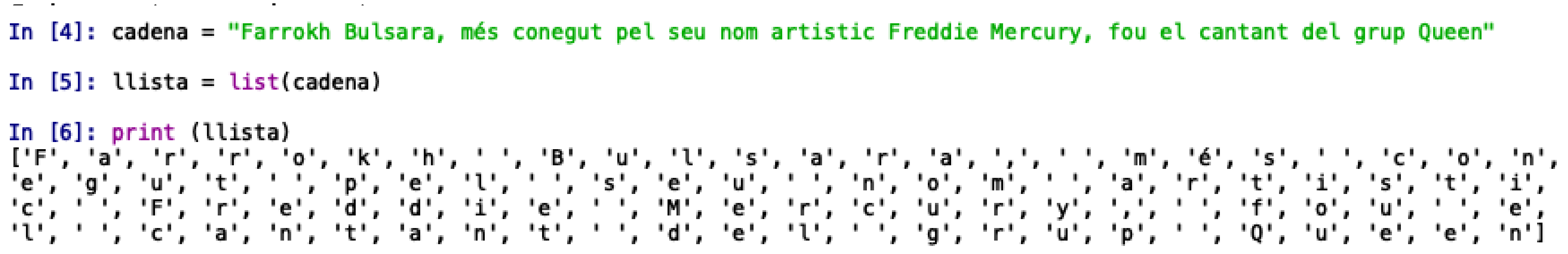
```
animals = ['cat', 'dog', 'rabbit', 'horse']

# get the index of 'dog’
myindex = animals.index('dog')

print(myindex)
# Output: 1
```

If the element is not found, a ValueError exception is raised.

Note: The `index()` method only returns the first occurrence of the matching element.

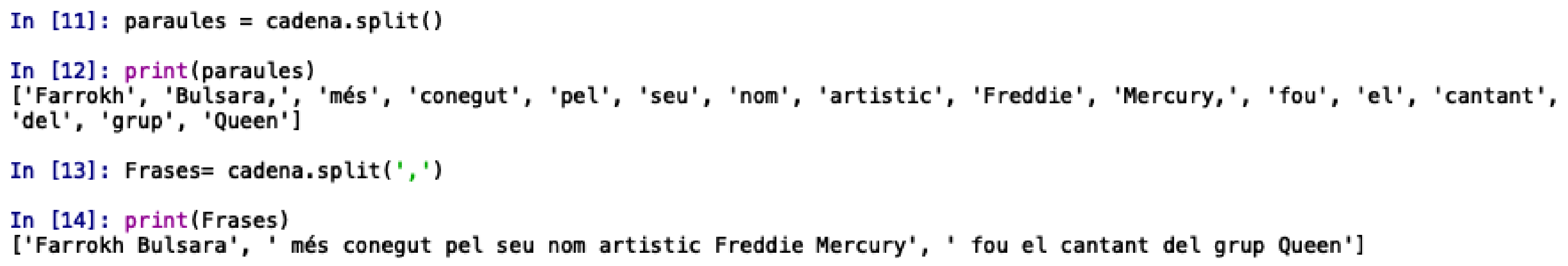
Operations with Lists: Conversion string ⇒ list

- To convert a string to a list, we use the function



```
In [4]: cadena = "Farrokh Bulsara, més conegut pel seu nom artistic Freddie Mercury, fou el cantant del grup Queen"

In [5]: llista = list(cadena)

In [6]: print (llista)
['F', 'a', 'r', 'r', 'o', 'k', 'h', ' ', 'B', 'u', 'l', 's', 'a', 'r', 'a', ',', ' ', 'm', 'é', 's', ' ', 'c', 'o', 'n',
'e', 'g', 'u', 't', ' ', 'p', 'e', 'l', ' ', 's', 'e', 'u', ' ', 'n', 'o', 'm', ' ', 'a', 'r', 't', 'i', 's', 't', 'i',
'c', ' ', 'F', 'r', 'e', 'd', 'd', 'i', 'e', ' ', 'M', 'e', 'r', 'c', 'u', 'r', 'y', ',', ' ', 'f', 'o', 'u', ' ', 'e',
'l', ' ', 'c', 'a', 'n', 't', 'a', 'n', 't', ' ', 'd', 'e', 'l', ' ', 'g', 'r', 'u', 'p', ' ', 'Q', 'u', 'e', 'e', 'n']
```

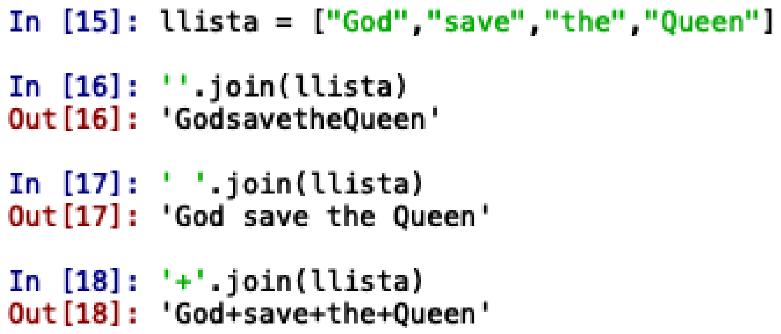
- The `split('character')` method divides a string into a list of sub-strings using a character (separator) given as a parameter. If no parameter is passed, it is divided by blank spaces:



```
In [11]: paraules = cadena.split()

In [12]: print(paraules)
['Farrokh', 'Bulsara,', 'més', 'conegut', 'pel', 'seu', 'nom', 'artistic', 'Freddie', 'Mercury,', 'fou', 'el', 'cantant',
'del', 'grup', 'Queen']

In [13]: Frases= cadena.split(',')

In [14]: print(Frases)
['Farrokh Bulsara', ' més conegut pel seu nom artistic Freddie Mercury', ' fou el cantant del grup Queen']
```

Operations with Lists: Conversion list ⇒ string

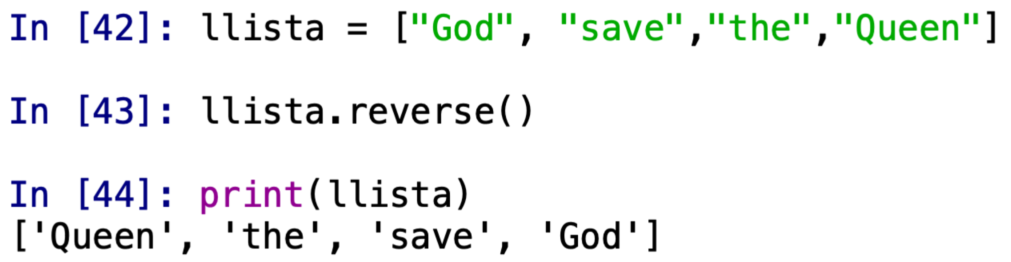
- To convert a list of characters to a string we can use the method `join(llista)`:



```
In [15]: llista = ["God","save","the","Queen"]

In [16]: ''.join(llista)
Out[16]: 'GodsavetheQueen'

In [17]: ' '.join(llista)
Out[17]: 'God save the Queen'

In [18]: '+'.join(llista)
Out[18]: 'God+save+the+Queen'
```

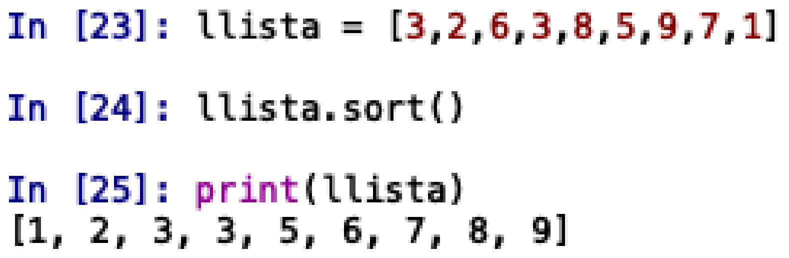
Operations with Lists: Sorting

- To convert a list in reverse order we can use the method `reverse()`:

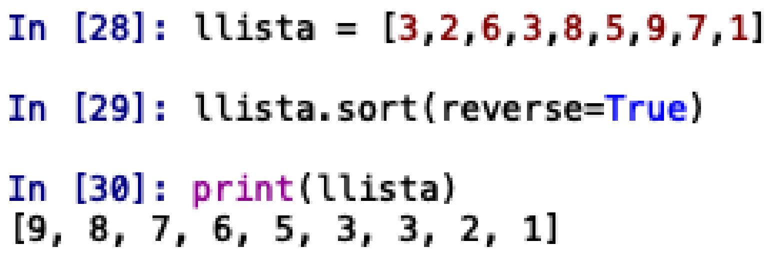


```
In [42]: llista = ["God", "save","the","Queen"]

In [43]: llista.reverse()

In [44]: print(llista)
['Queen', 'the', 'save', 'God']
```

- The method `sort(reverse,key)` sorts (orders) a list. With the parameter `reverse` we can revert the order:



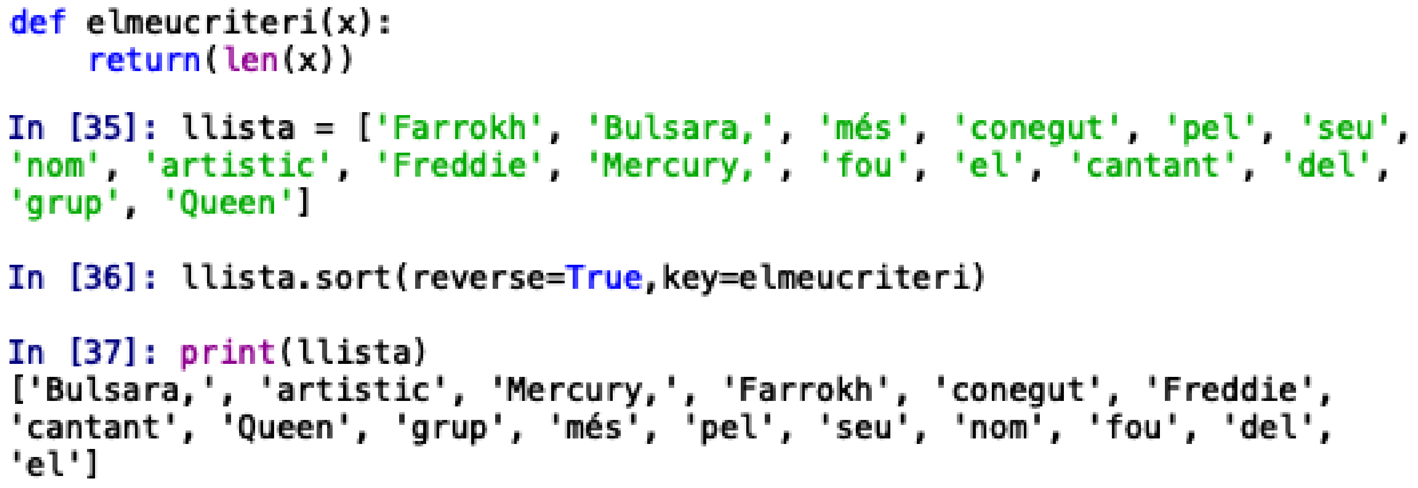
```
In [23]: llista = [3,2,6,3,8,5,9,7,1]

In [24]: llista.sort()

In [25]: print(llista)
[1, 2, 3, 3, 5, 6, 7, 8, 9]
```



```
In [28]: llista = [3,2,6,3,8,5,9,7,1]

In [29]: llista.sort(reverse=True)

In [30]: print(llista)
[9, 8, 7, 6, 5, 3, 3, 2, 1]
```

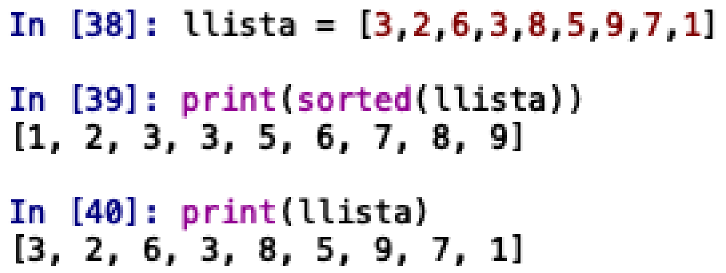
Operations with Lists: Sorting

- The method `sort(reverse,key)` sorts a list. If we define a `key` function, we can define our own sorting criteria:



```
def elmeucriteri(x):
    return(len(x))

In [35]: llista = ['Farrokh', 'Bulsara,', 'més', 'conegut', 'pel', 'seu',
'nom', 'artistic', 'Freddie', 'Mercury,', 'fou', 'el', 'cantant', 'del',
'grup', 'Queen']

In [36]: llista.sort(reverse=True,key=elmeucriteri)

In [37]: print(llista)
['Bulsara,', 'artistic', 'Mercury,', 'Farrokh', 'conegut', 'Freddie',
'cantant', 'Queen', 'grup', 'més', 'pel', 'seu', 'nom', 'fou', 'del',
'el']
```

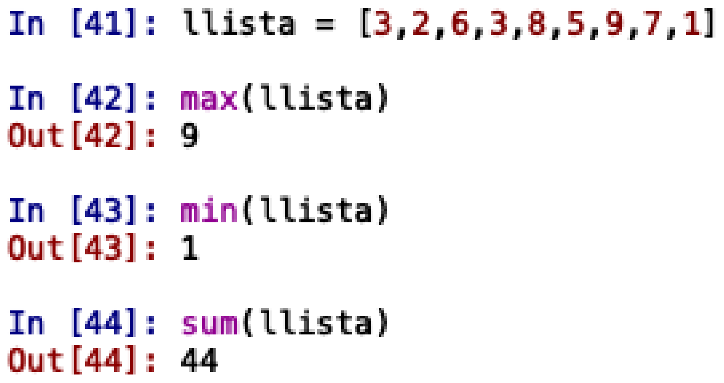
- Sorting methods **mutate** the list. If we do not want to change the list, we can use the function `sorted()`:



```
In [38]: llista = [3,2,6,3,8,5,9,7,1]

In [39]: print(sorted(llista))
[1, 2, 3, 3, 5, 6, 7, 8, 9]

In [40]: print(llista)
[3, 2, 6, 3, 8, 5, 9, 7, 1]
```

Other operations with lists

- The functions `sum(l)`, `max(l)`, and `min(l)`, applied to lists of numbers, are used to return the sum, the maximum and the minimum of the elements of the list:



```
In [41]: llista = [3,2,6,3,8,5,9,7,1]

In [42]: max(llista)
Out[42]: 9

In [43]: min(llista)
Out[43]: 1

In [44]: sum(llista)
Out[44]: 44
```

Example: Lyrics2list

- Make a program that converts the lyrics of a song into a list of words it is made of.
- The list of words:
  - must not contain any punctuation marks, such as ?!()#';,:.
  - all words must be written in lower case

```
bohemian ="Is this the real life? Is this just fantasy? Caught in a landslide, No escape from
reality. Open your eyes, Look up to the skies and see, I'm just a poor boy, I need no sympathy,
Because I'm easy come, easy go, Little high, little low, Any way the wind blows doesn't really
matter to me, to me. Mama, just killed a man, Put a gun against his head, Pulled my trigger, now
he's dead. Mama, life had just begun, But now I've gone and thrown it all away. Mama, ooh, Didn't
mean to make you cry, If I'm not back again this time tomorrow, Carry on, carry on as if nothing
really matters. Too late, my time has come, Sends shivers down my spine, Body's aching all the
time. Goodbye, everybody, I've got to go, Gotta leave you all behind and face the truth. Mama, ooh
(any way the wind blows), I don't wanna die, I sometimes wish I'd never been born at all. I see a
little silhouetto of a man, Scaramouche, Scaramouche, will you do the Fandango? Thunderbolt and
lightning, Very, very frightening me. (Galileo) Galileo. (Galileo) Galileo, Galileo Figaro
Magnifico-o-o-o-o. I'm just a poor boy, nobody loves me. He's just a poor boy from a poor family,
Spare him his life from this monstrosity. Easy come, easy go, will you let me go? Bismillah! No, we
will not let you go. (Let him go!) Bismillah! We will not let you go. (Let him go!) Bismillah! We
will not let you go. (Let me go!) Will not let you go. (Let me go!) Never let you go (Never, never,
never, never let me go) Oh oh oh oh No, no, no, no, no, no, no Oh, mama mia, mama mia (Mama mia,
let me go.) Beelzebub has a devil put aside for me, for me, for me. So you think you can stone me
and spit in my eye? So you think you can love me and leave me to die? Oh, baby, can't do this to
me, baby, Just gotta get out, just gotta get right outta here. (Ooooh, ooh yeah, ooh yeah) Nothing
really matters, Anyone can see, Nothing really matters, Nothing really matters to me. Any way the
wind blows."
```

Example: Lyrics2list (v1)

```
bohemian = "Is this the real life? Is this just fantasy? Caught in a landslide, No escape from reality. Open your eyes, Look up to the skies and see, I'm just a poor boy, I need no sympathy…"

aux=[]
for c in bohemian:
   if ('A' <= c <='Z'):
       aux.append(chr(ord(c)+32))
   else:
       if ((c >= 'a' and c <= 'z') or c == ' '):
             aux.append(c)

aux = "".join(aux)
out = aux.split()
```

Example: Lyrics2list (v2)

```
bohemian = "Is this the real life? Is this just fantasy? Caught in a landslide, No escape from reality. Open your eyes, Look up to the skies and see, I'm just a poor boy, I need no sympathy…"

aux=[]
for c in bohemian:
   if (('A' <= c <='Z') or ('a' <= c <= 'z') or c == ' '):
      aux.append(c)

aux = "".join(aux)
aux = aux.lower()
out = aux.split()
```

Functions with strings	Action
s.lower()	Convert to lowercase
s.upper()	Convert to uppercase
s.count(c)	Counts how many **c** appear in the string
t=s.replace(s1,s2)	Returns a string in which substring s1 is replaced by s2
s.find(c)	Search for c in the string (the first occurrence)

List comprehension

- **List comprehensions** allow to create lists in an elegant way by simplifying the code as much as possible
- Special syntax (the result is always a list)

```
newlist = [expression for variable in list if condition]
```

Example: given a list of integers, create a list of even numbers

```
numbers = [1, 2, 34, 86, 4, 5, 99, 890, 45]
```

```
even_nums = []
for num in numbers:
    if num % 2 == 0:
        even_nums.append(num)
print(even_nums)
```

Equivalent code

```
even_nums = [num for num in numbers if num%2 ==0]
```

List comprehension

- **List comprehensions** allow to create lists in an elegant way by simplifying the code as much as possible
- Special syntax (the result is always a list)

```
newlist = [expression for variable in list if condition]
```

Example: given a list of integers, return the square root of numbers that are greater than or equal to 0

```
l=[1, 9, -1, -4, 16, -2, 4]

l2 = [x**.5 for x in l if x>=0]
```

List comprehension

- **List comprehensions** can be nested

```
newlist = [expression for variable in list if condition]
```

Example: given a list, first add 2 to each element greater than 10, and then multiply by 5 those that are odd numbers.

```
l=[2, 5, 23, 12, 45, 29, 5, 10]

   Intermediate result [x+2 for x in l if x>10]

l2 = [x*5 for x in [x+2 for x in l if x>10] if x%2==1]

Output→
Intermediate result [x+2 for x in l if x>10] → [25, 14, 47, 31]
Final result: [125, 235, 155]
```

List comprehension

- **List comprehensions** can have nested loops (e.g. `for`)

```
newlist = [expression for variable in list if condition]
```

Example: given two lists, create combinations.

```
expressions = ['Hello', 'Greetings', 'Good bye']
names = ['Peter', 'Antoine', 'Mario']
frases = [expres+' '+name for expres in expressions
for name in names]
print(frases)
```

equivalent

```
for expres in expressions:
        for name in names:
                l.append(expres+" "+name)
```

```
Output → ['Hello Peter', 'Hello Antoine', 'Hello Mario',
'Greetings Peter', 'Greetings Antoine', 'Greetings Mario',
'Good bye Peter', 'Good bye Antoine', 'Good bye Mario']
```