# String Slicing

In this lesson, we'll understand what slicing is and how it can be applied to strings.

**We'll cover the following**



## Definition #

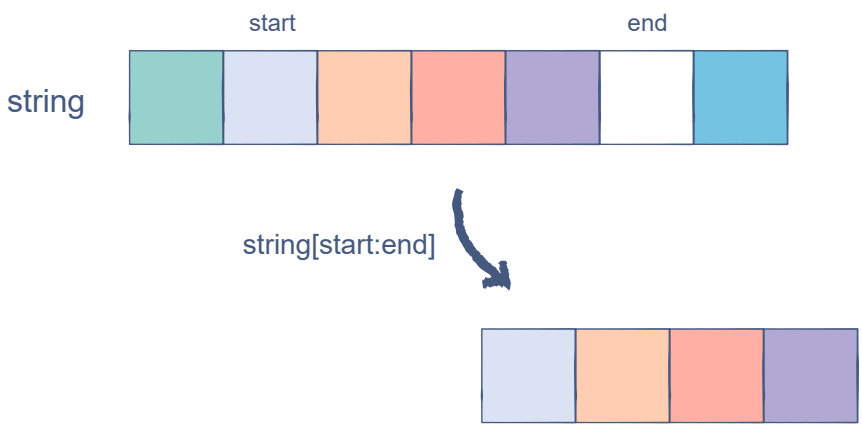
> **Slicing** is the process of obtaining a portion (substring) of a string by using its indices.

Given a string, we can use the following template to slice it and obtain a substring:

```
string[start:end]
```

- `start` is the index from where we want the substring to start.
- `end` is the index where we want our substring to end.

The character at the `end` index in the string, will not be included in the substring obtained through this method.



A substring!

Let's look at a few examples:

```
my_string = "This is MY string!"
print(my_string[0:4]) # From the start till before the 4th index
print(my_string[1:7])
print(my_string[8:len(my_string)]) # From the 8th index till the end
```

## Slicing with a Step #

Until now, we've used slicing to obtain a contiguous piece of a string, i.e., all the characters from the starting index to before the ending index are retrieved.

However, we can define a **step** through which we can skip characters in the string. The default step is `1`, so we iterate through the string one character at a time.

The step is defined after the `end` index:

```
string[start:end:step]
```

Let's see how this works:

```
my_string = "This is MY string!"
print(my_string[0:7])  # A step of 1
print(my_string[0:7:2])  # A step of 2
print(my_string[0:7:5])  # A step of 5
```

## Reverse Slicing #

Strings can also be sliced to return a reversed substring. In this case, we would need to switch the order of the `start` and `end` indices.

A negative step must also be provided:

```
my_string = "This is MY string!"
print(my_string[13:2:-1]) # Take 1 step back each time
print(my_string[17:0:-2]) # Take 2 steps back. The opposite of what happens in the slide above
```

## Partial Slicing #

One thing to note is that specifying the `start` and `end` indices is **optional**.

If `start` is not provided, the substring will have all the characters until the `end` index.

If `end` is not provided, the substring will begin from the `start` index and go all the way to the end.

Let's see this in action:

```
my_string = "This is MY string!"
print(my_string[:8])  # All the characters before 'M'
print(my_string[8:])  # Al the characters starting from 'M'
print(my_string[:])  # The whole string
print(my_string[::-1])  # The whole string in reverse (step is -1)
```

That’s pretty much all we need to know about string slicing. Play around with the strings above to get a better understanding of how slicing works.

---

In the next lesson, we’ll understand the purpose of operators in Python.