Basic Concepts of Py

① Indexing :- Insertion order is maintained by passing indexes

+ve indexing :- Always starts from first element & with value 0

-ve indexing :- ——"—— last element & ——"—— -1

s = "Python"

first elt

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

Last elt

	Low	High
+ve	0	5
-ve	-6	-1

② Hashing :- Accessing elts of sequence from their indexes

Syntax:

seq_name [valid index no] ⇒ o/p will elt at a given index

seq name → s = "Python"

0 1 2 3 4 5

-6 -5 -4 -3 -2 -1

```
print(s[0]) # 'P'   ← first elt
print(s[1]) # 'y'
print(s[5]) # 'n'
print(s[7]) # IndexError
```

```
print(s[-1]) # 'n'  # Last elt
print(s[-2]) # 'o'
print(s[-6]) # 'P'
print(s[-8]) # IndexError
```

③ Slicing :- Cutting off the seq into partition/slices

Syntax :-

seq_name [Low: start index : High: end index (value-1)]

s = "Python"

0 1 2 3 4 5

-6 -5 -4 -3 -2 -1

(a) seq[Low: start : High: end] (value-1)

```
print(s[0:5]) # Pytho
print(s[0:15]) # Python
print(s[1:4]) # yth
print(s[3:11]) # hon
print(s[8:15]) # '' + empty res
print(s[6:0]) # '' + empty res
print(s[2:3]) # t
```

(b) seq[start : -1]

```
print(s[0:-1]) # Python
print(s[2:-1]) # thon
print(s[4:-1]) # on
# Last 'n' elts
print(s[-3:-1]) # hon
print(s[-5:-1]) # ytho
print(s[-4:-1]) # on
```

(c) seq[0 : end] (value-1)

```
# First 'n' elts
print(s[:6]) # Python
print(s[:5]) # Pytho
print(s[:3]) # Pyt
print(s[:1]) # P
——"——(s[:4]) # Pyth
```

(d) seq[0 : -1]

```
print(s[:-1]) # Python
```

---

④ Striding / Stepping :-

(I) +ve stride

(II) -ve stride

(I) When stride is +ve

Syntax :- seq_name [Low: start index : High: end index (value-1) : +stride]

(a) seq[Low: start : High: end : +stride] (value-1)

```
print(s[0:5:2]) # Pto
print(s[1:6:2]) # yhn
print(s[0:6:3]) # Ph
print(s[1:6:3]) # yo
print(s[5:0:2]) # ''
print(s[8:11:2]) # ''
```

(b) seq[start : -1 : +stride]

```
print(s[0::2]) # Pto
print(s[0::3]) # Ph
——"——(s[0::4]) # Po
——"——(s[4::2]) # to
```

(c) seq[0 : end : +stride] (value-1)

```
print(s[:5:2]) # Pto
print(s[:6:3]) # Ph
print(s[:6:6]) # P
```

(d) seq[0 : -1 : +stride]

```
print(s[::2]) # Pto
——"——(s[::3]) # Ph
——"——(s[::5]) # Pn
```

(II) When stride is -ve the ordering is changed a bit since we're counting down

Syntax :-

seq_name [High: start index : Low: end index : -stride] (value+1)

(a) seq[High : Low : -stride] ⇒ [seq[High], seq[High-stride], seq[High-stride-stride], ---- seq[Low+1]]

```
print(s[5:0:-1]) # 'nohty'
print(s[5:0:-2]) # 'nhy'
print(s[5:1:-1]) # ''
print(s[0:5:-1]) # ''
print(s[5:0:-3]) # 'nt'
```

(b) seq[High : 0 : -stride] ⇒ [seq[High], seq[High-stride], ---- seq[0]]

```
print(s[5::-1]) # 'nohtyP'
print(s[4::-2]) # 'otP'
print(s[-1::-1]) # 'nohtyP'
```

⇒ It reverses any seq

(c) seq[: Low : -stride] ⇒ [seq[-1], seq[-1-stride], ---- seq[Low+1]]

```
print(s[:0:-1]) # 'nohty'
print(s[:0:-2]) # 'nhy'
print(s[:0:-3]) # 'nt'
```

(d) seq[: : -stride] ⇒ [seq[-1], seq[-1-stride], ----, seq[0]]

```
print(s[::-1]) # 'nohtyP'
print(s[-1::-1]) # 'nohtyP'
```

⇒ It reverses any seq

Select+Alt+3 =>Comment

Select+Alt+4 =>UnComment