* Supports real-coorld problem slaving.

29-09-25

SLICING:-

* It means extracting a part (subsequence) of a sequence like a list, tuple or string

⇒ It albuls you to access a range of elments easily

Syn:- var [start position : stop value : step size]

* Default Value is 0
* Stop position is end Of Sequence
* step size is 1

Eg:

```
      0  1  2  3  4
l1 = [10,20,30,40,50]
```

1. basic slicing:

print (l1 [1:4])

// [20, 30, 40] ⇒ starts from index 1 and end befo index (4)

2. Omitting start & stop:

print (list [:]) // entire list [10,20,30,40,50]

print (l1 [2:]) // from in 2 toend [30, 40, 50]

print (l1 [:3]) // from start to index (2)

[10, 20, 30]

③. Using step size:

```
print(l1[::2])  # every 2nd element
                [10, 30, 50]
print(l1[::-1])  # reverse order
                [50, 40, 30, 20, 10]
```

program to find the max value in list

```
l1 = [4,6,1,9,2]
max_val = l2[0]
for i in l2:
    if(i > max_val):
        max_val = i
print(max_val)   # 9
```

To print the value in ascending order

```
l2 = [4,6,1,9,2]
l2.sort(reverse=False)
print(l2)
```

o/p [1,2,4,6,9]

TUPLE:-

* It is represented by ()
* It is heterogeneous in nature
* It is ordered, immutable, allow the duplicates

Eg:

To print position and value

```
t1 = ('task', 'pen', 'paper')
for a in t1:
    print(a + ":")
    for pos, val in enumerate(a):
        print("pos=", pos, "value=", val)
```

o/p

task:

pos = 0, val = t
pos = 1, val = a
pos = 2, val = s
pos = 3, val = k

Pen:

pos = 0, val = p
pos = 1, val = e
pos = 2, val = n

paper:

pos = 0, val = p
pos = 1, val = a
pos = 2, val = p
pos = 3, val = e
pos = 4, val = r