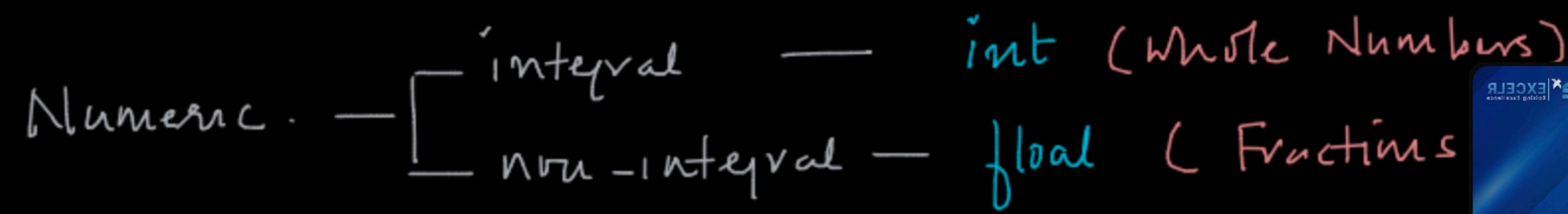


You are screen sharing

## Datatypes



Boolean → bool

True / False

 $x = 10$  $\boxed{10}$  $x \rightarrow \text{int.}$  $\boxed{\quad}$  $23 \rightarrow M_1 \rightarrow S_3$  $M_{24} \rightarrow$  $\Pi_L \rightarrow S_5$ 

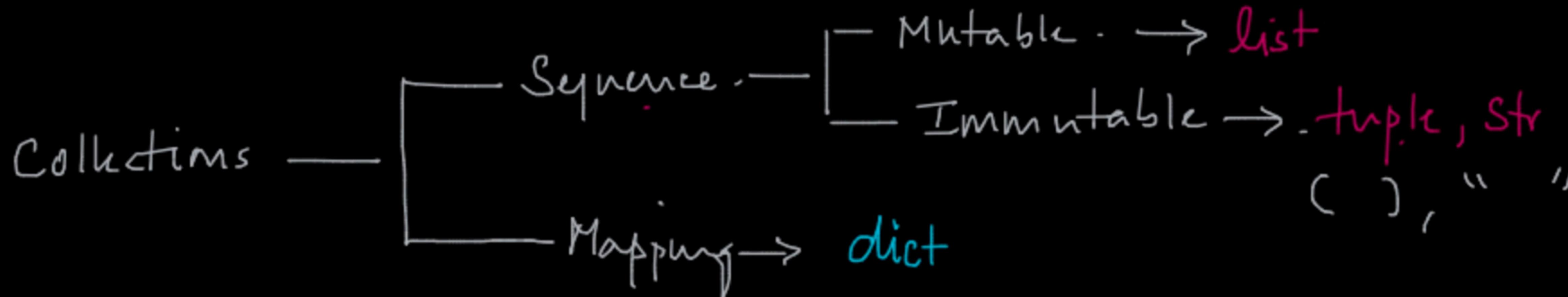
;

 $\vdots$ 

;

 $M_{23} \rightarrow 65$ 

;



Common properties

-7	-6	-5	-4	-3	-2	-1	elements
35	46	75	96	58	85	89	
0	1	2	3	4	5	6	→ index.

## (1) Positional Indexing-

$lst = [35, 46, \underline{75}, \underline{96}, 58, 85, 89]$  ✓  
 $\text{type}(lst) \rightarrow \text{list}$

$str1 = 'Hello'$   
 $\text{type}(str1) \rightarrow \text{string}$

$[H | e | l | l | o]$  → sequence of 'characters'

$lst2 = [10, 5.172, \text{True}, 'Apple', [20, 30, 40], (1, 2, 3)]$  → Homogeneous  
 $\text{type}(lst2) \rightarrow \text{list}$

$s2 = "Python 3.0"$   
 $\text{type}(s2) \rightarrow \text{string}$

Access Values - → Heterogeneous

$t1 = (25, 43, 74, 60)$

$\text{type}(t1) \rightarrow \text{tuple}$        $str1[1] \rightarrow 'e'$

$lst[3] \rightarrow 96$   
 $\hookrightarrow \text{extractor}$

$t1[-1] \rightarrow \text{last elements}$

$s2[-1] \rightarrow \text{last elements}$

$t1[1] \rightarrow 1$

$lst[-4] \rightarrow 96$



Geethika

You are screen sharing



Sequences are iterable

Heraclite → take me elsewhere

at a time

Shop-list = [ 'Apples', 'Rice', 'Soap',  
↓ ↓ ↓ ↓  
'Shampoo', 'Bhiller'

```
for mark in marklist:  
    → 5 time
```

→ if mark > 80

→ If mewk be and so  
First

→ - If new by and b to  
second.

```
→ for elem in shopping-list:
```

for elem in shopping-list:  
    → print(elem) → Apple

Plant ('Hu')

→ Coal  
→ Apple  
Rice -  
Soap

long  
strmer

if  $\text{MWS}[0] \geq g_0 \rightarrow \text{dist}$

if marks[0] between 60 and 80 → First -

11- marks [v] between 40 & 60 → semi

11- marks [d] between 40 & 60 → semi

```
if [new K[1]] > - -  
{  
    - - - - -  
    - - - - -  
    - - - - -
```

IF  $\lceil \text{new\_k}[2] \rceil > 80$  → first



rule

$$\rightarrow \boxed{l_2 = l_1} \quad \underline{\text{Assign}}$$

↓

$$[1, 2, 3]$$

$$l_1 \rightarrow \begin{bmatrix} 1 \\ 2 \\ 3 \\ \text{uno} \end{bmatrix}$$



$l_2.append(1\text{no})$  ✓

$l_2 \rightarrow [1, 2, 3, 1\text{no}]$

$l_1 \rightarrow [1, 2, 3, 1\text{no}]$

$l_2 = l_1.append()$  ↪  
2nd

$$l_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$l_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1\text{no} \end{bmatrix}$$