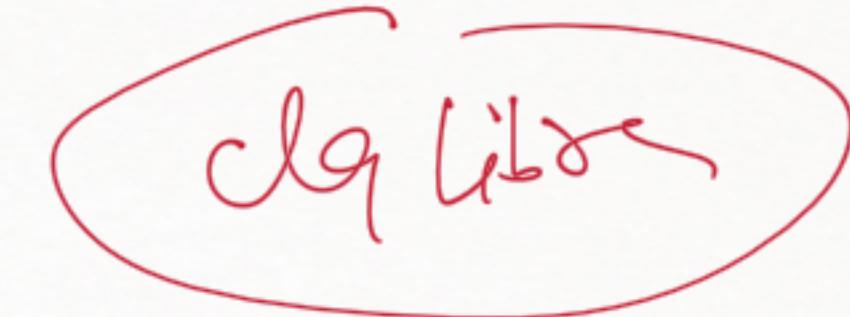


You



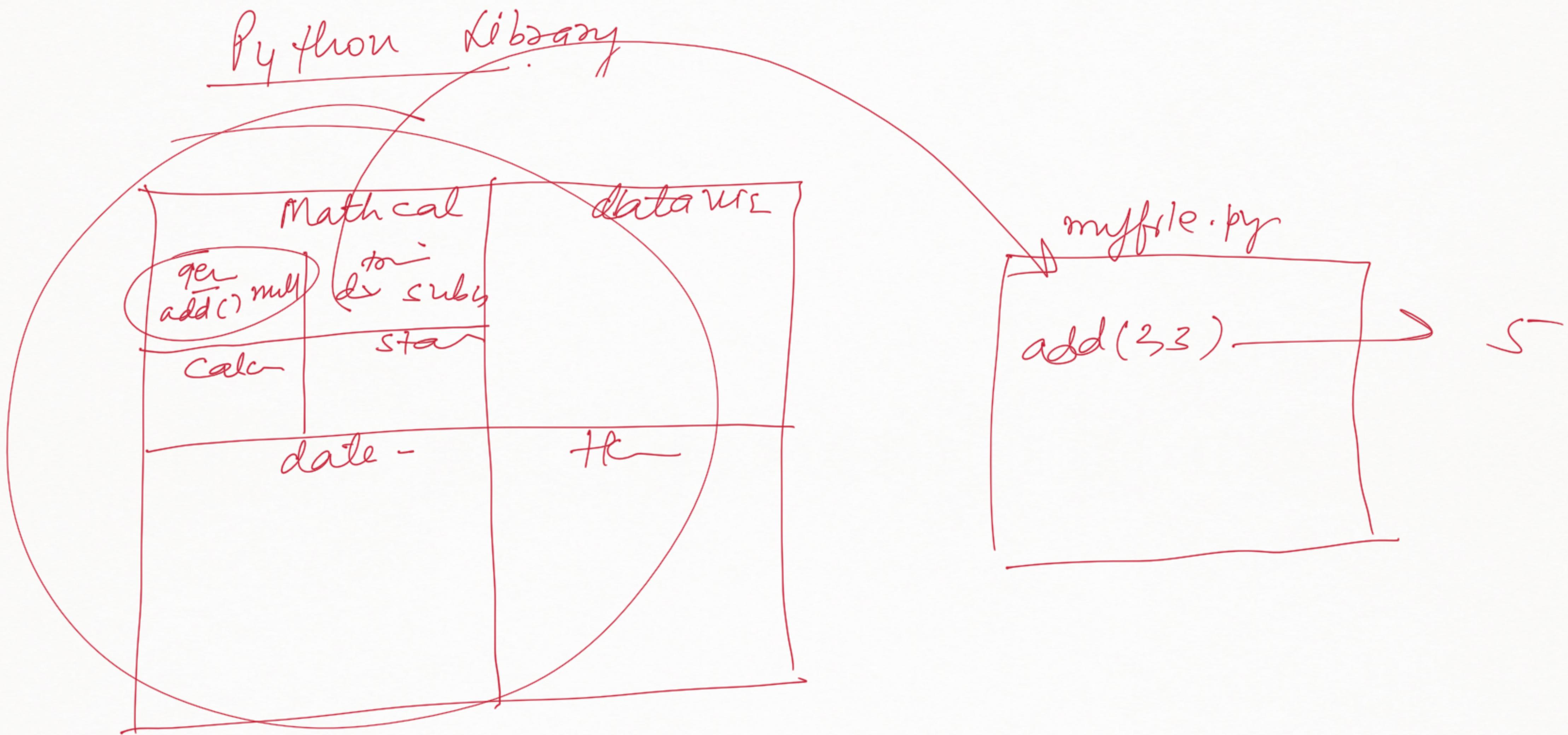
→ Extra cost

→ No extra cost

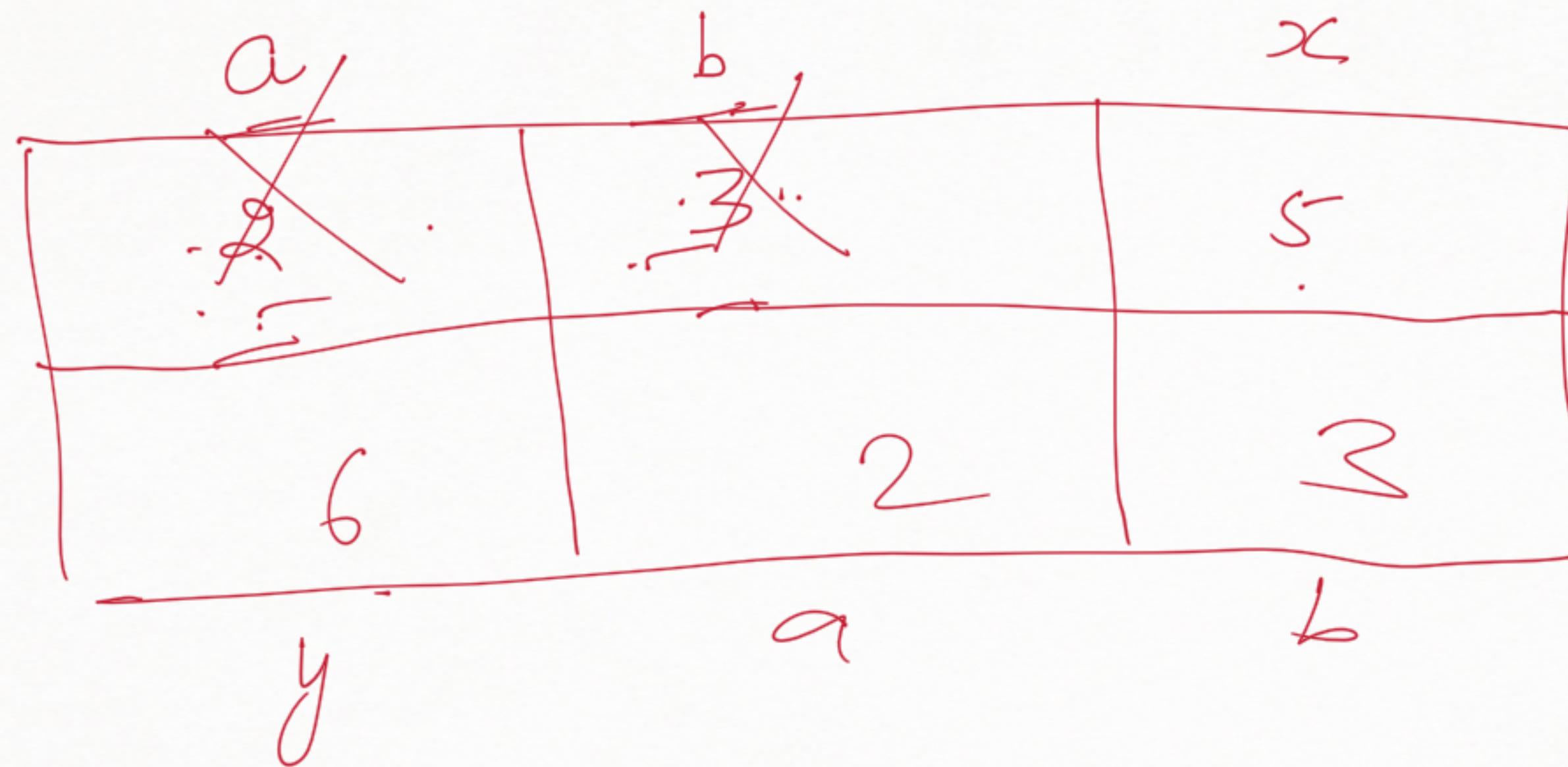
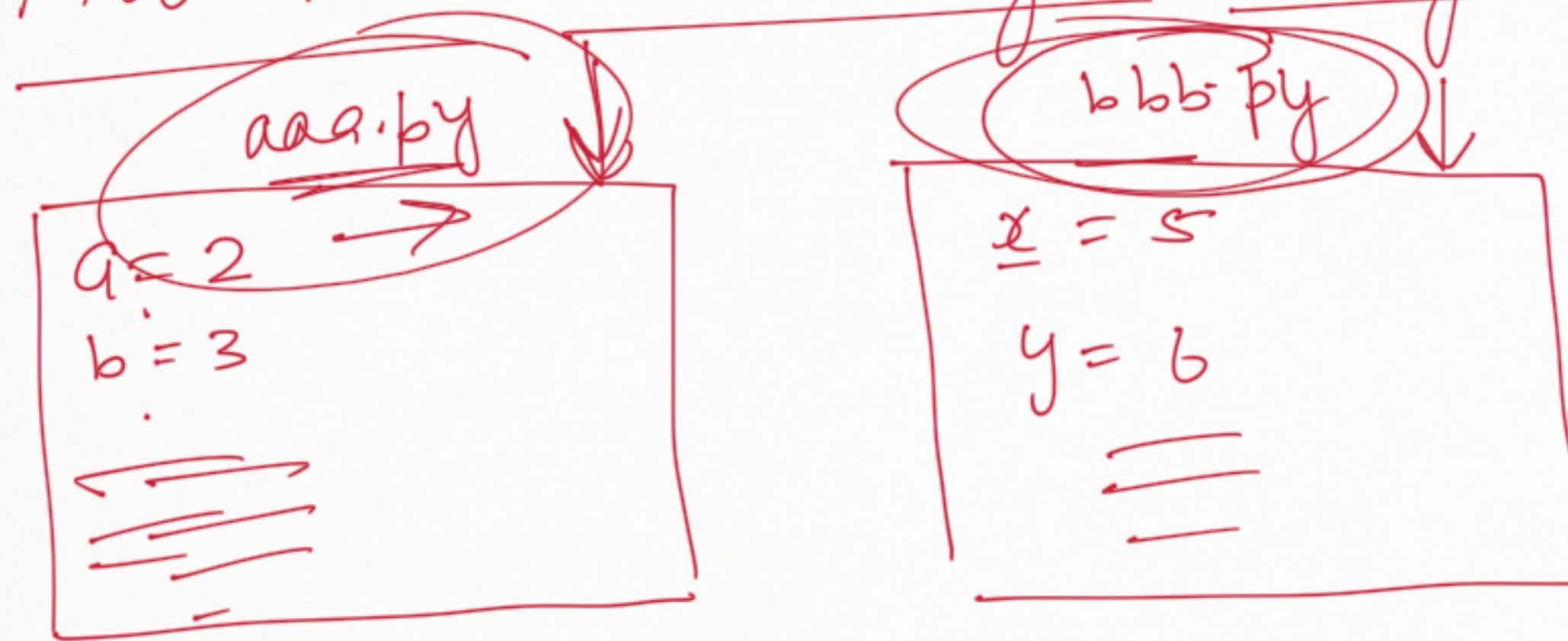


→ Extra time

→ No extra time



Automatic memory management & Garbage collector:



aaa.py ↴

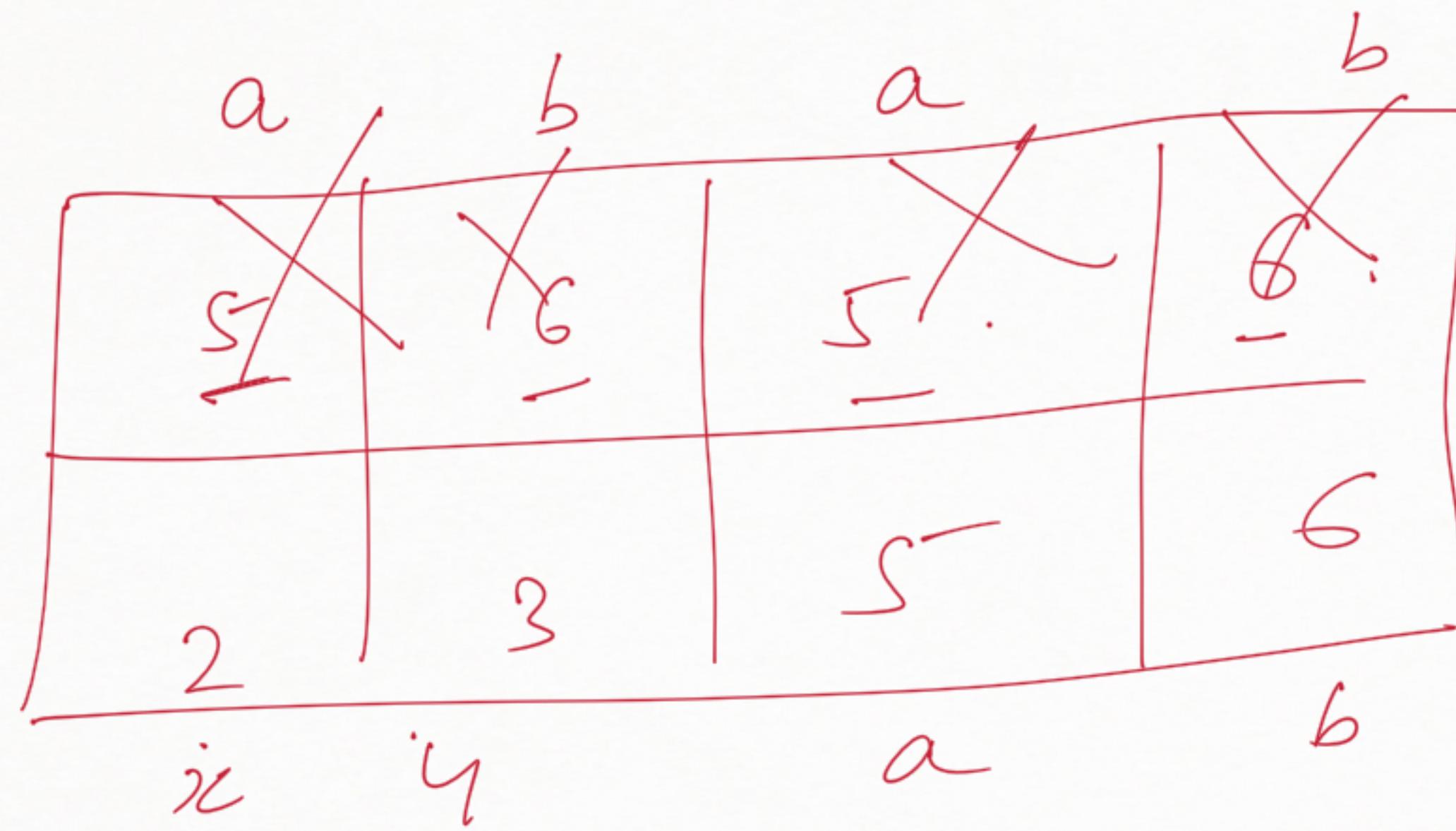
$$a = 5$$

$$b = 6$$

bbb.py ↴

$$x = 2$$

$$y = 3$$

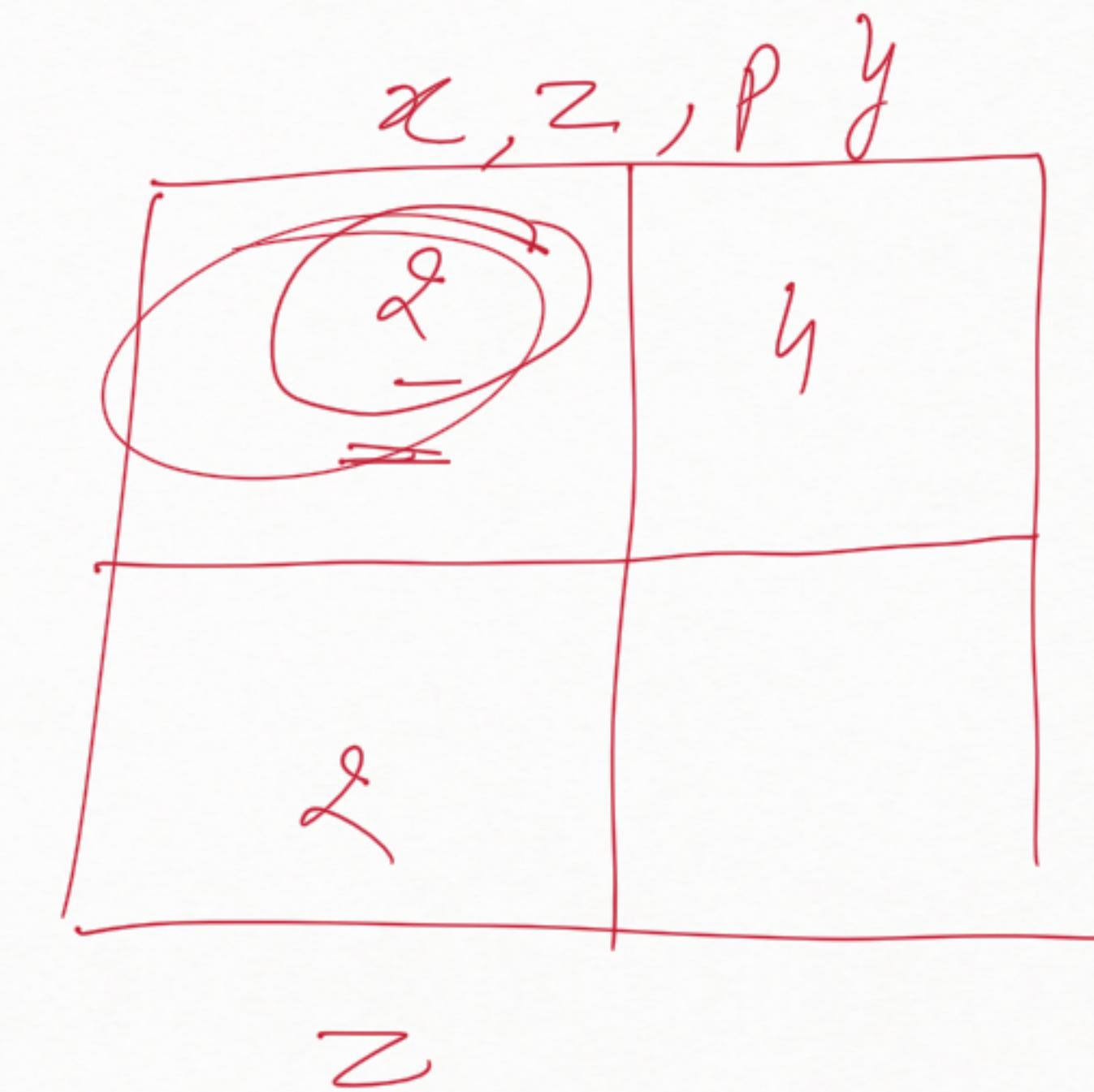


$$x = 2$$

$$y = 4$$

$$z = 2$$

$$p = 2$$



$\rightarrow a = \underline{2}$

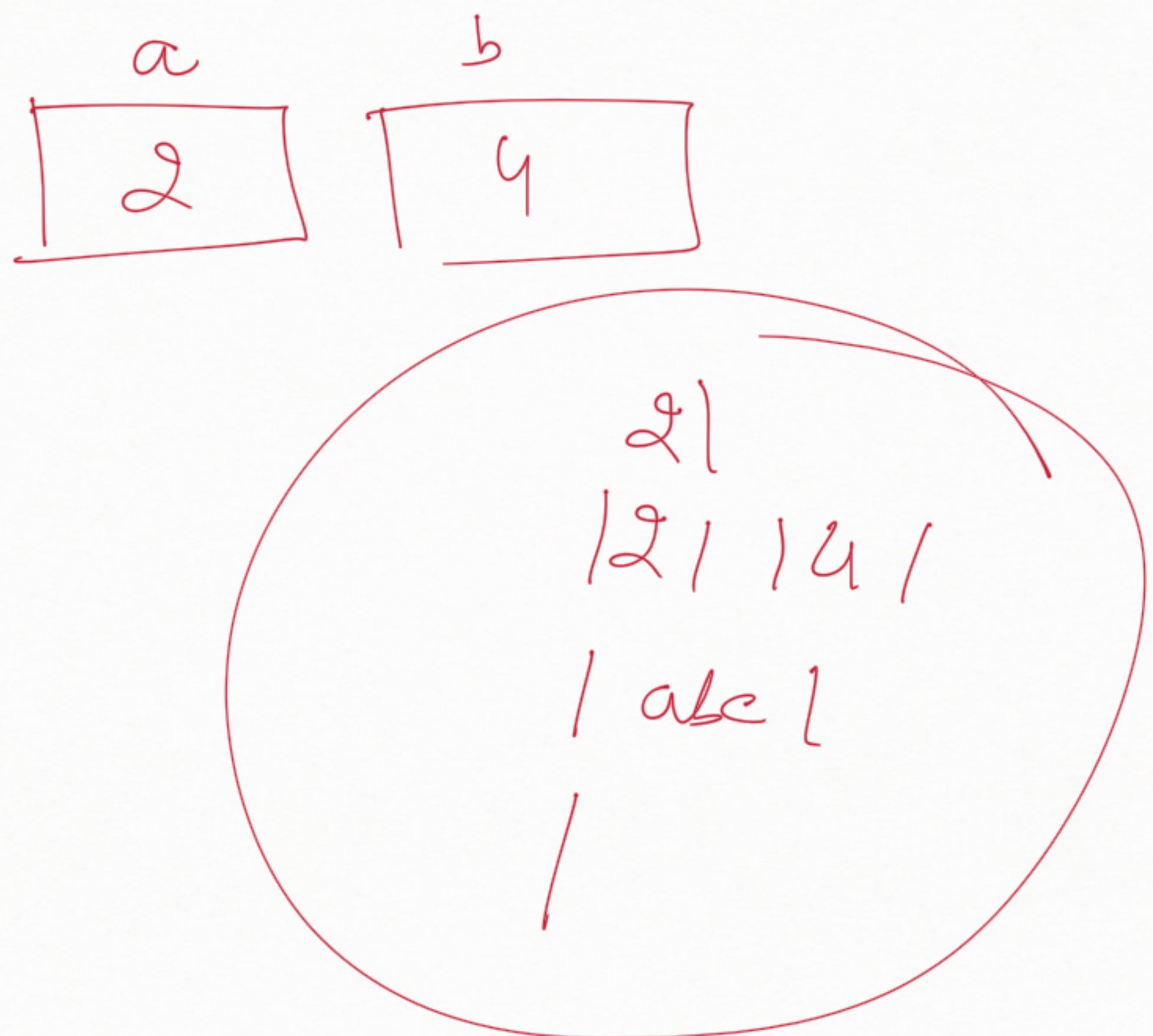
$\rightarrow b = \underline{4}$

$\rightarrow \text{print}(\underline{a})$

$\rightarrow \text{print}(a, \underline{b})$

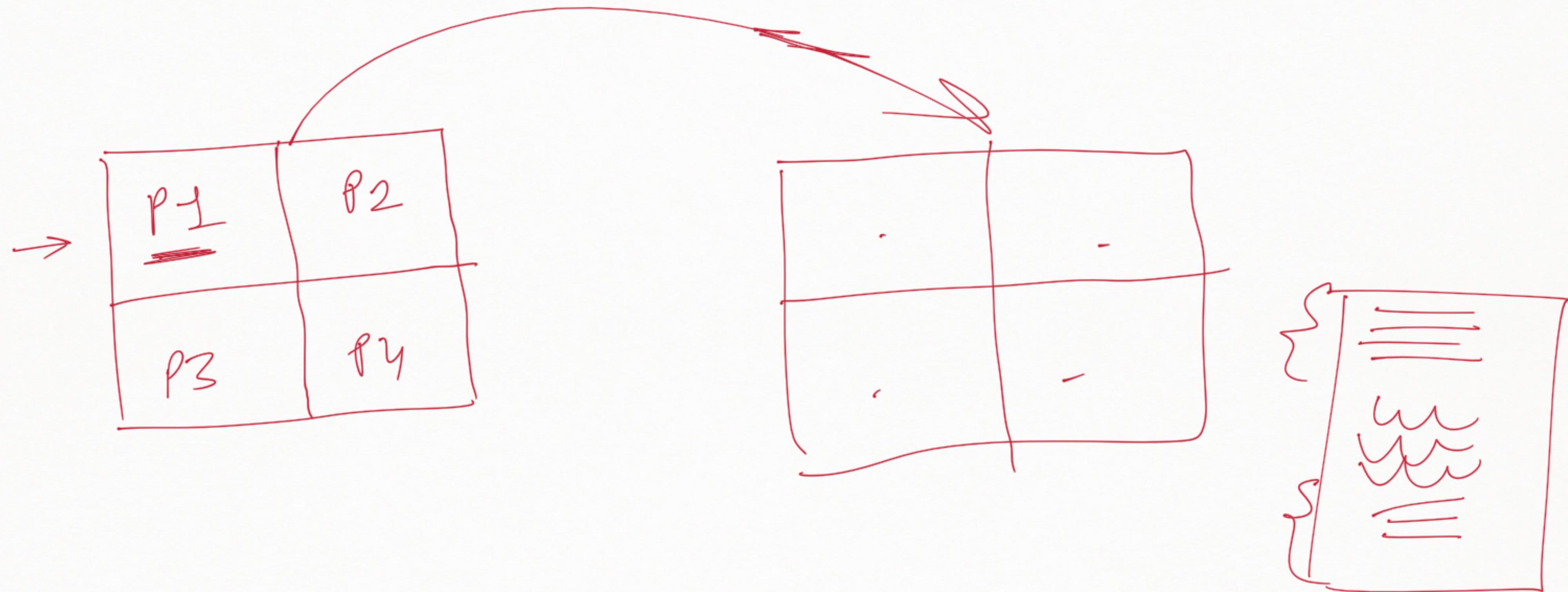
$\rightarrow \text{print}('abc' \underline{c})$

\rightarrow

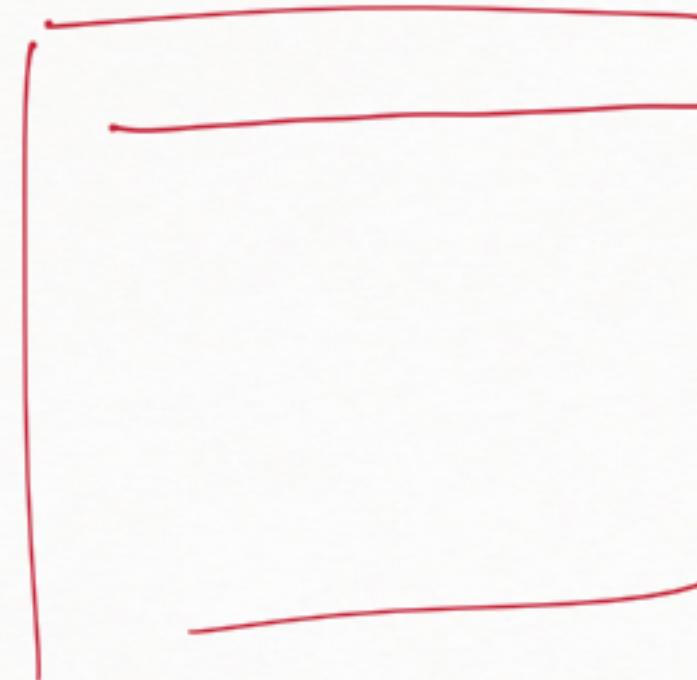


Scf = ''' end = ''\n''

Comments in Python:-

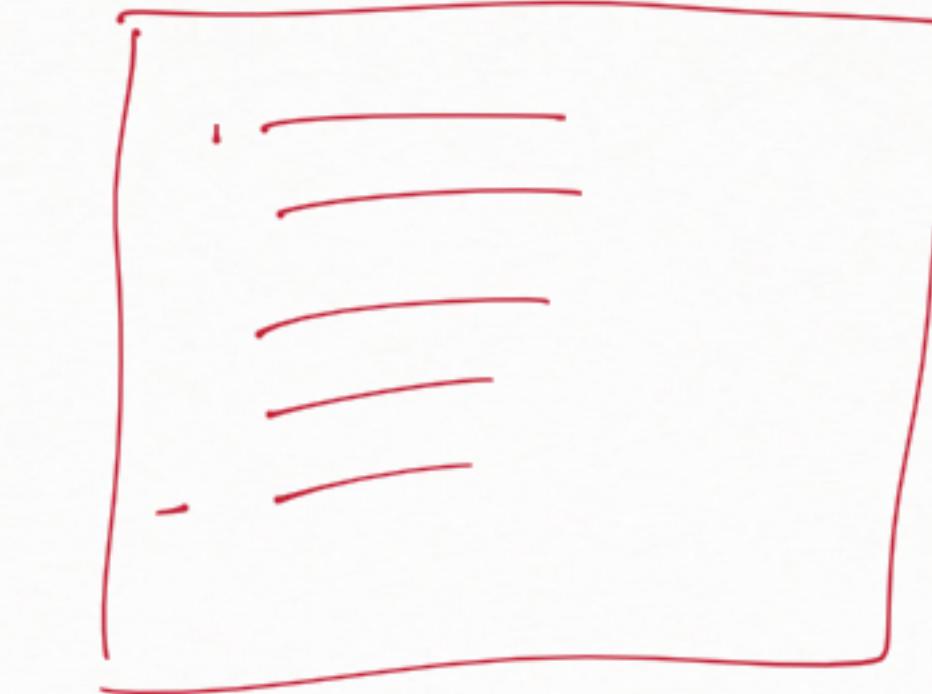


#



Single line

Comment



Multi line comment

. " " " = " "

" " "



" " "

Data type in Python:

Number

int \rightarrow -n to +n

float \rightarrow -n.n to +n.n

complex \rightarrow $2+9j$
 \rightarrow
real imag

$j \rightarrow \sqrt{-1}$.

String :

'2'
"2"
'''2'''
""2""

collection of charact -

'abc' → strg

'a' → str

| | | |
|----|----|----|
| -3 | -2 | -1 |
| a | b | c |
| 0 | 1 | 2 |

$x : 'abc'$

$x \leftarrow$

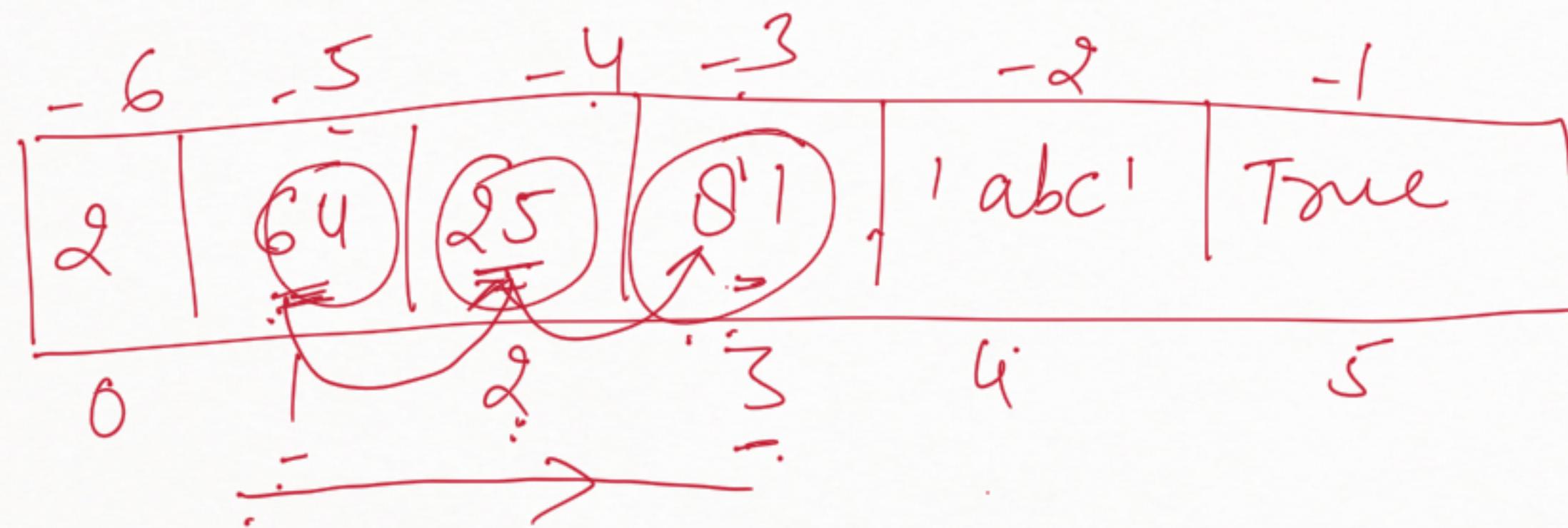
$x[0]$ or $x[-3]$

→ 'a'

$x[-2]$ or $x[1] \rightarrow b,$

$x[-1]$ or $x[2] \rightarrow 'c'$

$a = [2, 64, 25, 81, 'abc', \text{True}]$



$\ln[\underline{\text{start}} : \underline{\text{stop}} : \underline{\text{step}}]$

$a[-5 : -2 : 1]$

$[64, 25, 81]$

$$2 - 1 = (1,)$$
$$3 - 2 = (1,)$$

$$-4 + 5 = 1$$

$a[1 : 4 :]$

$a[-5 : 4 : 1]$

$a[1 : -2 : 1]$

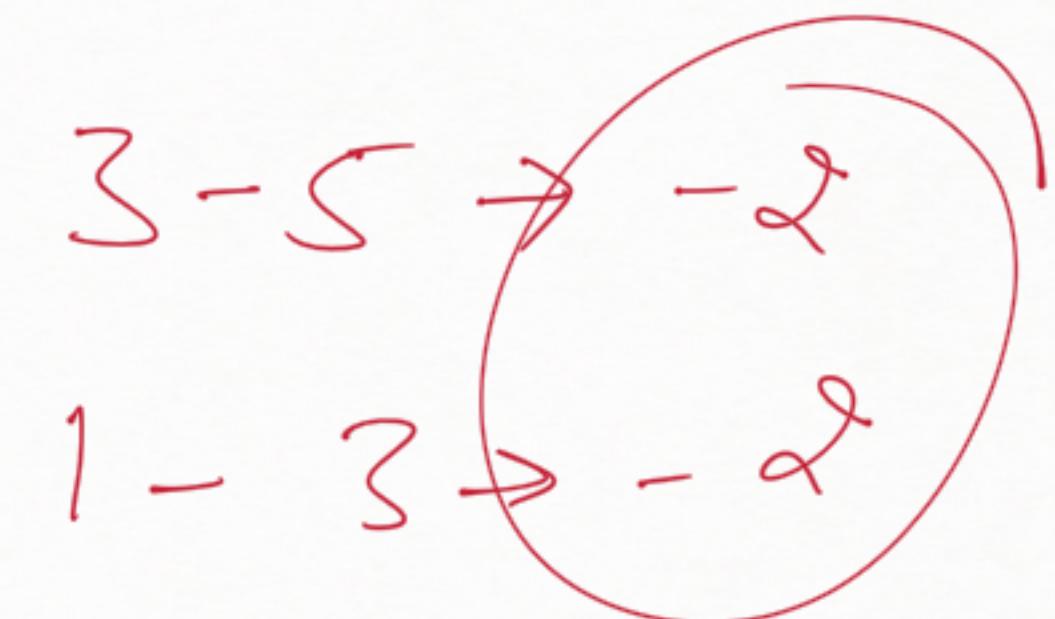
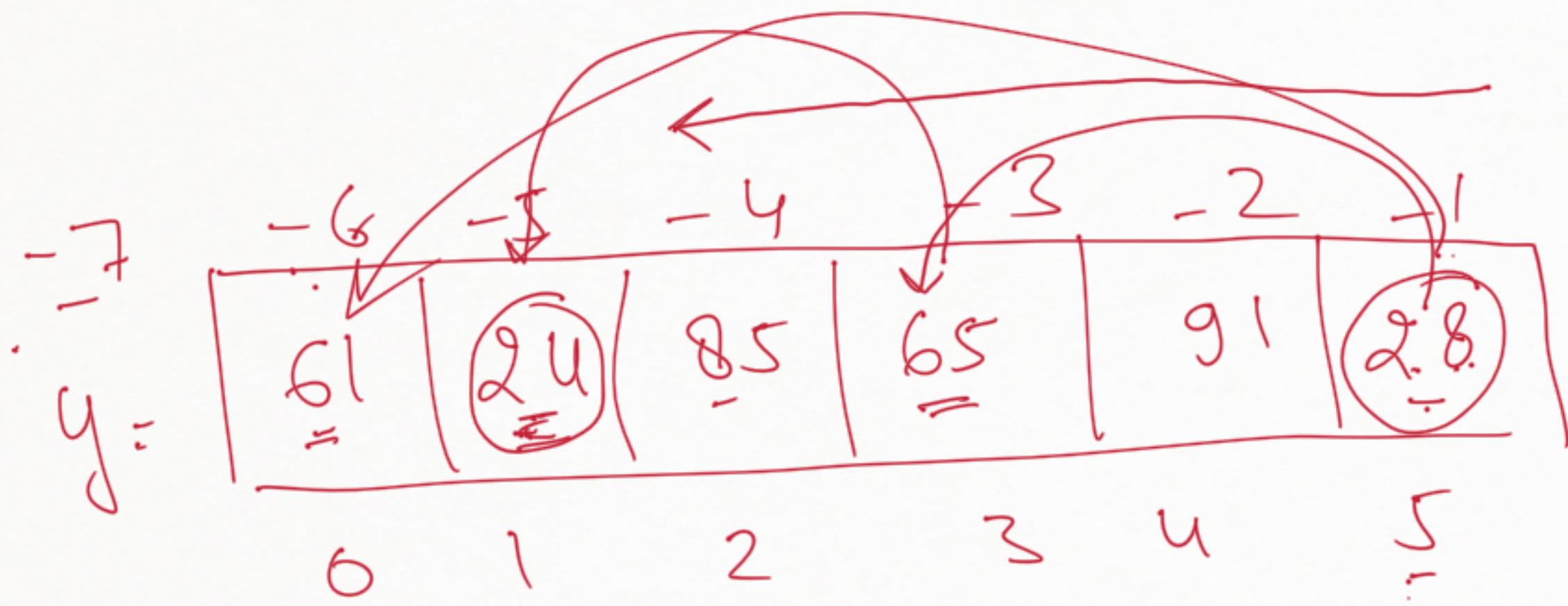
| -5 | 4 | 2 | -2 | -1 |
|--------|------|------|------|------|
| 24 | 81 | 61 | 92 | 85 |
| \div | | | | 4 |

$$4 - 0 = \textcircled{4}$$

$[24, 85]$

$x[0:5:4]$

$x[-5:5:4]$



$$[28, 65, 24]$$

$$y[5:-6:-2]$$

$$y[-1:0:-2]$$

$$[28, 61]$$

$$y[5:-7:-5]$$

$$y[-1:-7:-5]$$

$$0 - 5 \rightarrow -5$$

Start

M:N:L

M

M

M

M

0

0

0

0

Stop

N-1

N-1

len(a) - 1

"

N-1

N-1

len(a)-1

"

Step

L

+1

+1

L

L

+1

L

+1

$$a = \boxed{2 \downarrow 8 \downarrow 45 \downarrow 61 \downarrow 29} \quad \begin{matrix} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 2 & 3 & 4 \end{matrix}$$

a[1:5:2] $\rightarrow [8, 61]$

a[2:5] \rightarrow

list-name [start : stop : step]

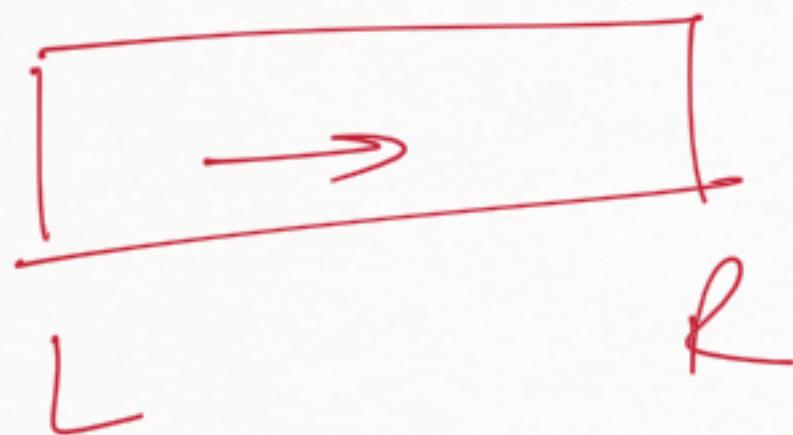
a =

| | | | | | |
|----|----|----|----|----|----|
| -6 | -5 | -4 | -3 | -2 | -1 |
| 2 | 8 | 9 | 27 | 67 | 25 |
| 0 | 1 | 2 | 3 | 4 | 5 |

[2, 8, 9]

a[0:3:1]

Slicing $L \rightarrow R$



Start $\rightarrow 0$

Stop $\rightarrow -1$

Step $\rightarrow +1$

Slicing $R \rightarrow L$



start $\rightarrow -1$

stop $\rightarrow 0$

step $\rightarrow +1$

$$a = \left[\begin{array}{cccccc|c} 2 & -6 & -5 & -4 & -3 & -2 & -1 \\ 2 & | & 9 & 61 & 28 & 49 & 62 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \right] \rightarrow$$

$$a[1:6:3] \rightarrow [9, 49]$$

$$a[1:6] \rightarrow [9, 61, 28, 49, 62] \\ \rightarrow$$

$$a[1:] \rightarrow [9, 61, 28, 49, 62, 40]$$

If step → $\frac{+}{-}$
 L to R

$\frac{-}{+}$
 R to L

| $a =$ | $\begin{array}{c} -7-6 \\ \hline 6 81 -6 49 82 83 45 \\ \hline 10 1. 2 3 4 5 6 \end{array}$ |
|--------------|---|
| $a[::]$ | $[4, 61, 26, 49, 82, 83, 45]$ |
| $a[6:-2:-1]$ | $[45, 83, 82, 49, 26, 61, 4]$ |
| $a[5:1]$ | $[]$ |
| $a[5:]$ | $[83, 45]$ |

$a[::] \rightarrow [4, 61, 26, 49, 82, 83, 45]$
 $a[6:-2:-1] \rightarrow [45, 83, 82, 49, 26, 61, 4]$
 $a[2:89:-2] \rightarrow [26, 82, 45]$
 $a[:: -24:-2] \rightarrow [45, 82, 26, 4]$

Adding elements

append() → add one element at last of the list

insert() → add one element at your desired location

extend() → add more than one element at last of the list
,

Removing elements

pop() → remove one element by using index from list

remove() → remove one element ————— value from list

del → remove one or more than one

clear() → remove all —————

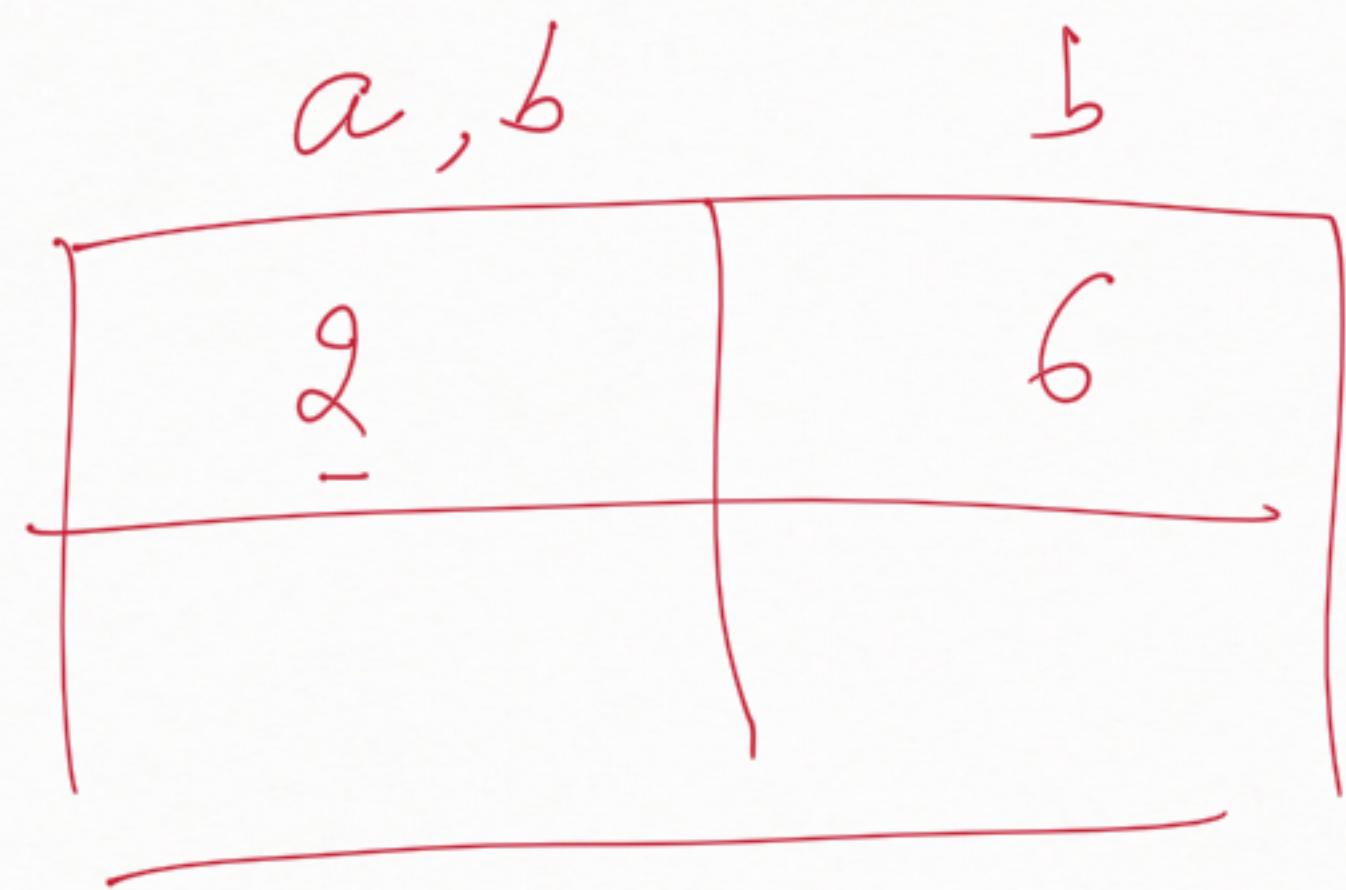
other methods

- {
 - index() → accoring index value
 - count() → occur of any el
 - reverse() → give the element
 - sort() → sorting into ascend or descend ^{ord}
 - copy()

$$a = 2$$

$$b = 6$$

$$\begin{matrix} c \\ - \end{matrix} = 2$$



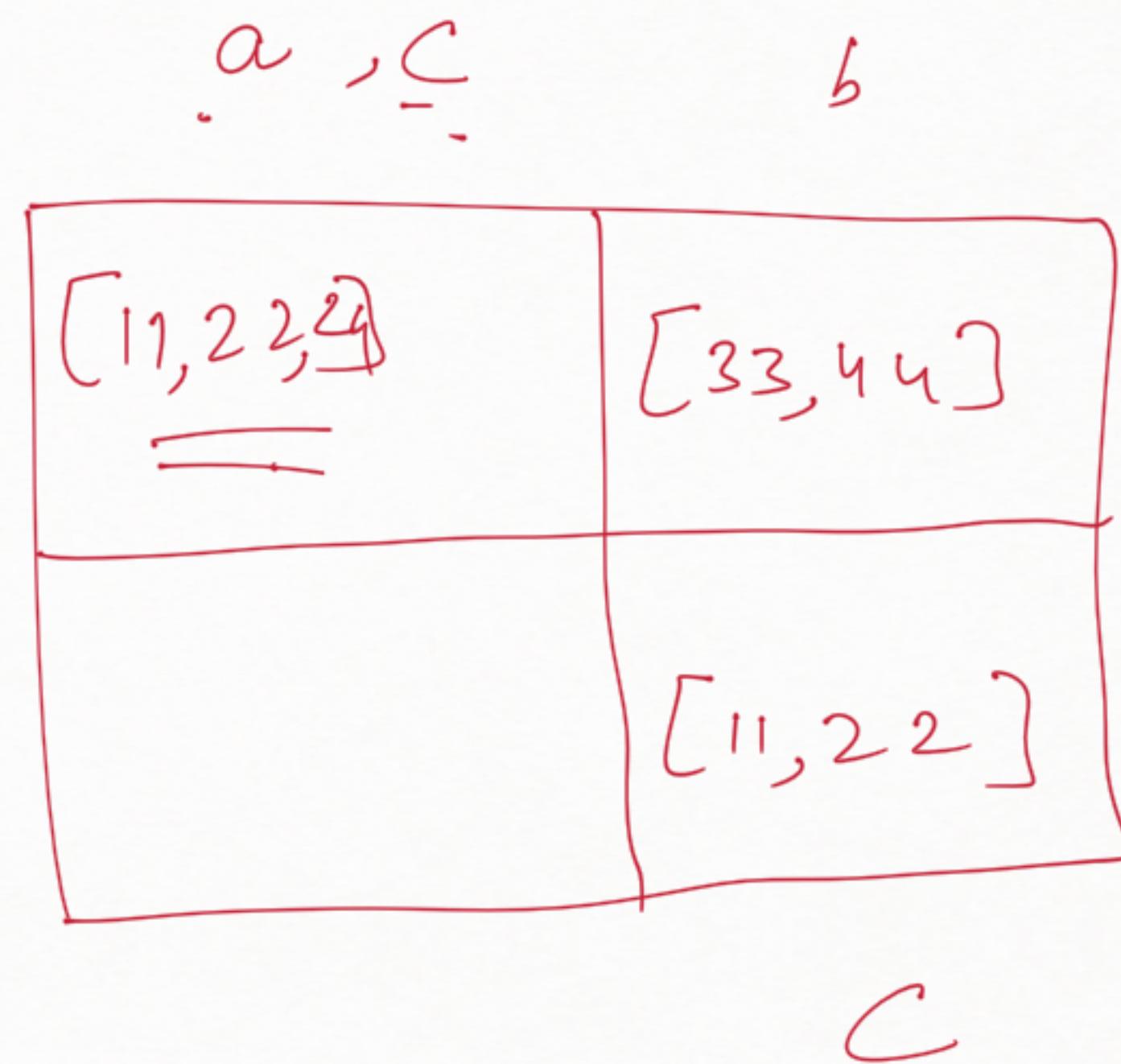
$a = [11, 22]$

$b = [33, 44]$

$c = a$

$a.append(22)$

$c = a.copy()$



c

Tuple : Collection of heterogeneous type of data,
indexed, ordered, immutable (unchangeable).

$a[-1][1]$

$$a = (2, 2.5, \text{True}, 3+9j, [2, 5, 6])$$

$a[0:3:1]$

| | | | | |
|----|-----|------|--------|-------------|
| -5 | -4 | -3 | -2 | -1 |
| 2 | 2.5 | True | $3+9j$ | $[2, 5, 6]$ |
| 0 | 1 | 2 | 3 | |
| -4 | -3 | 2 | -1 | |

$a[-1][::2] \rightarrow [2, 5]$

| | | | |
|----|----|---|----|
| -4 | -3 | 2 | -1 |
| 2 | 5 | 6 | |
| 0 | 1 | 2 | |

$a[0] \rightarrow 2$

$a[-5] \rightarrow 2$

$a[-3] \rightarrow a[2]$

$a[3].real \rightarrow 3.0$

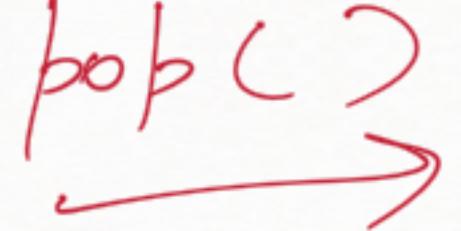
$a[-2].imag \rightarrow 9.0$

Dictionary :- collection of heterogeneous type of
data, unindexed, ordered, mutable...

Key : Value , Key Unique, -- , { }

```
Student = { name : 'Manoj',  
            age : 12,  
            sub : ['maths', 'chem', 'Python']  
        }
```

remove :

{ clear ()
del
popitem ()
pop 

Nested dictionary :-

dictionary inside dictionary

Set :- Collection of immutable - like of data ,
unique , mutable , { } , unordered , indexed

$$a = \{ 8, 2.5, \text{True}, 2+3j, (2,5,6) \}$$

$$a1 = \{ 2, 5, 6, 2, 8, 1 \} \quad \text{error X}$$

$$\{ 2, 5, 6, 8, 1 \}$$

Union

$$S_1 = \{1, 2, 3, 4\}$$

intersection

$$S_2 = \{3, 4, 5, 6\}$$

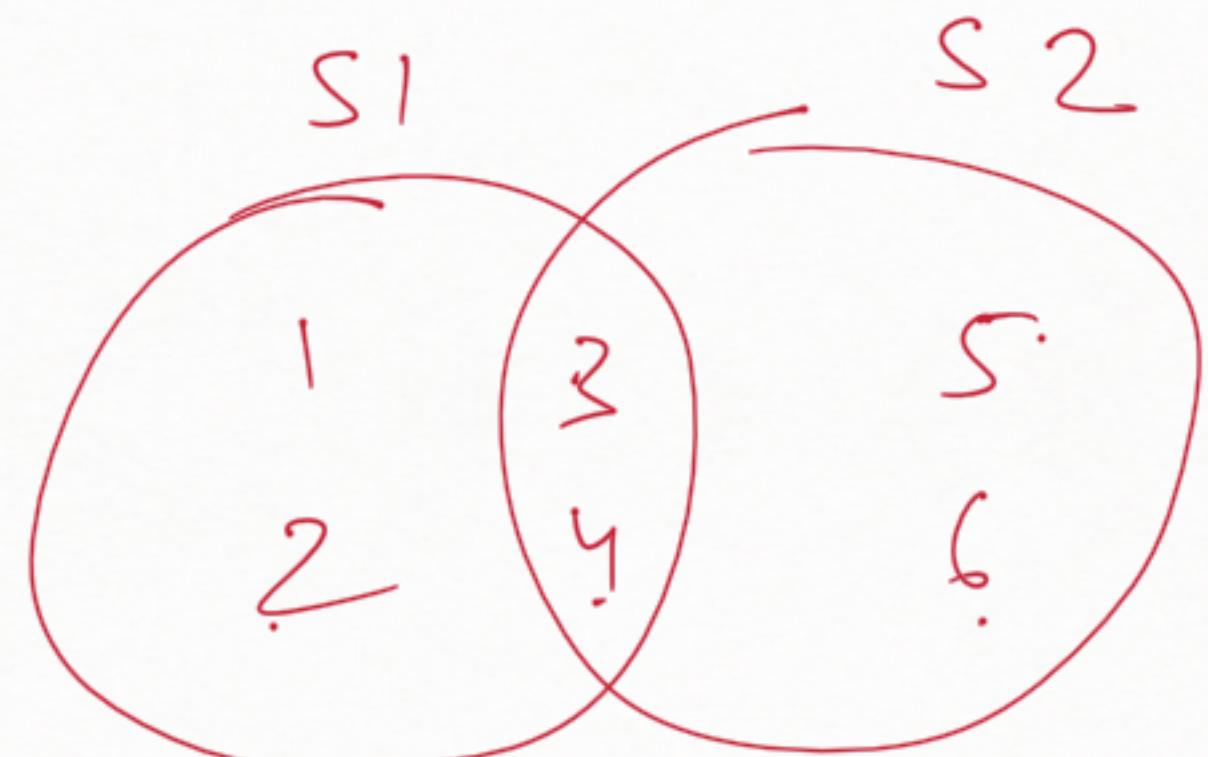
difference

symmetric

$$S_1 - S_2 = \{1, 2\}$$

$$S_2 - S_1 = \{5, 6\}$$

$$S_1 \cap S_2 = \{3, 4\}$$



$$S_1 \cup S_2$$

$$\{ \cancel{1}, \cancel{2}, \cancel{3}, 4, \cancel{3}, 4, 5, 6 \}$$

$$\{ \cancel{1}, \cancel{2}, \cancel{3}, 4, 5, 6 \}$$

adding element in set :

add → 1 element

update → more than 1 el.

del el.

discard()

remove()

pop()

del

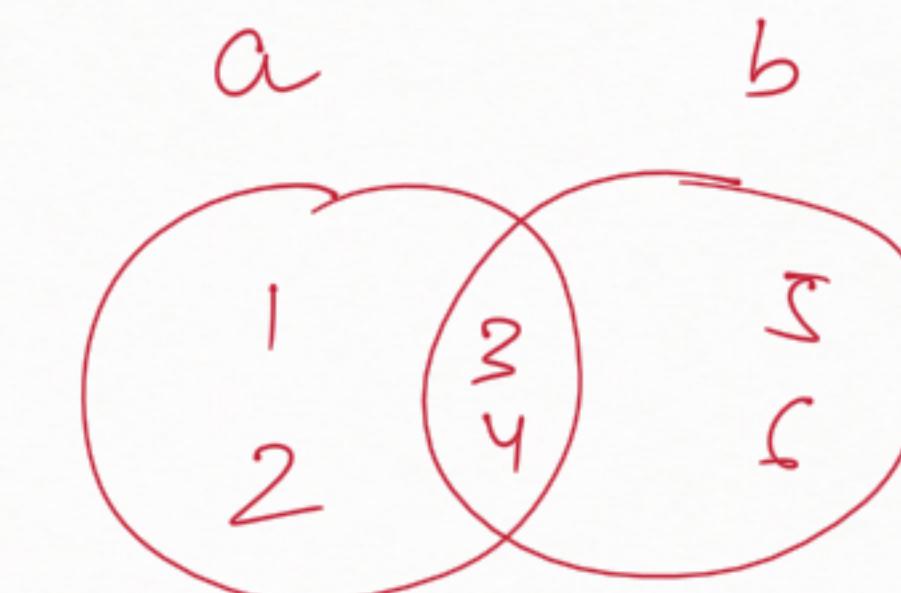
$$a = \{1, 2, 3, 4\}$$

$$b = \{3, 4, 5, 6\}$$

$$a.\text{difference}(b) = \{1, 2\}$$

$$a = \{1, 2, 3, 4\}$$

$$b = \{3, 4, 5, 6\}$$



$$\begin{aligned} a.\text{diff_update}(b) &= \{1, 2\} \\ \uparrow & \end{aligned}$$

$$a = \{1, 2\}$$

$$b = \{3, 4, 5, 6\}$$

Boolean :- True , False
 | 0

logical operator :- and, or, not, xor -----

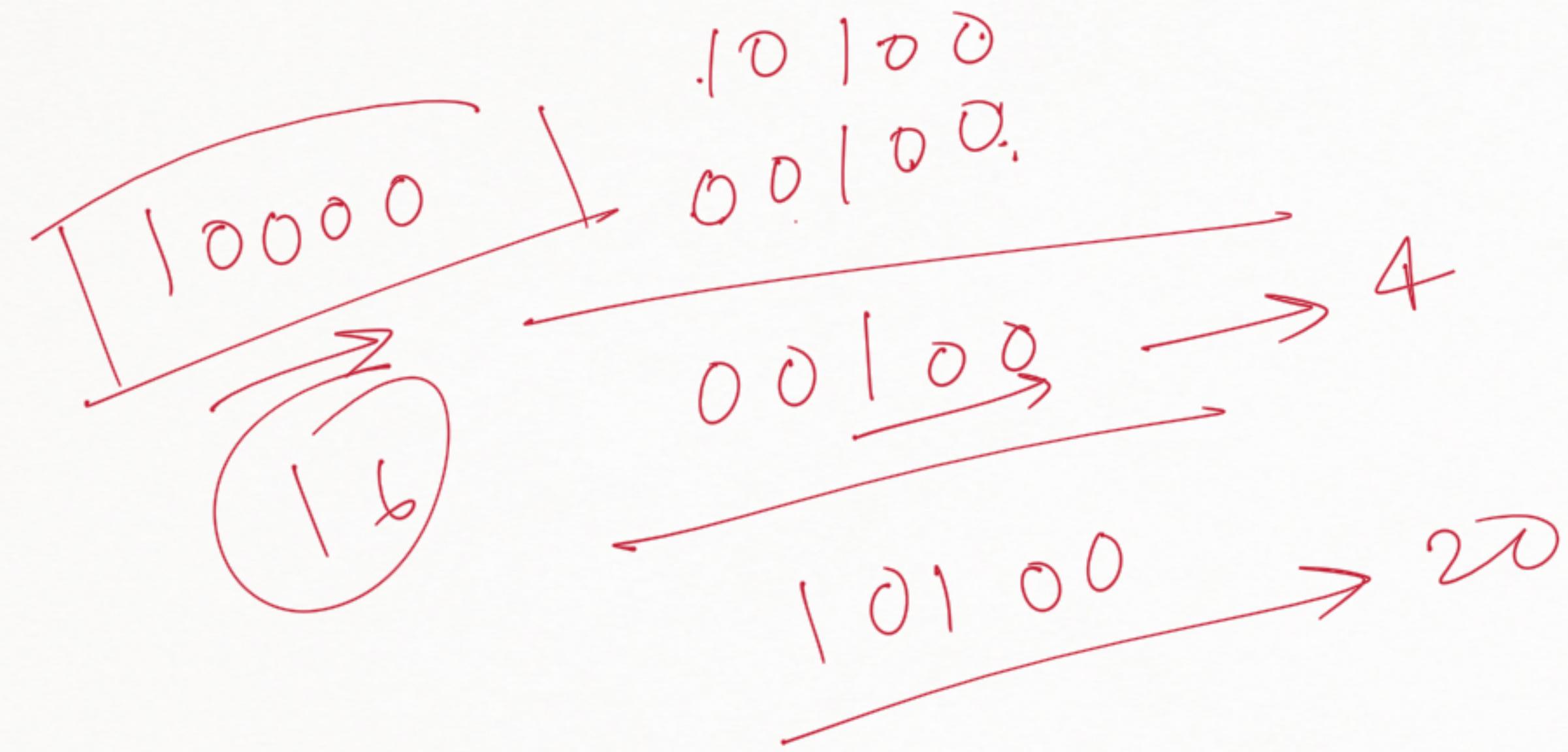
i_1 i_2 and o xor
0 0 0 0 0
0 1 0 1 1
1 0 0 1 1
1 1 1 1 0

not 0 \rightarrow 1
1 \rightarrow 0

$a = 20$

$b = 4$

$a \text{ and } b \Rightarrow 4$



and \rightarrow logical operators

$20 \rightarrow 10100$

$4 \rightarrow 100$

10100

~~2 | 4 |
2 | 2 | 0
1~~

~~2 | 20 |
2 | 10 | 0
2 | 5 | 0
2 | 2 | 1
2 | 1 | 0~~

$a = 2$

$b = 4$

$a+b \rightarrow 6$

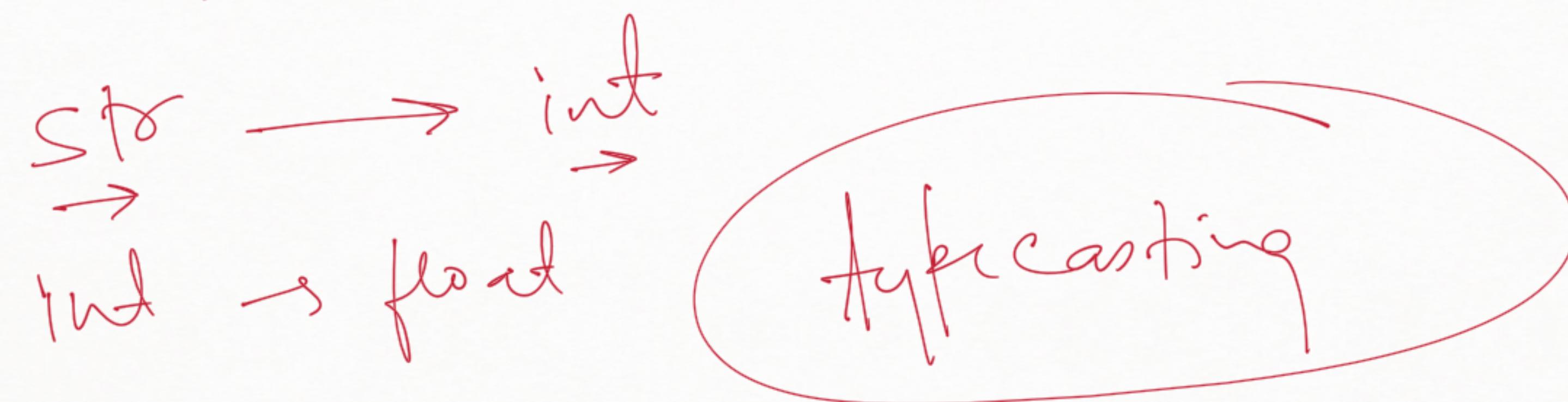
input()



string

str \rightarrow num

process of converting one data-type into
another type of data -



a = 'a'

$a: 'a' \rightarrow \text{int} \quad \text{str}$

`int(a)`

Control Statement:

if

if - else

if - elif - else

nested if

 ↳

if True / false
condition :

{
==
=
=



if - elif - else :

if ^{T/f} condite :

{ == }

→ elif ^{T/f} condition :

{ == }

→ elif ^{T/f} condite :

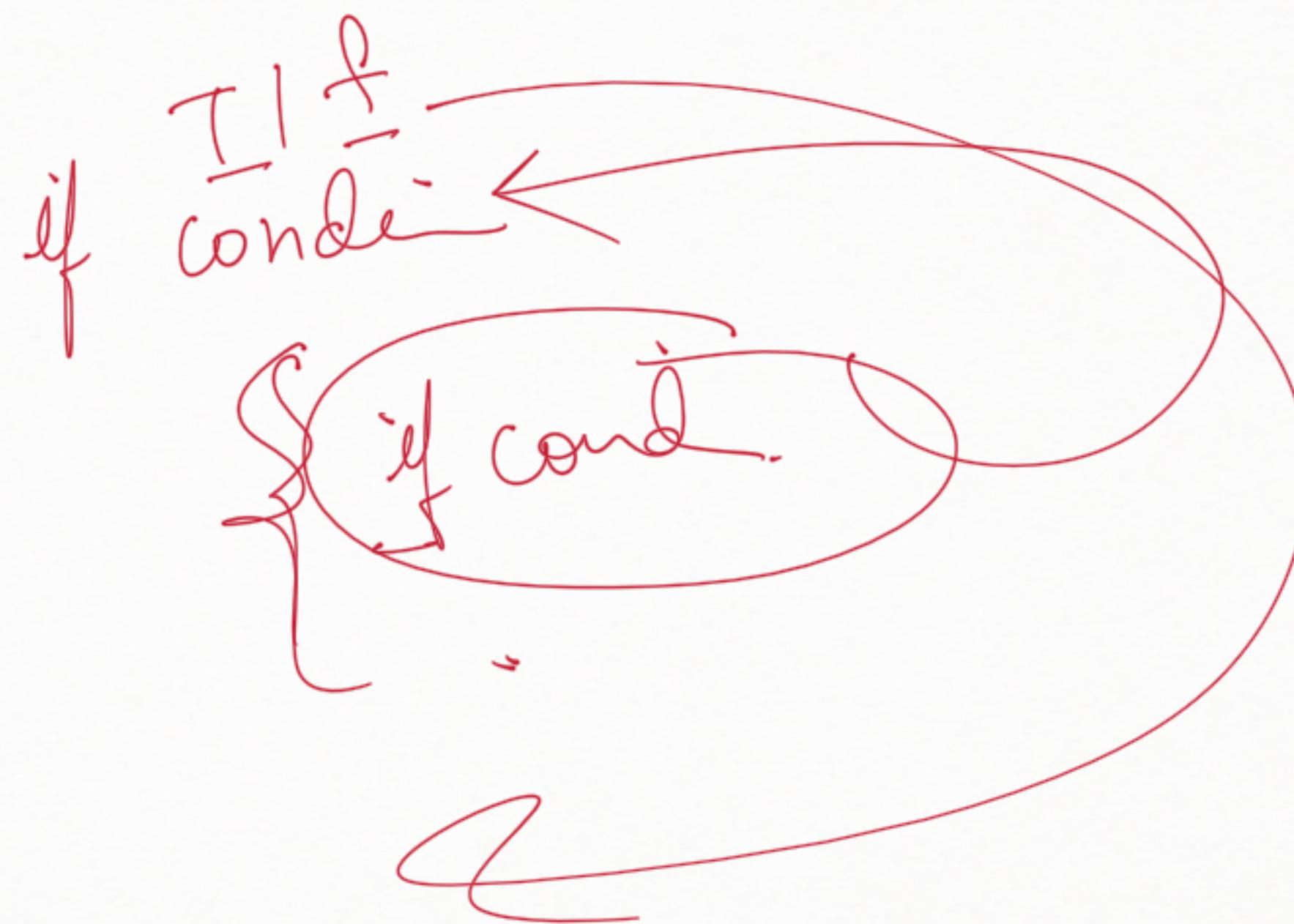
{ == }

→ else :

{ == }

Nested if.

if inside if



$a, b, c = 2, 5, 9$

if ($a \geq b$ and $a \geq c$):

print('a is greater than b and c')
→

$2 \geq 5$ and $2 \geq 9$

f and f → f

Project - I

Quiz Game

Loop :- Repetation of statement

While

for

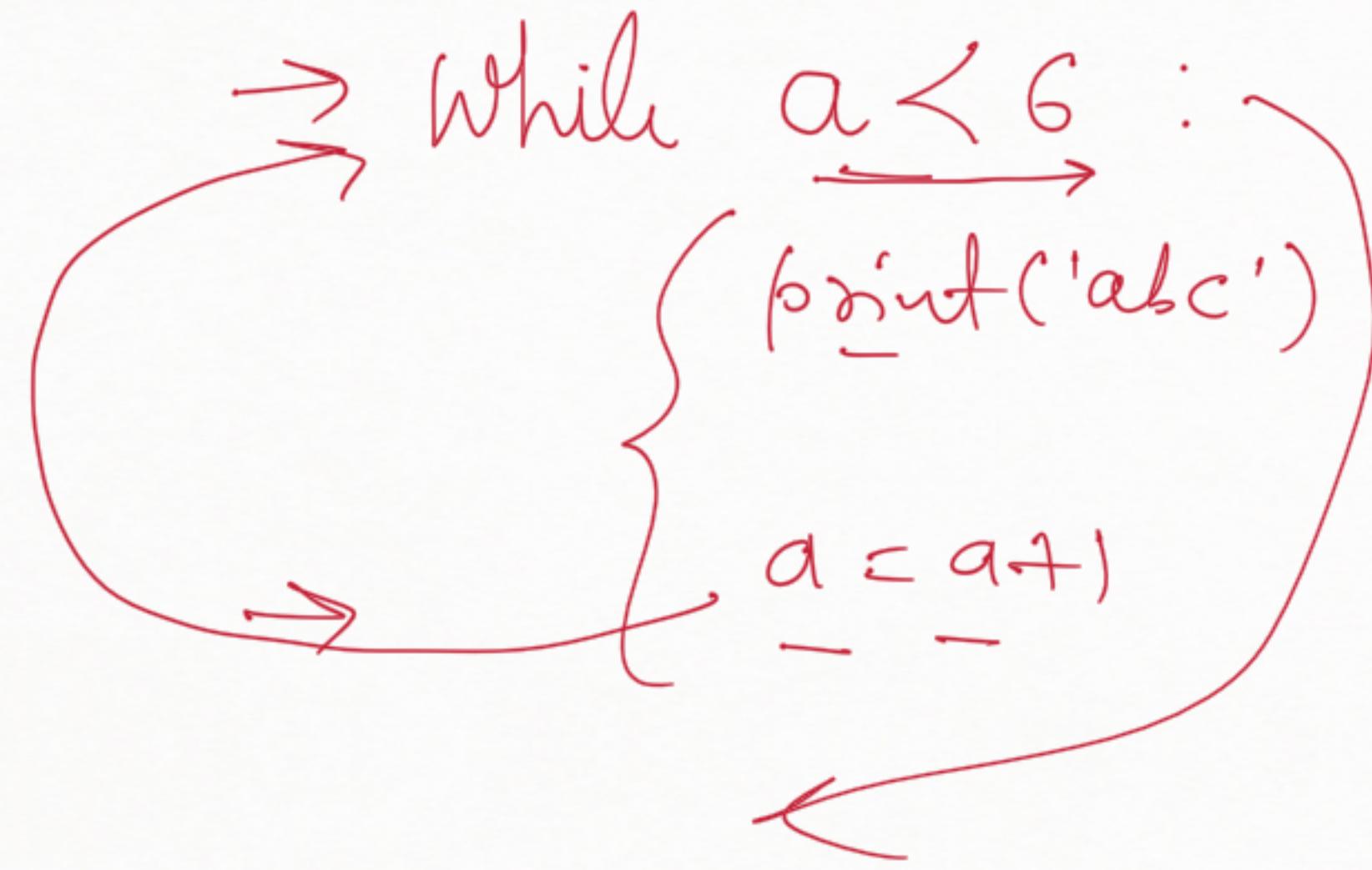
While loop :-

→ abc
ab c }
ab c }
ab c }
ab c }

initialization
condition
iter/dec

$\rightarrow \text{print('abc')}$

$a = 1$



$a = 1$

$1 \leq 6 \ T$

$a = 2$

$2 \leq 6 \ T$

$a = 3$

$3 \leq 6 \ T$

$a = 4$

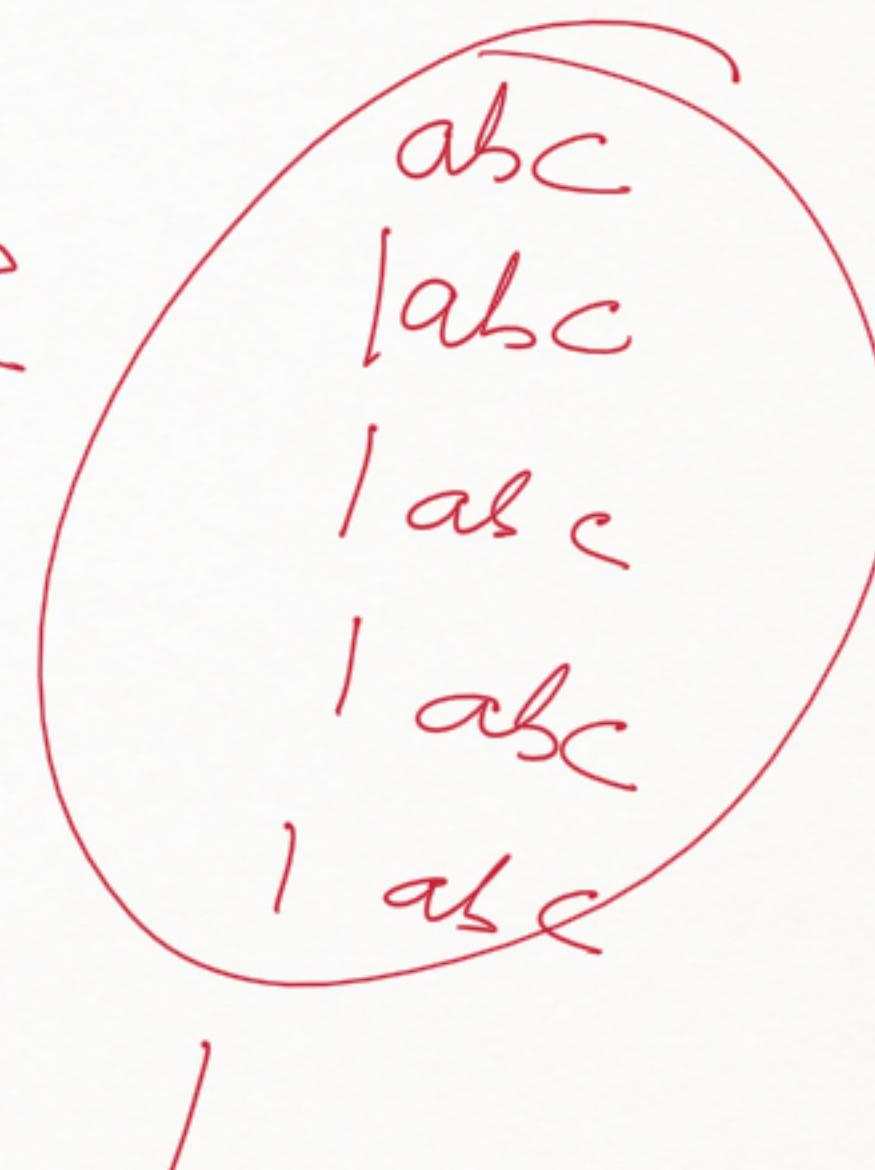
$4 \leq 6 \ T$

$a = 5$

$5 \leq 6 \ T$

$a = 6$

$6 \leq 6 \ F$



$\rightarrow .1:$
 $+1$
 $2+1 \}$
 $3+1$
u.
;
;

$a = 1$
 \rightarrow
while $a < 11:$
 {
 print(a)
 $a = a + 1$ or $a + 1$
 }

10
11
12
13

$a = 1$
 $1 < 11 \top$
 $2 < 11 \top$
 3

$\rightarrow 5 - 1$

$4 - 1$

$3 - 1$

$2 - 1$

!

$a = 5$

\rightarrow While $a > 0$:

$\rightarrow \text{print}(a)$

$a = a - 1$ or $a - 1$

$a = 5$

$5 > 0 \top$

$a = 4$

$4 > 0 \top$

$\begin{array}{c} 5 \\ | \\ 4 \\ | \\ 3 \\ | \end{array}$

$$\rightarrow \begin{array}{r} 2 + 2 \\ - 4 + 2 \\ \hline 6 + 2 \end{array}$$

8

10

12

14

16

18

→ 20

$$a = 2 + 2$$

$$a <= 20 \top$$

$$a = 4$$

$$4 <= 20 \top$$

$$a = 6$$

.

,

/

$$a = 2$$

while $a <= 20$:

→ print(a)

$$a = a + 2$$

$$\begin{array}{r} 2 \\ | 4 \\ 1 \end{array}$$

$$2 \times 1 = 2$$

$$2 \times 2 = 4$$

$$2 \times 3 = 6$$

.

.

.

.

$$2 \times 10 = 20$$

$$a = 1$$

while $a <= 10:$

print(
a * 2)

$$a = 1$$

$$L = 10T$$

$$a + 1$$

$$1 \times 2 = 2$$

$$a = 2$$

$$2 < 10 T$$

$$2 \times 2 = 4$$

1

2

3

4

5

6

7

8

9

10

1
2
3
4

1
2
3
4

1
2
3
4

1
2
3
4

Nested while loop:

while loop inside while loop

while

while

while —

—
—

* * * *

* * * *

* * * *

-

*
* *
* * *
* * * *

1 2 3 4
1. \rightarrow  $\rightarrow a = 1$

2. $\ast \times \ast \ast$

3. $\ast \ast \times \ast$

while $a < 3 :$

$\rightarrow b = 1$

while $b <= 4 :$

$\text{print}(\text{"*"}, \text{end}=\text{" "})$

$b += 1$

$\text{print}()$

$a = a + 1$

| | | | | |
|-----------------|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $\rightarrow 1$ | * | | | |
| $\rightarrow 2$ | * | * | | |
| $\rightarrow 3$ | * | * | * | |
| $\rightarrow 4$ | * | * | * | * |

```

row = 1
while row <= 4:
    col = 1
    while row >= col:
        print("*", end="")
    col += 1
    row += 1

```

| | |
|-----|------------|
| row | col |
| 1 | = 1 |
| 2 | => 1 2 |
| 3 | => 1 2 3 |
| 4 | => 1 2 3 4 |

$\text{print}()$
 $\text{row} += 1$

{ * * *
* * *
* *
*

Try to do it by yourself

all the best



for loop :-

for var in Sequence :
 range
 membership
 operator

Keyword Variable =

list
tuple
set

string

Membership operator :-

in }
not in }
 } True
 } False

$x = [2, 5, 'abc', 29]$

2 in x True

'abc' in x True

21 in x False

5 not in x False
29 not in x True
50 not in x True

Identity operator is $\left\{ \begin{array}{l} \text{is} \\ \text{is not} \end{array} \right\}$ true
false

$$a = 2$$

$$b = 5$$

$$c = 2$$

| a, c | b |
|--------|--------|
| 2 = | 5 - |

$a \text{ is } c$ True

$a \text{ is } b$ false

$$\rightarrow \text{id}(a) \neq \text{id}(b)$$

$\underline{a} \text{ is not } \underline{c}$ False

$a \text{ is not } b$ True

for x in $[2, 8, 5, 'abc']$:

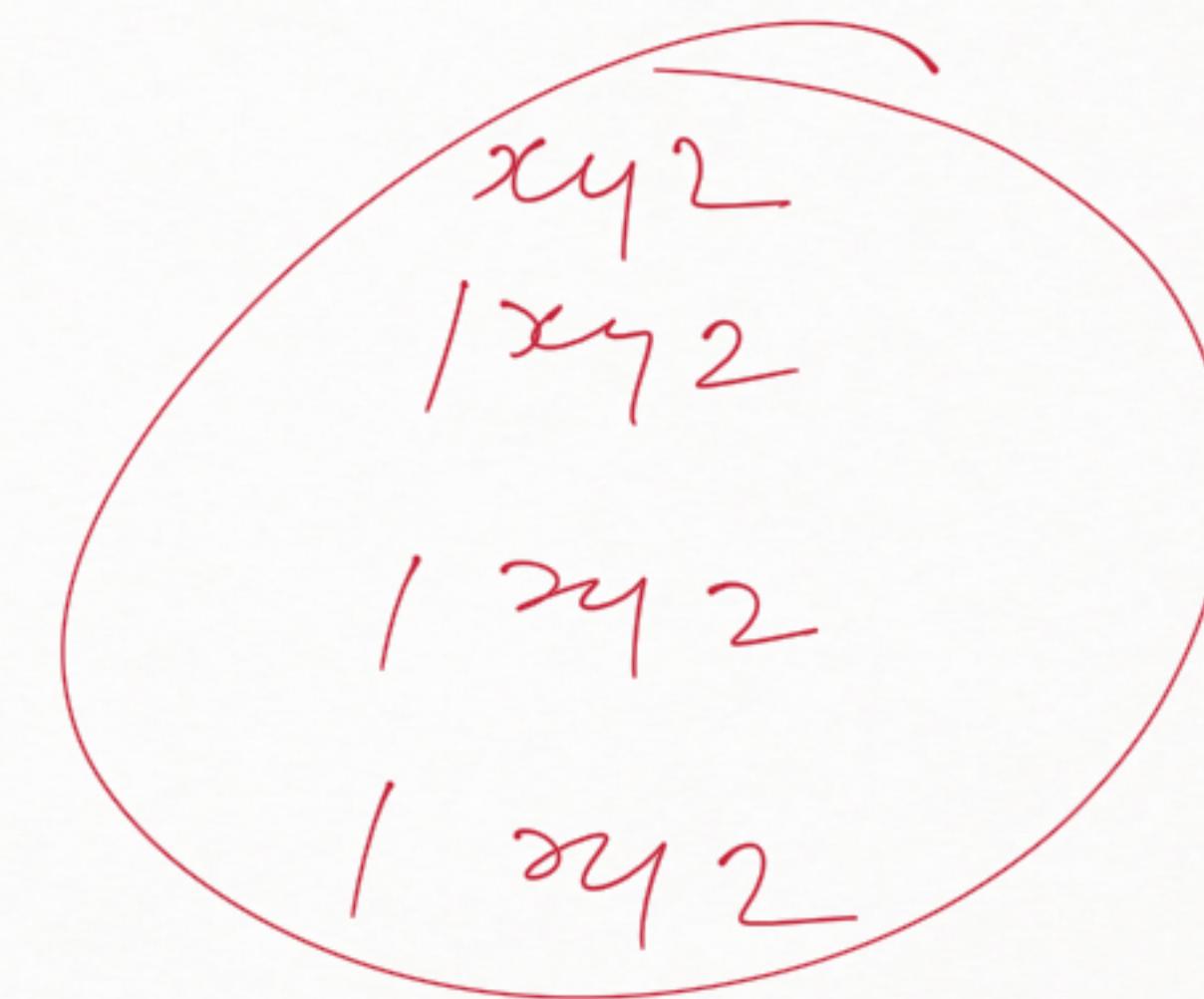
{ point (' $xy2$ ') \Rightarrow

2 in $[2, 8, 5, 'abc']$ True

8 in " " True

5 in " " True

'abc' in " " True



`range(start, stop , step)`

0

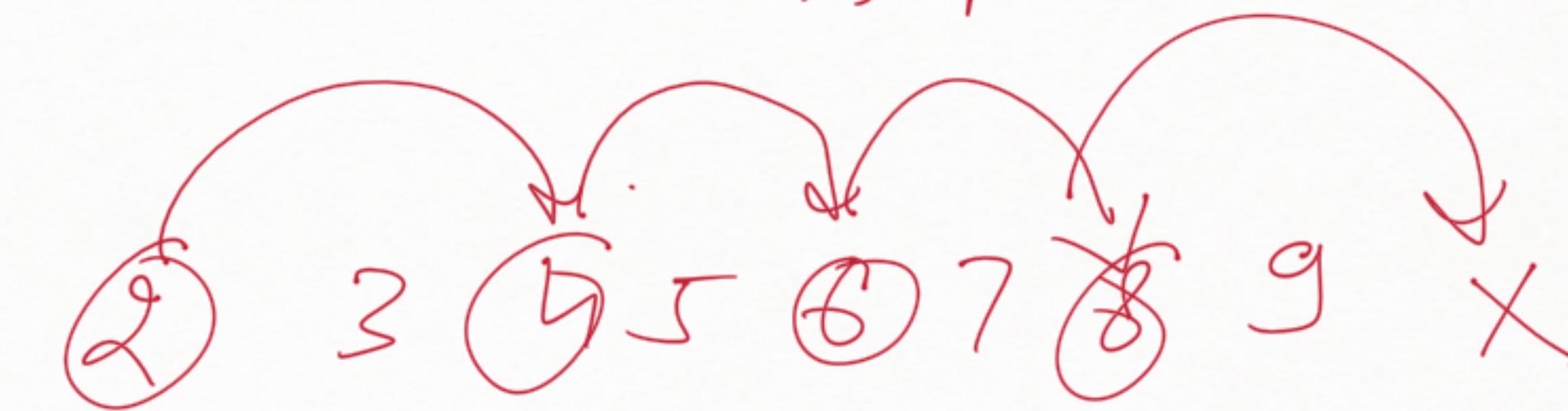
x

+1 -1

`range(0, 5, 1)`

0 1 2 3 4

`range(2, 10, 2)`



(2, 4, 6, 8)

$\text{rang } \tau(10, 2, -1)$

10 9 8 7 6 5 4 3

1 X2

2

3 X5

4

.

,

;

10

for i in range(1, 11):
 print(x)

1

2

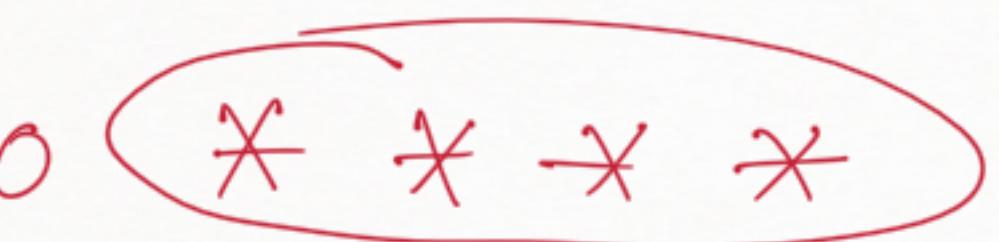
3

4

;

10

Nested for loop :-

0  * * * *

1 * * * *

* * * *

for i in range(3) :

 * * * *

| | | | | |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| 0 | * | * | * | * |
| 1 | * | * | * | * |
| 2 | * | * | * | * |

for row in range(3):

 for col in range(4):

 print("*", end="")

 ()

for row in range(1, 5) :

| | | | |
|---|---|---|-------|
| 0 | → | 1 | * |
| 1 | . | 2 | * * |
| 2 | . | 3 | * * * |
| 3 | . | 4 | * * * |

for col in range(row) :

(n(*, end))

both

* * * *

* * *

* *

*

file Handling :-

Open()

Write()

Read()

close()

mode.

'r' → read

'w' → write / create

'a' → append

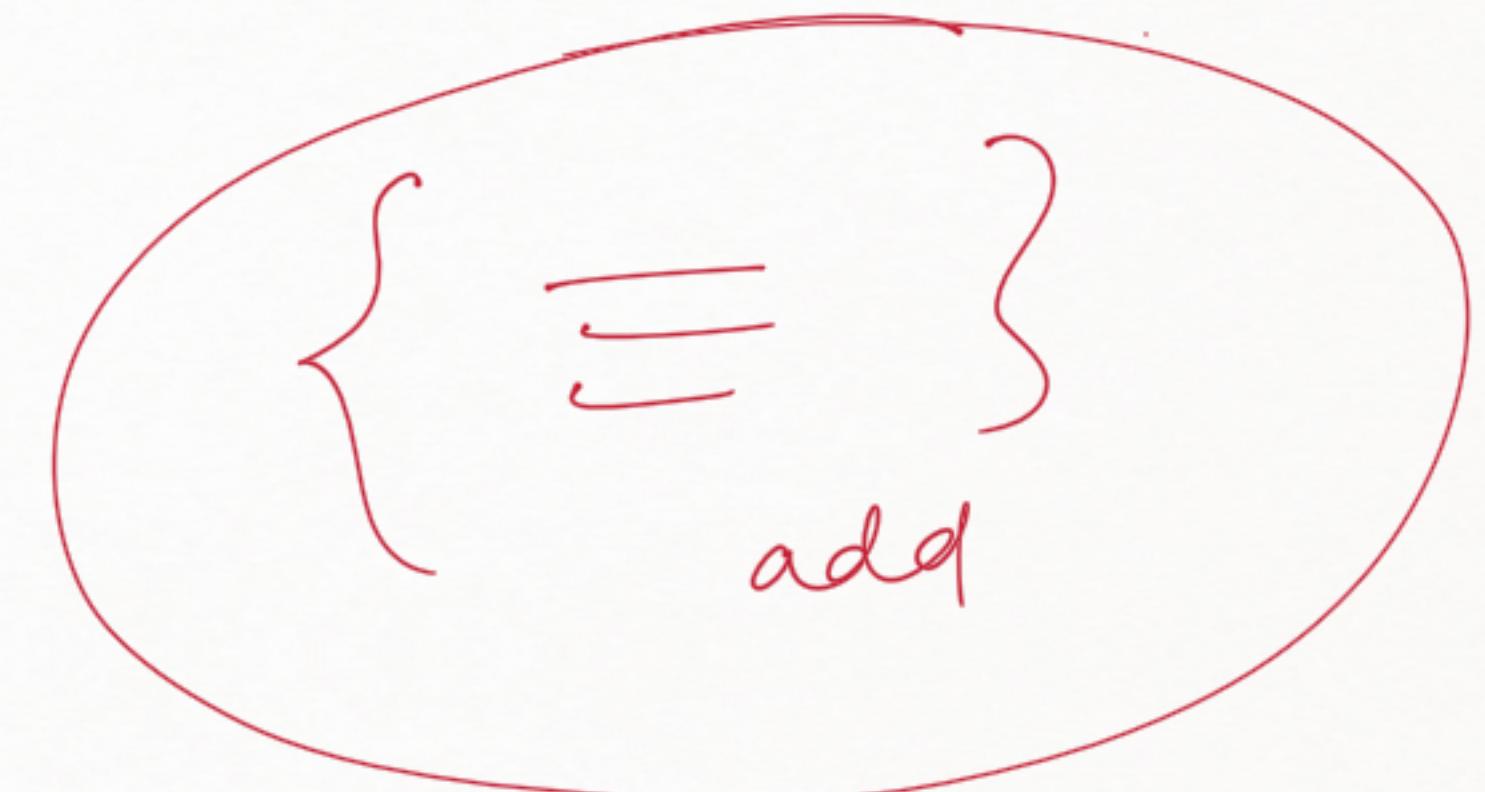
'rb'

'wb' =

function in Python :-

Block of code , by which you can perform a specific task -

plumber \rightarrow  ~~2-3 years~~



defining a function → { ≡ }

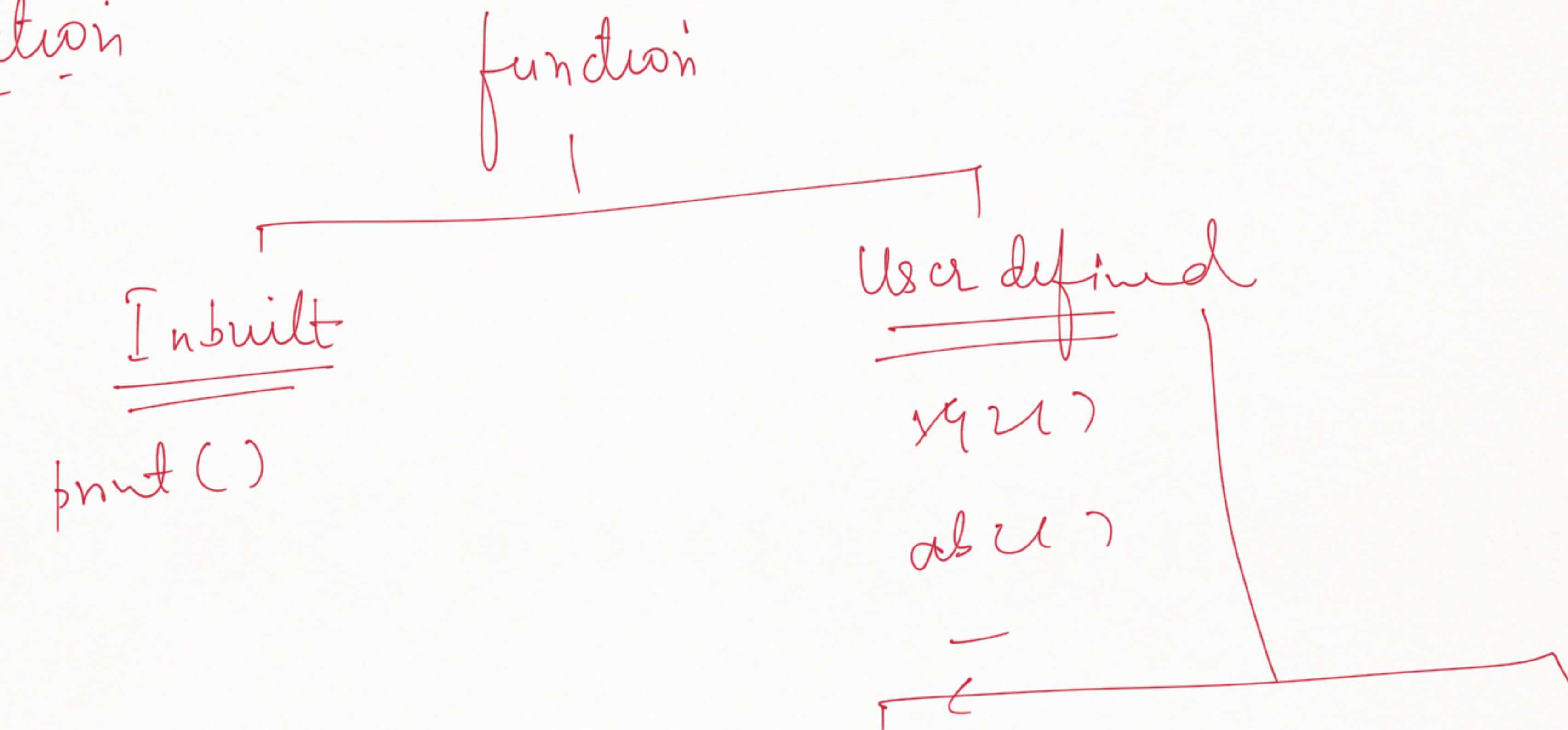
calling a funcn →

function -name()

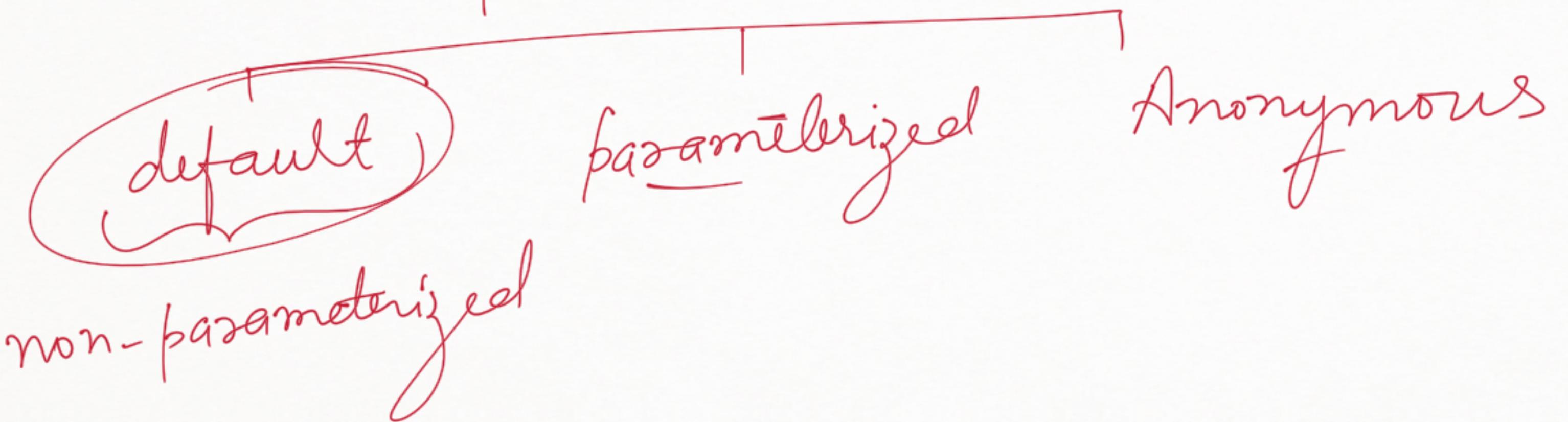
```
def name():
    print ('Hello')
    print ("My name is Malleka")
    print ('Bye')
```

```
name()
name()
Hello
|
|
```

Types of function



user defined



Anonymous function:-

function without name

or

nameless fun^t

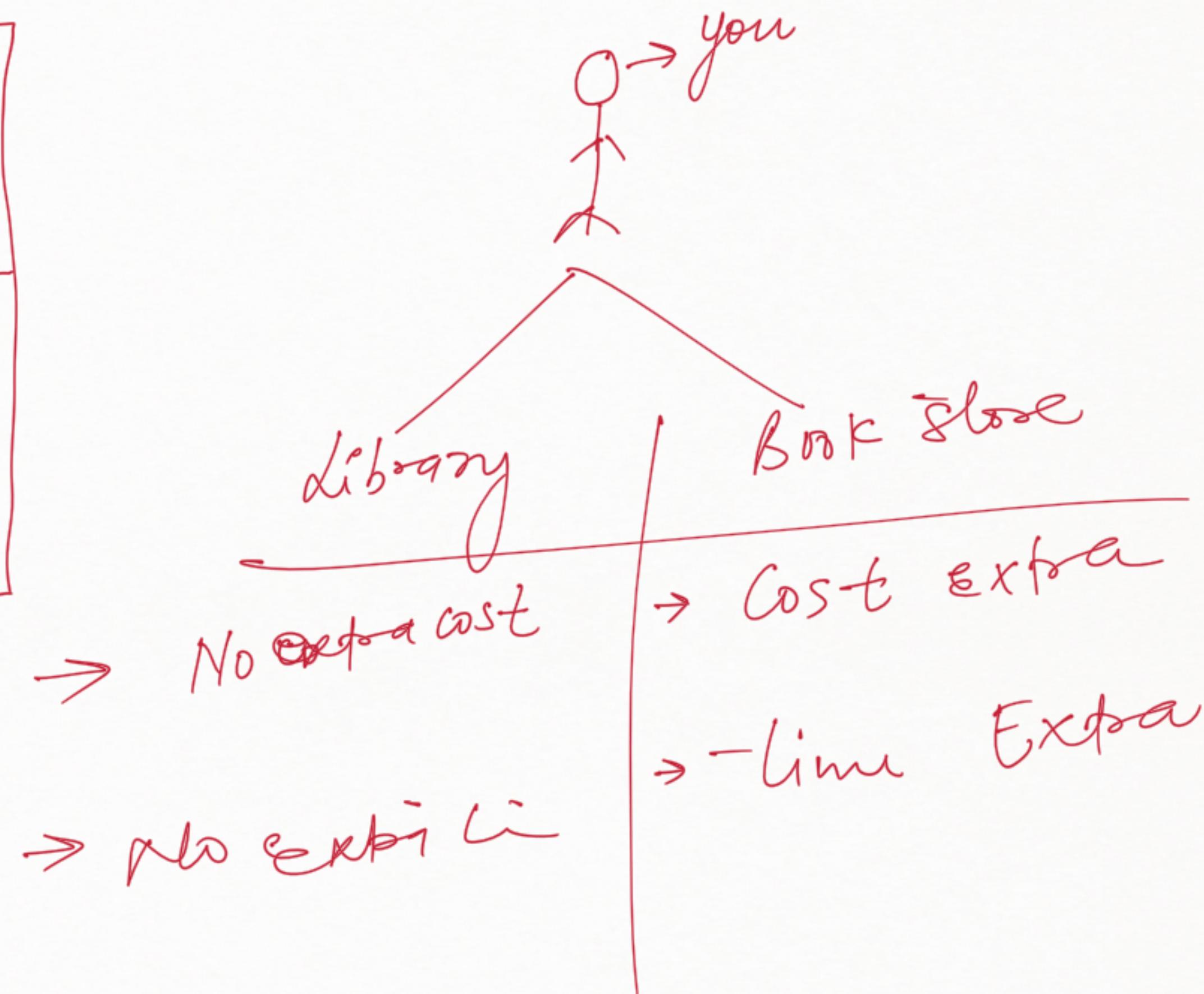
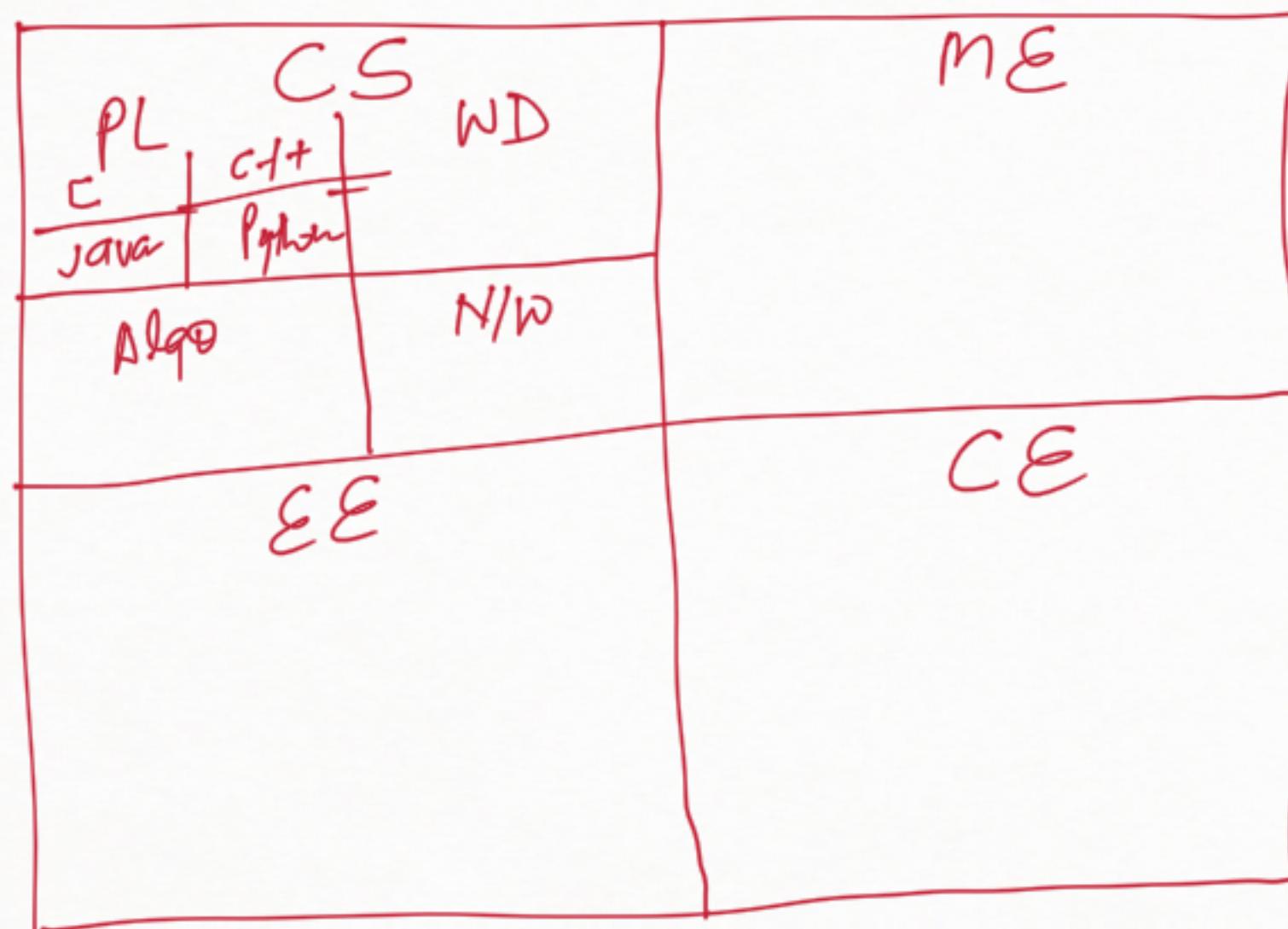
[lambda]

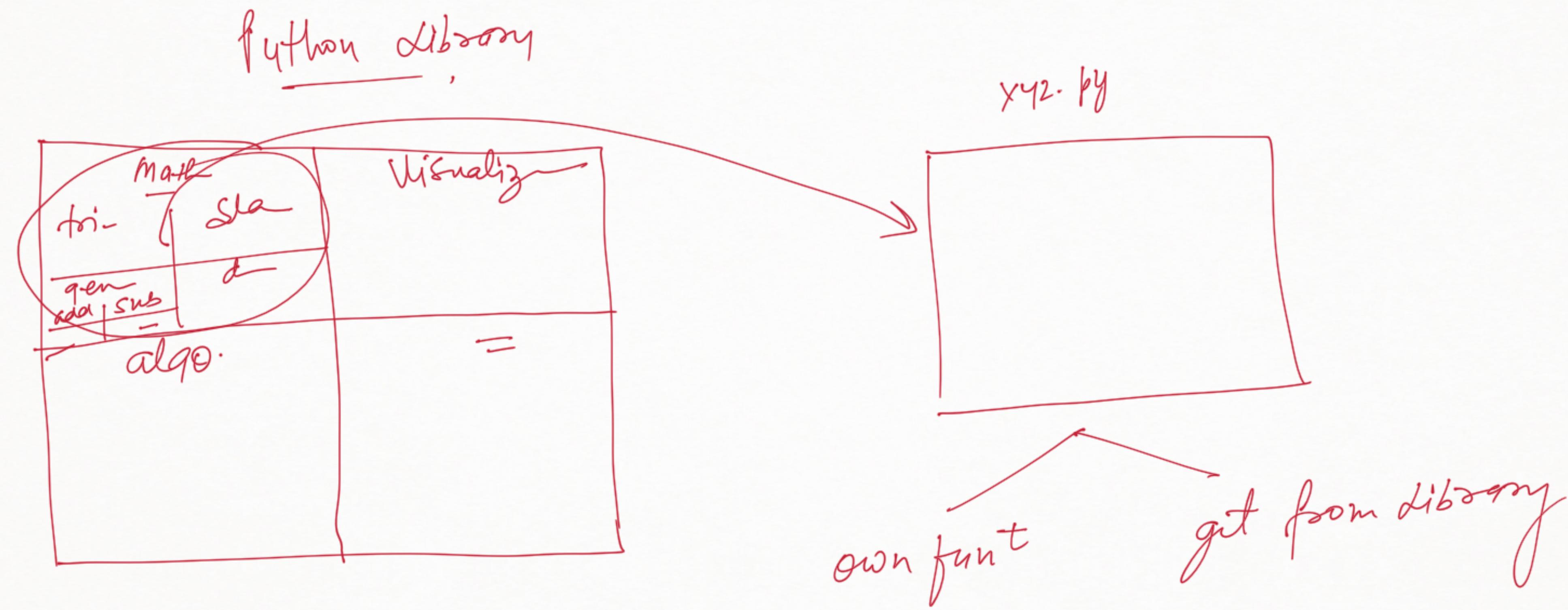
def add(a,b):
 print(a+b)

lambda a,b : a+b

Packages and Modules :-

College library



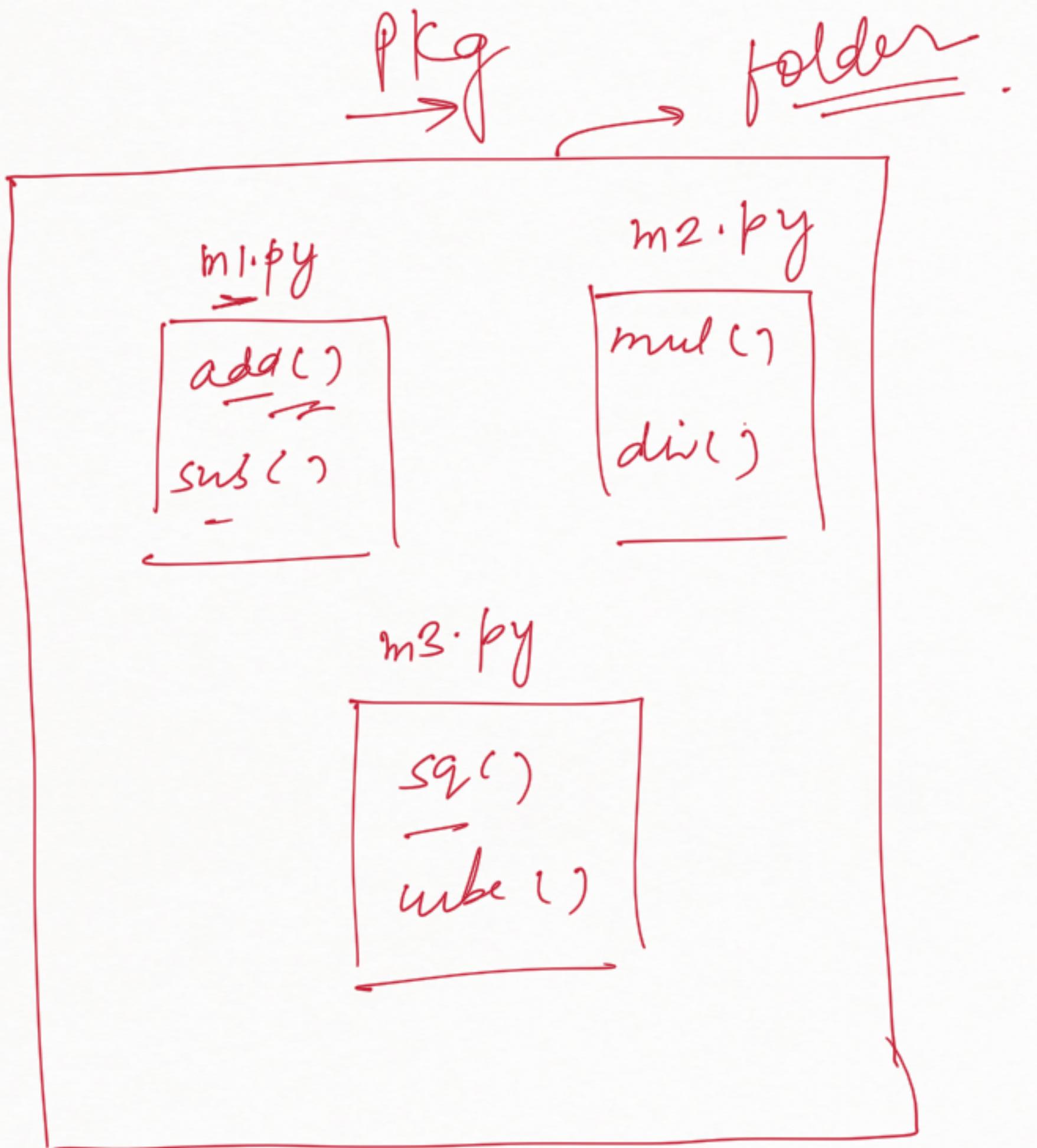


Library :- collection of packages

package :- collection of modules

Module :- collection of functions or -
.py file

function :- _____ .



myfile.py →

```
from pkg import m1, m3  
  
m1.add()  
m1.sub()  
m3.cube()
```