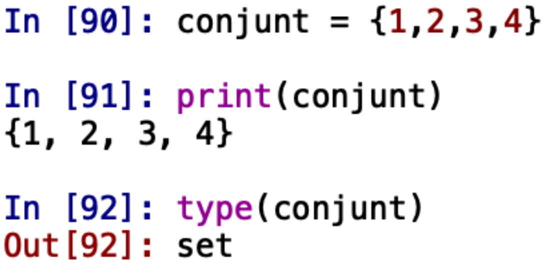
L9c: Advanced data structures

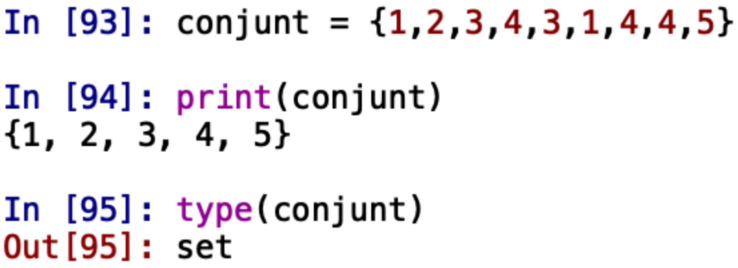
Sets

Sets: Definition

- A set is an **unordered collection** of values.
- Values are **unique**, no repeated items.
- We cannot identify each element of the set (**no index**).
- It is a set in the mathematical sense of the term.
- The syntax for expressing literals of type set is written as a **comma**-separated list of elements, inside **braces**.



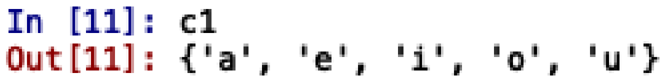
```
In [90]: conjunt = {1,2,3,4}

In [91]: print(conjunt)
{1, 2, 3, 4}

In [92]: type(conjunt)
Out[92]: set
```



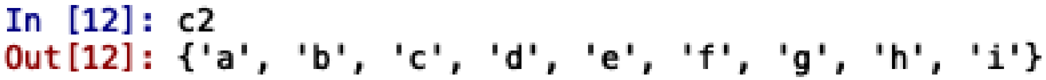
```
In [93]: conjunt = {1,2,3,4,3,1,4,4,5}

In [94]: print(conjunt)
{1, 2, 3, 4, 5}

In [95]: type(conjunt)
Out[95]: set
```

Operations

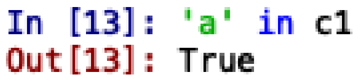
- **Membership** (`in`)
- **Difference** (-) Elements are in one set but not in the other one.
- **Or** (|) Elements present in one set or in the other. The union in a mathematical sense.
- **And** (`&`) Elements present in one set and in the other. The intersection in the mathematical sense.
- **Xor** (`^`) Elements present in one set or the other, but not in both.



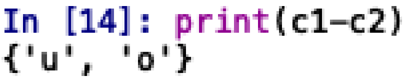
```
In [11]: c1
Out[11]: {'a', 'e', 'i', 'o', 'u'}
```



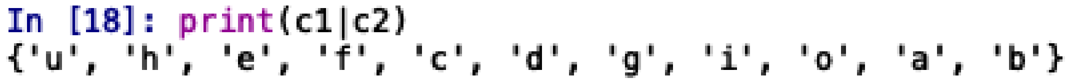
```
In [12]: c2
Out[12]: {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'}
```



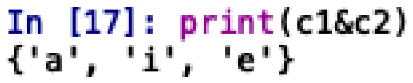
```
In [13]: 'a' in c1
Out[13]: True
```



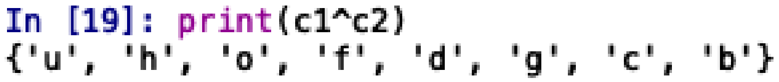
```
In [14]: print(c1-c2)
{'u', 'o'}
```



```
In [18]: print(c1|c2)
{'u', 'h', 'e', 'f', 'c', 'd', 'g', 'i', 'o', 'a', 'b'}
```



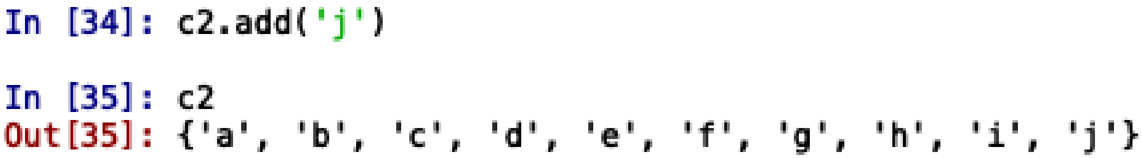
```
In [17]: print(c1&c2)
{'a', 'i', 'e'}
```



```
In [19]: print(c1^c2)
{'u', 'h', 'o', 'f', 'd', 'g', 'c', 'b'}
```

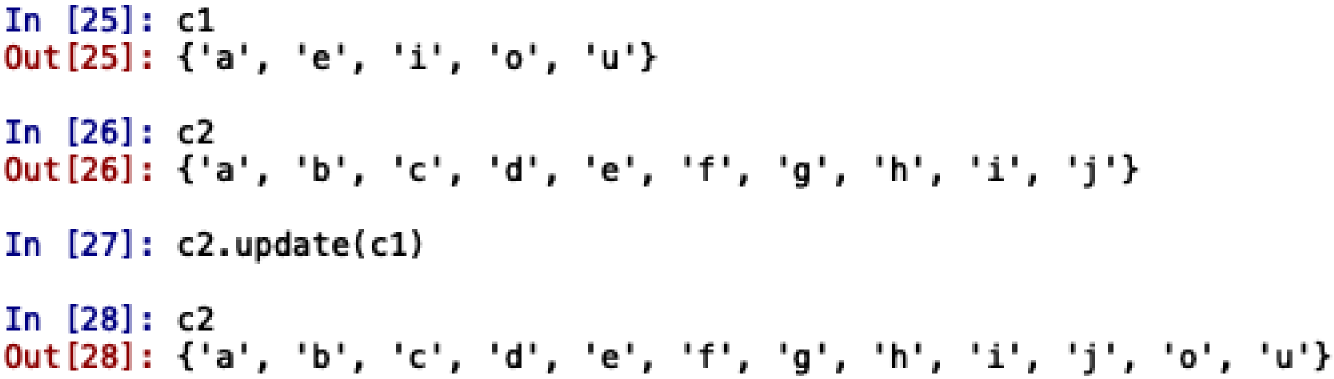
Modification of sets: Add and Remove

- To add an element to the set, we use `add`:



```
In [34]: c2.add('j')

In [35]: c2
Out[35]: {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}
```

- To add a set to another set, we use `Update`:



```
In [25]: c1
Out[25]: {'a', 'e', 'i', 'o', 'u'}

In [26]: c2
Out[26]: {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}

In [27]: c2.update(c1)

In [28]: c2
Out[28]: {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'o', 'u'}
```

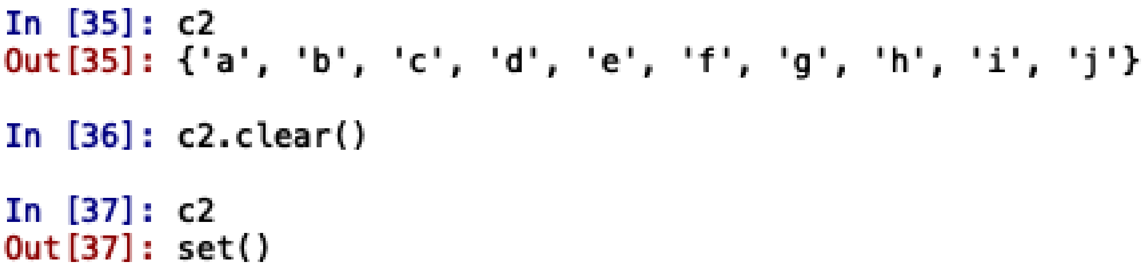
- To remove an element from a set, we use `remove`:

But if the element does not exist, it will raise an error

```
In [29]: c2.remove('j')

In [30]: c2
Out[30]: {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'o', 'u'}
```

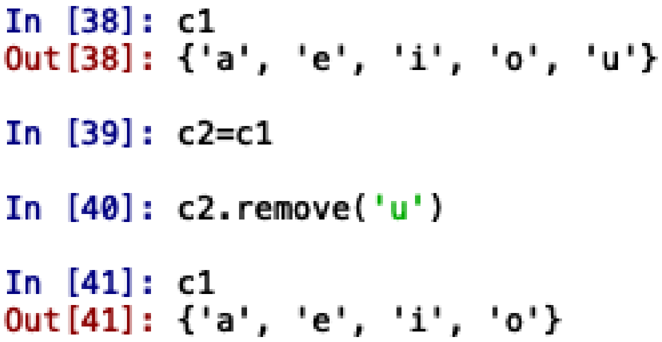
Other operations

- To empty a set, we use `clear`:

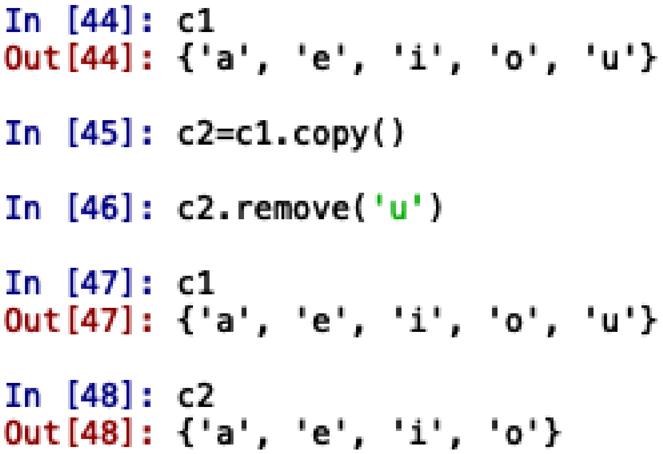


```
In [35]: c2
Out[35]: {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'}

In [36]: c2.clear()

In [37]: c2
Out[37]: set()
```

- Sets are mutable. To make a copy, we use the method `copy`:



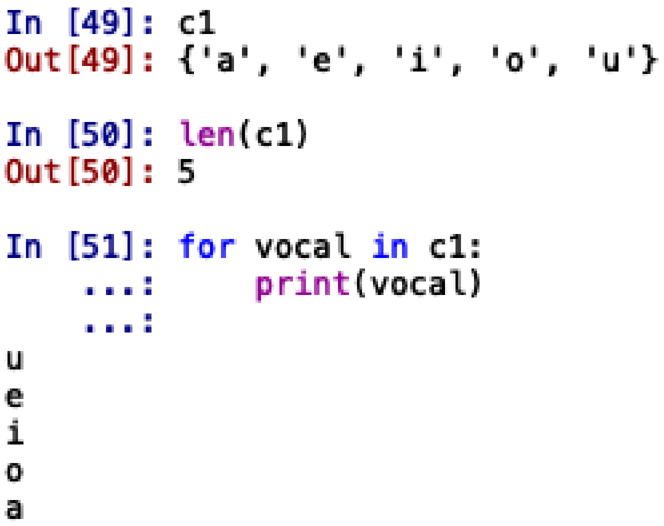
```
In [38]: c1
Out[38]: {'a', 'e', 'i', 'o', 'u'}

In [39]: c2=c1

In [40]: c2.remove('u')

In [41]: c1
Out[41]: {'a', 'e', 'i', 'o'}
```



```
In [44]: c1
Out[44]: {'a', 'e', 'i', 'o', 'u'}

In [45]: c2=c1.copy()

In [46]: c2.remove('u')

In [47]: c1
Out[47]: {'a', 'e', 'i', 'o', 'u'}

In [48]: c2
Out[48]: {'a', 'e', 'i', 'o'}
```

Iteration

- Sets are iterable. So, we can use `len` and `for`:



```
In [49]: c1
Out[49]: {'a', 'e', 'i', 'o', 'u'}

In [50]: len(c1)
Out[50]: 5

In [51]: for vocal in c1:
    ...:     print(vocal)
    ...:
u
e
i
o
a
```

- They are unordered objects, so they do not support sequence operations such as indexing or the cut operator `[n:m]`

Exercise: Unique words

- This problem is a continuation of Problem *Lyrics2list* (converts the lyrics of a song into a list of words)

- We want to know how many unique words a text has.

- Create a function called `Uniques` that receives a list of strings as a parameter. The function must return a list in which repeated strings have been removed.

- We want to know how many different words the Queen's Bohemian Rhapsody song has

Exercise: Unique words

```
def Uniques(llista):
    aux=set()
    for e in llista:
        aux.add(e)

    out = []
    for e in aux:
        out.append(e)
    return (out)
```

To create an empty set

Equivalent code:

```
def Uniques(llista):
    conjunt = set(llista)
    u_llista = list(conjunt)
    return (u_llista)
```