Basic Usage of Slices

Let’s create a basic list:

```
>>> nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]

```

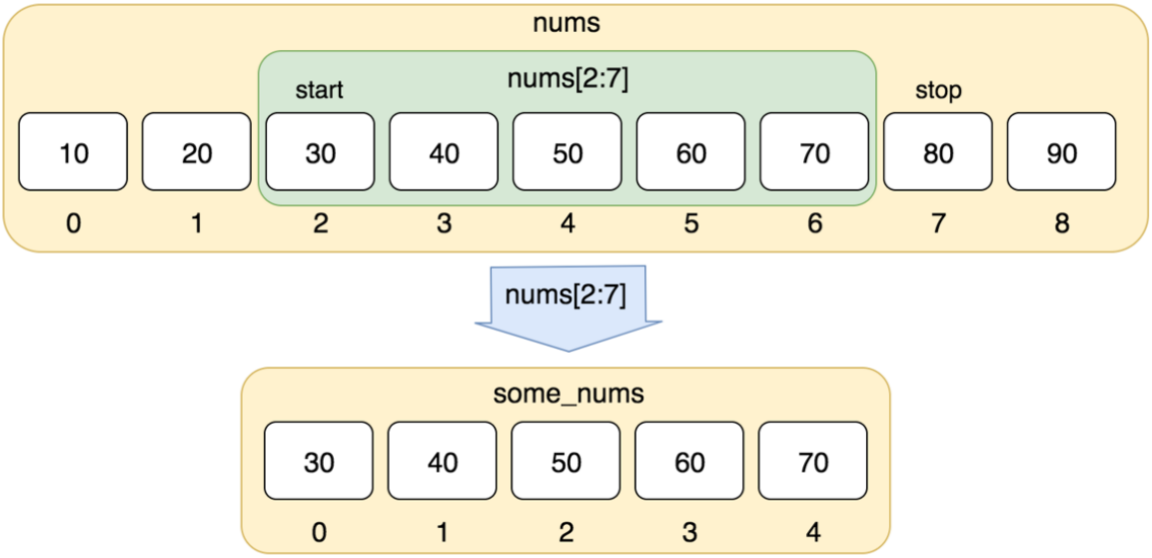
What if we want to take a sublist from the `nums` list? This is a snap when using slice:

```
>>> nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> some_nums = nums[2:7]
>>> some_nums
[30, 40, 50, 60, 70]

```

So, here is our first example of a slice: *2:7*. The full slice syntax is: *start:stop:step*. `start` refers to the index of the element which is used as a start of our slice. `stop` refers to the index of the element we should stop just before to finish our slice. `step` allows you to take each nth-element within a *start:stop* range.

In our example `start` equals `2`, so our slice starts from value `30`. `stop` is `7`, so the last element of the slice is `70` with index `6`. In the end, slice creates a new list(we named it `some_nums`) with selected elements.



We did not use `step` in our slice, so we didn't skip any element and obtained all values within the range.

With slices we can extract an arbitrary part of a list, e.g.:

```
>>> nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> nums[0:4]
[10, 20, 30, 40]
```

Here we start from the first element(index `0`) and take a list till the element with index `4`.

Taking n last elements of a list

Negative indexes allow us to easily take n-last elements of a list:

```
>>> nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> nums[-3:]
[70, 80, 90]
```

Here, the `stop` parameter is skipped. That means you take from the `start` position, till the end of the list. We start from the third element from the end (value `70` with index `-3`) and take everything to the end.

Taking every nth-element of a list

What if we want to have only every 2-nd element of `nums`? This is where the `step` parameter comes into play:

```
>>> nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> nums[::2]
[10, 30, 50, 70, 90]
```

Here we omit `start`/`stop` parameters and use only `step`. By providing `start` we can skip some elements:

```
>>> nums[1::2]
[20, 40, 60, 80]
```

And if we don't want to include some elements at the end, we can also add the `stop` parameter:

We can freely mix negative and positive indexes in `start` and `stop` positions:

```
>>> nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> nums[1:-1]
[20, 30, 40, 50, 60, 70, 80]
>>> nums[-3:8]
[70, 80]
>>> nums[-5:-1]
[50, 60, 70, 80]
```