SLICING : (Important)

In Python, slicing is a technique used to extract a portion (subsequence) of a sequence such as a string, list, or tuple. Slicing allows you to create a new sequence containing a subset of elements from the original sequence, based on specified start and end indices.

The basic syntax for slicing is as follows:

sequence[start:end:step]

- **sequence**: This refers to the original sequence from which you want to extract a portion.
- **start**: The index of the sequence where the slicing starts (inclusive). If not provided, it defaults to 0 (the beginning of the sequence).
- **end**: The index of the sequence where the slicing ends (exclusive). If not provided, it goes until the end of the sequence.
- **step**: This optional parameter specifies the increment between indices while slicing. If not provided, it defaults to 1.

EXAMPLES :

```
# Slicing a string
text = "Hello, World!"
print(text[7:])
# Output: "World!"
print(text[:5])
 # Output: "Hello"
print(text[7:12])
# Output: "World"
print(text[::2])
# Output: "Hlo ol!"

# Slicing a list
numbers = [1, 2, 3, 4, 5]
print(numbers[2:])
```

```
 # Output: [3, 4, 5]
print(numbers[:3])
 # Output: [1, 2, 3]
print(numbers[1:4])
 # Output: [2, 3, 4]
print(numbers[::2])
# Output: [1, 3, 5]

# Slicing a tuple
fruits = ("apple", "banana", "cherry", "date")
print(fruits[1:])
 # Output: ("banana", "cherry", "date")
print(fruits[:3])
# Output: ("apple", "banana", "cherry")
print(fruits[::2])
# Output: ("apple", "cherry")
```

EXAMPLE-

```
  +---+---+---+---+---+---+
              | P | y | t | h | o | n |
              +---+---+---+---+---+---+
Slice position: 0   1   2   3   4   5   6
Index position:   0   1   2   3   4   5

>>> p = ['P','y','t','h','o','n']
# Why the two sets of numbers:
# indexing gives items, not lists
>>> p[0]
 'P'
>>> p[5]
 'n'

# Slicing gives lists
>>> p[0:1]
 ['P']
>>> p[0:2]
 ['P','y']
```

```
>>> p[5] # the last of six items, indexed from zero
 'n'
>>> p[0:5] # does NOT include the last item!
 ['P','y','t','h','o']
>>> p[0:6] # not p[0:5]!!!
 ['P','y','t','h','o','n']

>>> p[0:4] # Start at the beginning and count out 4 items
 ['P','y','t','h']
>>> p[1:4] # Take one item off the front
 ['y','t','h']
>>> p[2:4] # Take two items off the front
 ['t','h']

>>> p[2:3]
 ['t']
>>> p[2:3] = ['T']
>>> p
 ['P','y','T','h','o','n']

>>> p[2:4]
 ['T','h']
>>> p[2:4] = ['t','r']
>>> p
 ['P','y','t','r','o','n']

>>> p = ['P','y','t','h','o','n'] # Start over
>>> p[2:4] = ['s','p','a','m']
>>> p
 ['P','y','s','p','a','m','o','n']

>>> p = ['P','y','t','h','o','n']
>>> p[0:4]
 ['P','y','t','h']
>>> p[1:4]
 ['y','t','h']
>>> p[2:4]
 ['t','h']
>>> p[3:4]
 ['h']
>>> p[4:4]
 []


>>> p = ['P','y','t','h','o','n']
>>> p[2:4] = ['x','y'] # Assigned list is same length as slice
>>> p
 ['P','y','x','y','o','n'] # Result is same length
```

```
>>> p = ['P','y','t','h','o','n']
>>> p[3:4] = ['x','y'] # Assigned list is longer than slice
>>> p
 ['P','y','t','x','y','o','n'] # The result is longer
>>> p = ['P','y','t','h','o','n']
>>> p[4:4] = ['x','y']
>>> p
 ['P','y','t','h','x','y','o','n'] # The result is longer still

>>> p = ['P','y','t','h','o','n']
>>> p[0:4]
 ['P','y','t','h']
>>> p[1:4]
 ['y','t','h']
>>> p[2:4]
 ['t','h']
>>> p[3:4]
 ['h']
>>> p[4:4]
 []
>>> p[5:4]
 []
>>> p[6:4]
 []


>>> p[5:3:-1]
 ['n','o']

>>> p[4:4]
 []
>>> p[5:4]
 []
>>> p[6:4]
 []
```

Now,

```
+---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
   0   1   2   3   4   5
  -6  -5  -4  -3  -2  -1
```

```
>>> x = [1,2,3,4,5,6]
>>> x[::-1]
[6,5,4,3,2,1]

>>> x = [1,2,3,4,5,6]
>>> x[::-2]
```

```
[6,4,2]
```

- **Slicing in Python**
- [a:b:c]
- len = length of string, tuple or list
- c -- default is +1. The sign of c indicates forward or backward, absolute value of c indicates steps. Default is forward with step size 1. Positive means forward, negative means backward.
- a -- When c is positive or blank, default is 0. When c is negative, default is -1.
- b -- When c is positive or blank, default is len. When c is negative, default is -(len+1).
- Understanding index assignment is very important.
- In forward direction, starts at 0 and ends at len-1
- In backward direction, starts at -1 and ends at -len

EXAMPLE :

```
0   1   2  3   4  5  6  7  8  9  10  11
           a   s   t  r   i  n  g
   -9  -8  -7   -6   -5  -4   -3  -2  -1
```

```
>>> l1[:-3:-1] # a default is -1
[4, 3]

>>> l1[::] # c default is +1, so a default is 0, b default is len
[2, 3, 4]

>>> l1[::-1] # c is -1 , so a default is -1 and b default is -(len+1)
[4, 3, 2]


>>> l1[-100:-200:-1] # Interesting
[]

>>> l1[-1:-200:-1] # Interesting
[4, 3, 2]


>>> l1[-1:-1:1]
[]
```

```
>>> l1[-1:5:1] # Interesting
[4]


>>> l1[1:-7:1]
[]

>>> l1[1:-7:-1] # Interesting
[3, 2]

>>> l1[:-2:-2] # a default is -1, stop(b) at -2 , step(c) by 2 in reverse direction
[4]
```

Index:

```
     ------------>
  0   1   2   3   4
+---+---+---+---+---+
| a | b | c | d | e |
+---+---+---+---+---+
  0  -4  -3  -2  -1
     <------------
```

Slice:

```
   <---------------|
|--------------->
:   1   2   3   4   :
+---+---+---+---+---+
| a | b | c | d | e |
+---+---+---+---+---+
:  -4  -3  -2  -1   :
|--------------->
   <---------------|
```

- **Understanding the difference between indexing and slicing:**

Wiki Python has this amazing picture which clearly distinguishes indexing and slicing.

```
Index from rear:    -6  -5  -4  -3  -2  -1
Index from front:    0   1   2   3   4   5
                   +---+---+---+---+---+---+
                   | a | b | c | d | e | f |
                   +---+---+---+---+---+---+
Slice from front:  :   1   2   3   4   5   :
Slice from rear:   :  -5  -4  -3  -2  -1   :
```

It is a list with six elements in it. To understand slicing better, consider that list as a set of six boxes placed together. Each box has an alphabet in it.

Indexing is like dealing with the contents of box. You can check contents of any box. But you can't check the contents of multiple boxes at once. You can even replace the contents of the box. But you can't place two balls in one box or replace two balls at a time.

```
In [122]: alpha = ['a', 'b', 'c', 'd', 'e', 'f']

In [123]: alpha
Out[123]: ['a', 'b', 'c', 'd', 'e', 'f']

In [124]: alpha[0]
Out[124]: 'a'

In [127]: alpha[0] = 'A'

In [128]: alpha
Out[128]: ['A', 'b', 'c', 'd', 'e', 'f']
```

Slicing is like dealing with boxes themselves. You can pick up the first box and place it on another table. To pick up the box, all you need to know is the position of beginning and ending of the box.

You can even pick up the first three boxes or the last two boxes or all boxes between 1 and 4. So, you can pick any set of boxes if you know the beginning and ending. These positions are called start and stop positions.

The interesting thing is that you can replace multiple boxes at once. Also you can place multiple boxes wherever you like.

```
In [130]: alpha[0:1]
Out[130]: ['A']

In [131]: alpha[0:1] = 'a'

In [132]: alpha
```

```
Out[132]: ['a', 'b', 'c', 'd', 'e', 'f']

In [133]: alpha[0:2] = ['A', 'B']

In [134]: alpha
Out[134]: ['A', 'B', 'c', 'd', 'e', 'f']

In [135]: alpha[2:2] = ['x', 'xx']

In [136]: alpha
Out[136]: ['A', 'B', 'x', 'xx', 'c', 'd', 'e', 'f']
```

- **Slicing With Step:**

Till now you have picked boxes continuously. But sometimes you need to pick up discretely. For example, you can pick up every second box. You can even pick up every third box from the end. This value is called step size. This represents the gap between your successive pickups. The step size should be positive if You are picking boxes from the beginning to end and vice versa.

```
In [137]: alpha = ['a', 'b', 'c', 'd', 'e', 'f']

In [142]: alpha[1:5:2]
Out[142]: ['b', 'd']

In [143]: alpha[-1:-5:-2]
Out[143]: ['f', 'd']

In [144]: alpha[1:5:-2]
Out[144]: []

In [145]: alpha[-1:-5:2]
Out[145]: []
```

EXAMPLE :

```
+---+---+---+---+---+---+---+---+---+
 | C | O | M | P | U | T | E | R | S |
 +---+---+---+---+---+---+---+---+---+
 0   1   2   3   4   5   6   7   8   9
-9  -8  -7  -6  -5  -4  -3  -2  -1
```

```
COMPUTERS[ 4 : 7 ]     =  UTE
COMPUTERS[ 2 : 5 : 2 ] =  MU
COMPUTERS[-5 : 1 :-1 ] =  UPM
COMPUTERS[ 4 ]         =  U
COMPUTERS[-4 :-6 :-1 ] =  TU
COMPUTERS[ 2 :-3 : 1 ] =  MPUT
```

```
COMPUTERS[ 2 :-3 :-1 ] =
COMPUTERS[   :   :-1 ] =  SRETUPMOC
COMPUTERS[-5 :   ]     =  UTERS
COMPUTERS[-5 : 0 :-1 ] =  UPMO
COMPUTERS[-5 :   :-1 ] =  UPMOC
COMPUTERS[-1 : 1 :-2 ] =  SEUM
```

EXAMPLE :

str="Name string"
Slicing example: [start:end:step]

```
str[start:end] # Items start through end-1
str[start:]    # Items start through the rest of the array
str[:end]      # Items from the beginning through end-1
str[:]         # A copy of the whole array
```

Below is the example usage:

```
print str[0] = N
print str[0:2] = Na
print str[0:7] = Name st
print str[0:7:2] = Nm t
print str[0:-1:2] = Nm ti
```