----: Indexing in Python :---

Index is a stored position of an object from ordered collections like string,list,tuple.

Positive Index	**0**	**1**	**2**	**3**	**4**
	H	**E**	**L**	**L**	**O**
Negative Index	**-5**	**-4**	**-3**	**-2**	**-1**

For example, if we have a string "HELLO", we can access the first letter "H" using its index 0 by using the square bracket notation: string[0]

```
                                         |----> positive indexing
                                         |
 <-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|------|------>
      -5      -4      -3      -2      -1       0       1       2       3       4       5
                                         |
       Negative indexing<------
```

=> -ve indexing start with -1	=> +ve index start with 0
=> -ve indexing read from R to L	=> +ve index read from L to R
=> -ve indexing write from L to R	=> +ve indexing write from L to R
=> +ve indexing stop point in (stop+1)	=> +ve indexing stop point in (stop-1)

Python's built-in index() function is a useful tool for finding the index of a specific element in a sequence. This function takes an argument representing the value to search for and returns the index of the first occurrence of that value in the sequence.

If the value is not found in the sequence, the function raises a ValueError. For example, if we have a list [1, 2, 3, 4, 5], we can find the index of the value 3 by calling list.index(3), which will return the value 2 (since 3 is the third element in the list, and indexing starts at 0).

Python Index Examples

The method index() returns the lowest index in the list where the element searched for appears. If any element which is not present is searched, it returns a **ValueError**.

Example:--

```
list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
element = 3
print(list.index(element))

O/P:-
2
```

Example:--(Throws a ValueError)

```
list = [4, 5, 6, 7, 8, 9, 10]
element = 3 # Not in the list
print(list.index(element))

O/P:-
Traceback (most recent call last):
  File "e:\DataSciencePythonBatch\index.py", line 7, in <module>
    print(list.index(element))
ValueError: 3 is not in list
```

Example:--(Index of a string element)

```
list = [1, 'two', 3, 4, 5, 6, 7, 8, 9, 10]
element = 'two'
print(list.index(element))

O/P:-
1
```

What does it mean to return the lowest index?

```
list = [3, 1, 2, 3, 3, 4, 5, 6, 3, 7, 8, 9, 10]
element = 3
print(list.index(element))
O/P:-
0
```

Find element with particular start and end point:--

Syntax:-

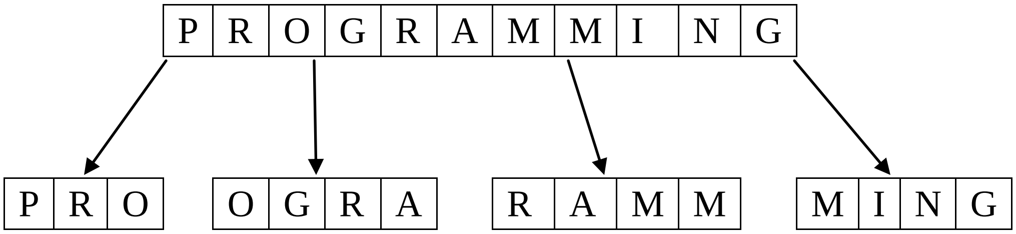
collection.index(element, start, stop)
collection.index(element)
collection.index(element, start)

Example:- index() provides you an option to give it hints to where the value searched for might lie.

```
list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
element = 7
print(list.index(element, 5, 8))

O/P:-
6
```

-----: Slicing in Python:-----



Slicing is the extraction of a part of a string, list, or tuple. It enables users to access the specific range of elements by mentioning their indices.

Syntax: Object [start : stop : step/direction]
Object [start : stop]

- **start:** The start parameter in the slice function is used to set the starting position or index of the slicing. The default value of the start is 0.
- **stop:** The stop parameter in the slice function is used to set the end position or index of the slicing[(n-1) for positive value and (n+1) for negative value].
- **step:** The step parameter in the slice function is used to set the number of steps to jump. The default value of the step is 1.

Rules for working :---

Step1:- Need to check step direction by default it’s goes to positive direction.

Setp2:- Need to check start-point and end-point direction. If not given it follows step derection

Step3:- If both directions are matched, then working fine and we are getting O/P

Step4:- Otherwise it gives empty subsequence.

-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
I		L	O	V	E		P	Y	T	H	O	N
0	1	2	3	4	5	6	7	8	9	10	11	12

Ex:-1

```
var = "I love python"
print(var[::])

O/P:-
I love python
```

Ex:2

```
var = "I love python"
print(var[::-1])

O/P:-
nohtyp evol I
```

Ex:-3

```
var = "I love python"
print(var[-2:-5:])

O/P:-
```

Ex:-4

```
var = "I love python"
print(var[2:5:-1])

O/P:-
```

Ex:-5

```
var = "I love python"
print(var[::2])

O/P:-
Ilv yhn
```

Ex:-6

```
var = "I love python"
print(var[::-2])
O/P:-
nhy vlI
```

-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
W	E	L	C	O	M	E		T	O		M	Y		B	L	O	G
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Ex:-7,8,9,10,11,12

```
var = "WELCOME TO MY BLOG"
print(var[3:18]) O/P:- COME TO MY BLOG

print(var[2:14:2]) O/P:- LOET Y

print(var[:7]) O/P:- WELCOME

print(var[8:-1:1]) O/P:- TO MY BLO

print(var[-6:-9:-3]) O/P:- Y

print(var[-9:-9:-1]) O/P:-
```

Range()

Range() function is used to generate collection in python.

Syntax:

range(start,stop/end,step/direction)

Note :-

1. **for** +ve direction collection step must be +ve.
2. **for** -ve direction collection step must be –ve.
3. for +ve direction collection stop/end point must be (required+1).
4. for -ve direction collection stop/end point must be (required-1).
5. Start point is always what we require.

```
my_range = range(1,11)
print(list(my_range))

my_range = range(1,11,-1)
print(list(my_range))

my_range = range(-1,-11,-1)
print(list(my_range))

my_range = range(-1,-11,1)
print(list(my_range))

my_range = range(11)
print(list(my_range))



O/P:--
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[]
[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]
[]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
my_range = range(2,11,2)
print(list(my_range))

my_range = range(1,10,2)
print(list(my_range))

my_range = range(-2,-11,-2)
print(list(my_range))

my_range = range(-1,-10,-2)
print(list(my_range))

[2, 4, 6, 8, 10]
[1, 3, 5, 7, 9]
[-2, -4, -6, -8, -10]
[-1, -3, -5, -7, -9]
```

```
my_range = range(5,2,-1)
print(list(my_range))

my_range = range(-5,-2,1)
print(list(my_range))

my_range = range(5,6,1)
print(list(my_range))

my_range = range(-5,-6,-1)
print(list(my_range))

O/P:--
[5, 4, 3]

[-5, -4, -3]

[5]

[-5]
```