Useful functions while working with arrays:

- Push an element to the end of a: `push!(a,b)` (as a single element even if it is an Array. Equivalent to python `append`)
- To append all the elements of b to a: `append!(a,b)` (if b is a scalar obviously push! and append! are the same. Attention that a string is treated as a list!. Equivalent to Python `extend` or `+=`)
- Concatenation of arrays (new array): `a = [1,2,3]; b = [4,5]; c = vcat(1,a,b)`
- Remove an element from the end: `pop!(a)`
- Removing an element at the beginning (left): `popfirst!(a)`
- Remove an element at an arbitrary position: `deleteat!(a, pos)`
- Add an element (b) at the beginning (left): `pushfirst!(a,b)` (no, `appendfirst!` doesn't exists!)
- Sorting: `sort!(a)` or `sort(a)` (depending on whether we want to modify or not the original array)
- Reversing an arry: `a[end:-1:1]`
- Checking for existence: `in(1, a)`
- Getting the length: `length(a)`
- Get the maximum value: `maximum(a)` or `max(a...)` (`max` returns the maximum value between the given arguments)
- Get the minimum value: `minimum(a)` or `min(a...)` (`min` returns the maximum value between the given arguments)
- Empty an array: `empty!(a)` (only column vector, not row vector)
- Transform row vectors in column vectors: `b = vec(a)`
- Random-shuffling the elements: `shuffle(a)` (or `shuffle!(a)`. From Julia 1.0 this require `using Random` before)
- Checking if an array is empty: `isempty(a)`
- Find the index of a value in an array: `findall(x -> x == value, myarray)`. This is a bit tricky. The first argument is an anonymous function that returns a boolean value for each value of `myarray`, and then `find()` returns the index position(s).
- Delete a given item from a list:

```
deleteat!(myarray, findall(x -> x == myunwanteditem, myarray))
```

Reference: https://syl1.gitbook.io/julia-language-a-concise-tutorial/language-core/getting-started