# Slicing (List,Tuple And String)

---

***Slicing in Python*** is a feature that enables accessing parts of the sequence. In slicing a string, we create a substring, which is essentially a string that exists within another string. We use slicing when we require a part of the string and not the complete string.

## Slicing in String:

### Syntax :

> *string[start : end : step]*
>
> - *start : We provide the starting index.*
> - *end : We provide the end index(this is not included in substring).*
> - *step : It is an optional argument that determines the increment between each index for slicing.*

```
1  str ="Geeks for Geeks !"
2  print(str[: 3])
3  print(str[1 : 5 : 2])
4  print(str[-1 : -12 : -2])
```

```
Gee
ek
!seGrf
```

In Python, a string is a sequence of characters enclosed within single quotes (`' '`), double quotes (`" "`), or triple quotes (`''' '''` or `""" """`). Strings are used to represent text, and they can include letters, numbers, symbols, and whitespace.

Strings are immutable in Python, meaning once a string is created, its contents cannot be changed.

| Method | Description | Syntax | Example Usage | Example Output |
| --- | --- | --- | --- | --- |
| `str.find(sub[, start[, end]])` | Searches for the first occurrence of the substring `sub` within the optional `start` and `end` range, and returns the lowest index of the substring. Returns `-1` if not found. | `'hello'.find('l')` | `'hello'.find('l')` | `2` |
| `str.rfind(sub[, start[, end]])` | Searches for the last occurrence of the substring `sub` within the optional `start` and `end` range, and returns the highest index of the substring. Returns `-1` if not found. | `'hello world'.rfind('l')` | `'hello world'.rfind('l')` | `9` |

| | | | | |
|---|---|---|---|---|
| `str.index(sub[, start[, end]])` | Searches for the first occurrence of the substring `sub` within the optional `start` and `end` range, and returns the lowest index of the substring. Raises `ValueError` if not found. | `'hello'.index('l')` | `'hello'.index('l')` | `2` |

| | | | |
|---|---|---|---|
| `str.isalnum()` | Checks if all characters in the string are alphanumeric. | `'abc123'.isalnum()` | `True` |
| `str.isalpha()` | Checks if all characters in the string are alphabetic. | `'abc'.isalpha()` | `True` |
| `str.isdigit()` | Checks if all characters in the string are digits. | `'123'.isdigit()` | `True` |
| `str.islower()` | Checks if all characters in the string are lowercase. | `'hello'.islower()` | `True` |
| `str.isspace()` | Checks if all characters in the string are whitespace. | `' '.isspace()` | `True` |
| `str.istitle()` | Checks if the string is title-cased. | `'Hello World'.istitle()` | `True` |

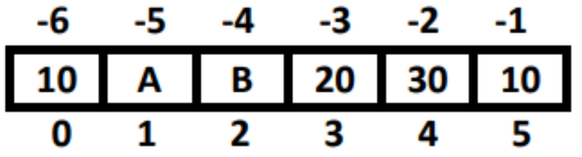
# List Data Structure

**If we want to represent a group of individual objects as a single entity where insertion order preserved and duplicates are allowed, then we should go for List.**

**insertion order preserved.**
**duplicate objects are allowed**
**heterogeneous objects are allowed.**
**List is dynamic because based on our requirement we can increase the size and decrease the size.**
**In List the elements will be placed within square brackets and with comma seperator.**

**We can differentiate duplicate elements by using index and we can preserve insertion order by using index. Hence index will play very important role.**
**Python supports both positive and negative indexes. +ve index means from left to right where as negative index means right to left**

**[10,"A","B",20, 30, 10]**




**List objects are mutable.i.e we can change the content.**

| Method/Function | Description | Example | Output |
| --- | --- | --- | --- |
| append(x) | Adds an item `x` to the end of the list. | `lst = [1, 2, 3]`<br>`lst.append(4)` | `[1, 2, 3, 4]` |
| extend(iterable) | Extends the list by appending elements from the iterable (e.g., list, tuple). | `lst = [1, 2, 3]`<br>`lst.extend([4, 5])` | `[1, 2, 3, 4, 5]` |
| insert(i, x) | Inserts an item `x` at a given position `i`. | `lst = [1, 2, 3]`<br>`lst.insert(1, 5)` | `[1, 5, 2, 3]` |
| remove(x) | Removes the first occurrence of an item `x` from the list. | `lst = [1, 2, 3, 2]`<br>`lst.remove(2)` | `[1, 3, 2]` |
| pop([i]) | Removes and returns the item at the given position `i`. If no index is specified, it removes and returns the last item. | `lst = [1, 2, 3]`<br>`lst.pop()`<br>`lst.pop(0)` | `3`<br>`[2, 3]` |
| clear() | Removes all items from the list, making it empty. | `lst = [1, 2, 3]`<br>`lst.clear()` | `[]` |
| index(x[, start[, end]]) | Returns the index of the first occurrence of item `x`. Searches within a given range if s ↓ ied. | `lst = [1, 2, 3, 2]`<br>`lst.index(2)` | `1` |

| | | | |
|---|---|---|---|
| count(x) | Returns the number of times an item `x` appears in the list. | `lst = [1, 2, 3, 2]`<br>`lst.count(2)` | `2` |
| sort(key=None, reverse=False) | Sorts the list in ascending order by default. Can be customized using the `key` parameter and `reverse` flag. | `lst = [3, 1, 2]`<br>`lst.sort()`<br>`lst.sort(reverse=True)` | `[1, 2, 3]`<br>`[3, 2, 1]` |
| reverse() | Reverses the elements of the list in place. | `lst = [1, 2, 3]`<br>`lst.reverse()` | `[3, 2, 1]` |
| copy() | Returns a shallow copy of the list. | `lst = [1, 2, 3]`<br>`lst_copy = lst.copy()` | `lst_copy = [1, 2, 3]` |
| max(iterable) | Returns the largest item in the list or the largest of two or more arguments. | `lst = [1, 2, 3]`<br>`max(lst)` | `3` |
| min(iterable) | Returns the smallest item in the list or the smallest of two or more arguments. | `lst = [1, 2, 3]`<br>`min(lst)` | `1` |
| sum(iterable, start=0) | Returns the sum of all items in the list, optionally starting with the `start` value. | `lst = [1, 2, 3]`<br>`sum(lst)` | `6` |
| sorted(iterable, key=None, reverse=False) | Returns a new sorted list from the items in the iterable. Th ginal list is not modified. | `lst = [3, 1, 2]`<br>`sorted(lst)`<br>`sorted(lst,` | `[1, 2, 3]`<br>`[3, 2, 1]` |