In [3]:
```
a = 3
print(a)
```

```
3
```

Data Structures

Lists

List Slicing

In [ ]:
```
a = [1,2,3,4,5,6,7,8,9,10]

a[3:8:-1]
```

Negative Indexing

Quizzes

Mix negative and postive index

Doubts

Python Refresher - 3

Iteration: Looping

While Loops

In [5]:
```
x=0

while (x < 100):
    x+=2

print(x)
```

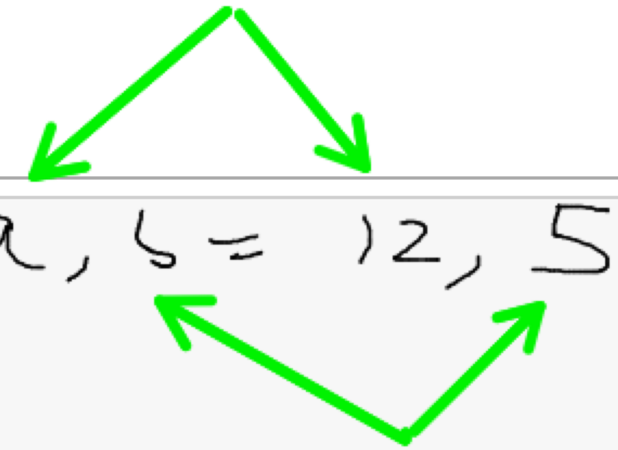
```
100
```

In [ ]:

In [ ]:

DSML Intermediate : Python Refresher - 2

Recap



In [9]:
```
a, b = 12, 5

if a+b:
    print('True')
else:
    print('False')
```

```
True
```

Data Structures

Lists

List Slicing

In [4]:
```
a = [1,2,3,4,5,6,7,8,9,10]

a[3:8:-1]
```

DSML Intermediate : Python Refresher - 2

Recap

In [10]:
```
a, b = 12, 5
print(a)
print(b)
```

0 → F (int)

```
12
5
```

'9' → F (str)

In [11]:
```
print(a + b)
```

0.0 → F (bool)

```
17
```

In [14]:
```
if 0.0:
    print('True')
else:
    print('False')
```

```
False
```

In [ ]:

In [ ]:

```
print(board)
```

```
['', '', '']
```

In [28]:
```
id(board[0]) == id(board[1]) == id (board[2])
```

Out[28]:
```
True
```

In [44]:
```
board = [1, 1, 2]

id(board[0]) == id(board[1]) == id(board[2])
```

Out[44]:
```
False
```

In [47]:
```
id(board) # id of the list
```

Out[47]:
```
140538585904832
```

In [48]:
```
id(board[0])
```

Out[48]:
```
140539114613040
```

In [49]:
```
id(board[1])
```

Out[49]:
```
140539114613040
```

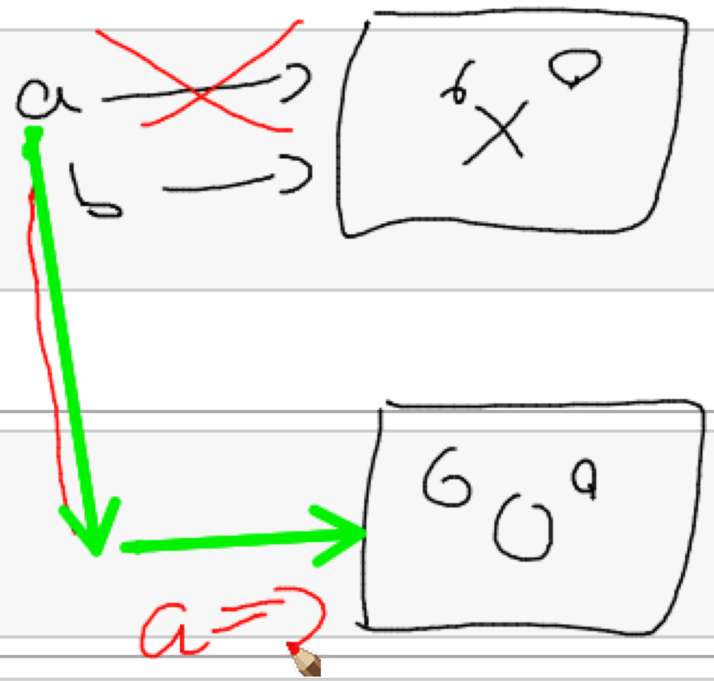
In [ ]:

In [ ]:

In [ ]:

In [56]:
```
a = 'x'
b = 'x'

id(a) == id(b)
```

Out[56]: True



In [ ]:
```
a = '0'

print(a, b)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
0 x
```

In [58]:
```
board = [''] * 3
board[0] = '0'
print(board)
```

```
['0', '', '']
```

In [ ]:

In [59]:
```
board = [['', '', ''], ['', '', ''], ['', '', '']]
```

In [60]:
```
print(board)
```

```
[['', '', ''], ['', '', ''], ['', '', '']]
```

In [62]:
```
board = [['', '', '']] * 3
print(board)
```

```
[['', '', ''], ['', '', ''], ['', '', '']]
```

In [ ]:

In [ ]:

In [ ]:

```
board[0] = '0'
print(board)
```

```
['0', '', '']
```

In [ ]:

In [71]:
```
board = [['', '', ''], ['', '', ''], ['', '', '']]
print(board)
```

```
[['', '', ''], ['', '', ''], ['', '', '']]
```

In [72]:
```
board[0]
```

Out[72]:
```
['', '', '']
```

In [ ]:
```
id(board[0])
id(board[1])
id(board[2])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

, C, C++, and Java

Python 3.6
(known limitations)

```
1  board = [['', '', ''], ['', '', ''], ['', '', '']]
2
3  board[0][0] = '0'
```

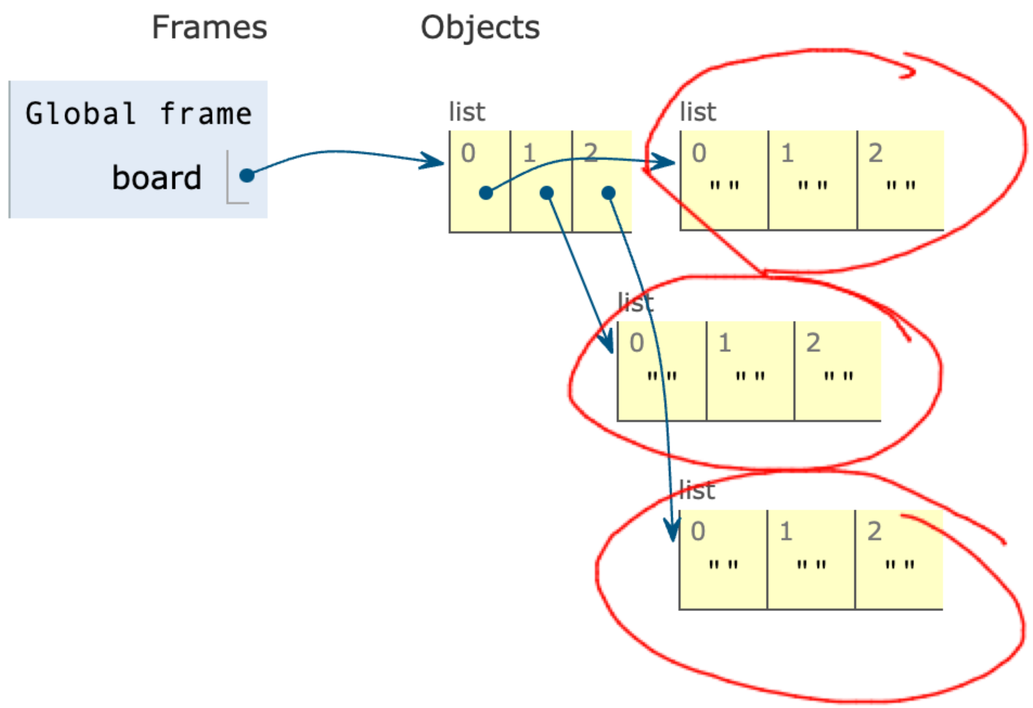
Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 2 of 2

Customize visualization



```
print(id(board[1]))
print(id(board[2]))
```

```
140539129311360
140538585723200
140538585723904
```

In [75]:
```
id(board[0][0]) == id(board[1][1]) == id(board[2][2])
```

Out[75]:
```
True
```

In [77]:
```
for i in range(len(board)):
    for j in range(len(board[0])):
        print(id(board[i][j]))
```

```
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
```

In [ ]:

In [ ]:

In [57]:
```
a = '0'

print(a, b)
```

```
0 x
```

In [58]:
```
board = [''] * 3
board[0] = '0'
print(board)
```

```
['0', '', '']
```

deep copies

In [ ]:

In [71]:
```
board = [['', '', ''], ['', '', ''], ['', '', '']]
print(board)
```

```
[['', '', ''], ['', '', ''], ['', '', '']]
```

In [72]:
```
board[0]
```

Out[72]:
```
['', '', '']
```

In [74]:
```
print(id(board[0]))
print(id(board[1]))
print(id(board[2]))
```

```
140539129311360
140538585723200
140538585723904
```

In [80]:
```
id(board[0]) == id(board[1]) == id(board[2])
```

Out[80]:
```
True
```

In [84]:
```
a = [1, 2, 3]
b = [1, 2, 3] # deep copy => creating a new list

b[1] = 5
print(a)
print(b)
```

```
[1, 2, 3]
[1, 5, 3]
```

In [83]:
```
a = [1, 2, 3]
b = a

b[1] = 5
print(a)
print(b)
```

```
[1, 5, 3]
[1, 5, 3]
```

In [ ]:

In [ ]:

Out[80]: True

In [86]:
```
a = [1, 2, 3]
b = [1, 2, 3] # deep copy => creating a new list => different object

b[1] = 5
print(a)
print(b)
```

```
[1, 2, 3]
[1, 5, 3]
```

In [85]:
```
a = [1, 2, 3]
b = a   # shallow copy => the same object in the memory

b[1] = 5
print(a)
print(b)
```

```
[1, 5, 3]
[1, 5, 3]
```

In [ ]:

In [ ]:

In [42]:
```
# board = [123252] * 3
# print(board)
```

Strings => Immutable

Lists => Mutable

List Multiplication

Up Next

0. HW to be discussed
1. Tuples
2. Sets and Dictionaries
3. Lists => Deep Copy
4. Comprehension
5. Some more important methods in the list, set, tuples, dict

Doubts

Python Refresher - 3

Iteration: Looping

```
l.append(10)
```

In [93]:
```
print(l)
```

```
[4, 6, 7, 10]
```

In [94]:
```
l.append([1, 2, 3])
print(l)
```

```
[4, 6, 7, 10, [1, 2, 3]]
```

In [95]:
```
len(l)
```

Out[95]: 5

last object

In [ ]:
```
l.pop() # what does pop return?
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
[4, 6, 7, 10, [1, 2, 3]]
```

In [111]:
```
len(l)
```

Out[111]: 5

In [112]:
```
l.pop() # what does pop return? => last element
```

Out[112]:
```
[1, 2, 3]
```

In [113]:
```
print(l)
```

```
[4, 6, 7, 10]
```

In [115]:
```
# l.pop(7) # pop(index)
```

In [116]:
```
l.remove(7)
```

In [117]:
```
print(l)
```

```
[4, 6, 10]
```

In [ ]:

In [ ]:

In [135]:
```
4 in l # citizenship / membership operator
```

Out[135]: True

In [136]:
```
7 in l
```

Out[136]: False

In [137]:
```
l = [35, 12, 56, 77]
```

Iteration Protocol

In [138]:
```
# whether some thing is iterable
```

In [139]:
```
iter(l)
```

Out[139]: <list_iterator at 0x7fd1b0a7f8b0>

In [ ]:

In [ ]:

In [ ]:

Tuples

Not Trusted | Python 3 (ipykernel)

```
In [167]: dir(l)

Out[167]: ['__add__',
           '__class__',
           '__class_getitem__',
           '__contains__',
           '__delattr__',
           '__delitem__',
           '__dir__',
           '__doc__',
           '__eq__',
           '__format__',
           '__ge__',
           '__getattribute__',
           '__getitem__',
           '__gt__',
           '__hash__',
           '__iadd__',
           '__imul__',
           '__init__',
           '__init_subclass__',
           '__iter__',
           '__le__',
           '__len__',
           '__lt__',
           '__mul__',
           '__ne__',
           '__new__',
           '__reduce__',
           '__reduce_ex__',
```

all fn of list object l

In [ ]:

In [ ]:

BODMAS

Creation of Tuples

In [182]:
```
t = (30)

type(t)
```

Out[182]: int

(50 + 30 − 7 +3)

50 + 30 − 21

= 80 − 21

= 59

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Out[191]: tuple

In [193]:
```
t = (1,)
type(t)
```

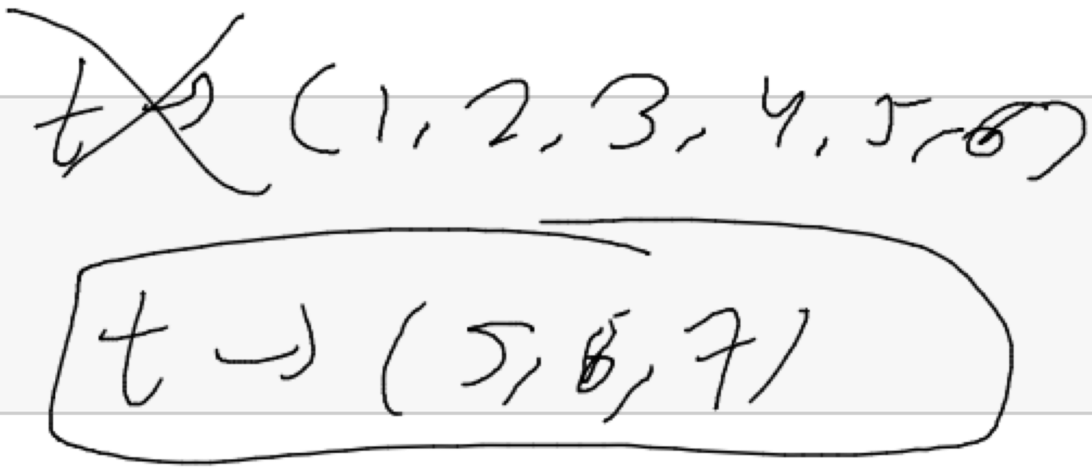
Out[193]: tuple

In [195]:
```
t[0] = 5 # immutable
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [195], in <cell line: 1>()
----> 1 t[0] = 5

TypeError: 'tuple' object does not support item assignment
```

In [196]:
```
t = (1, 2, 3, 5, 6)

t = (5, 6, 7)

print(t)
```

```
(5, 6, 7)
```



In [ ]:

In [193]:
```
t = (1,)
type(t)
```

Out[193]:
```
tuple
```

In [195]:
```
t[0] = 5 # immutable
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [195], in <cell line: 1>()
----> 1 t[0] = 5

TypeError: 'tuple' object does not support item assignment
```

In [197]:
```
t = (1, 2, 3, 5, 6)
print(id(t))

t = (5, 6, 7)
print(id(t))
print(t)
```

```
140538590094656
140538588030656
(5, 6, 7)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Tuples Lecture

Unpacking

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Not Trusted | Python 3 (ipykernel)

Out[218]: tuple

In [219]:
```
def foo():
    return 100, 200, 300
```

In [221]:
```
x = foo()
x
```

Out[221]: (100, 200, 300)

packed tuple

In [ ]:
```
we, are, together = foo()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [227]:
```
we, are, together = foo()
```

In [223]:
```
print(we)
```

```
100
```

In [224]:
```
print(are)
```

```
200
```

In [225]:
```
print(together)
```

```
300
```

1 => (12, 5)

2 => a, b = (12, 5)

In [ ]:
```
a, b = 12, 5
```

In [ ]:

In [ ]:

Swap

In [ ]:

Swap

In [236]:
```
a, b = 1, 2
```

In [237]:
```
print(a)
```

```
1
```

In [238]:
```
print(b)
```

```
2
```

In [239]:
```
a, b = b, a
```

In [240]:
```
print(a)
```

```
2
```

In [241]:
```
print(b)
```

```
1
```

In [ ]:

In [ ]:

How ?

1. packing
(b, a)
(2, 1)

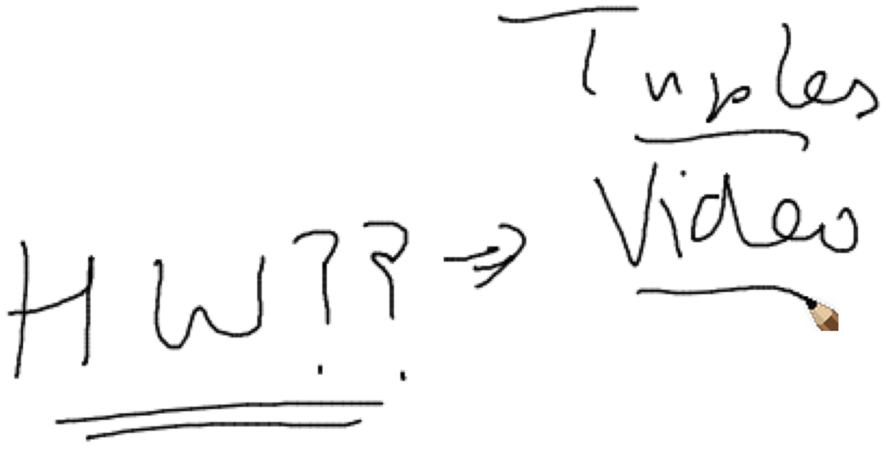
2. Unpacking.
a, b = (2, 1)
a => 2
b => 1

Hint: Think about mutability and immutability

Strings => Immutable

Lists => Mutable

List Multiplication



Up Next

0. HW to be discussed
1. Tuples
2. Sets and Dictionaries
3. Lists => Deep Copy
4. Comprehension
5. Some more important methods in the list, set, tuples, dict

Doubts

Python Refresher - 3

Iteration: Looping

In [260]:
```
# step 1: packing into tuple

x = b, a
```

In [263]:
```
print(a, id(a))
```

```
1999 140538590154544
```

In [264]:
```
print(b, id(b))
```

```
2999 140538590154000
```

In [265]:
```
print(x)
```

```
(2999, 1999)
```

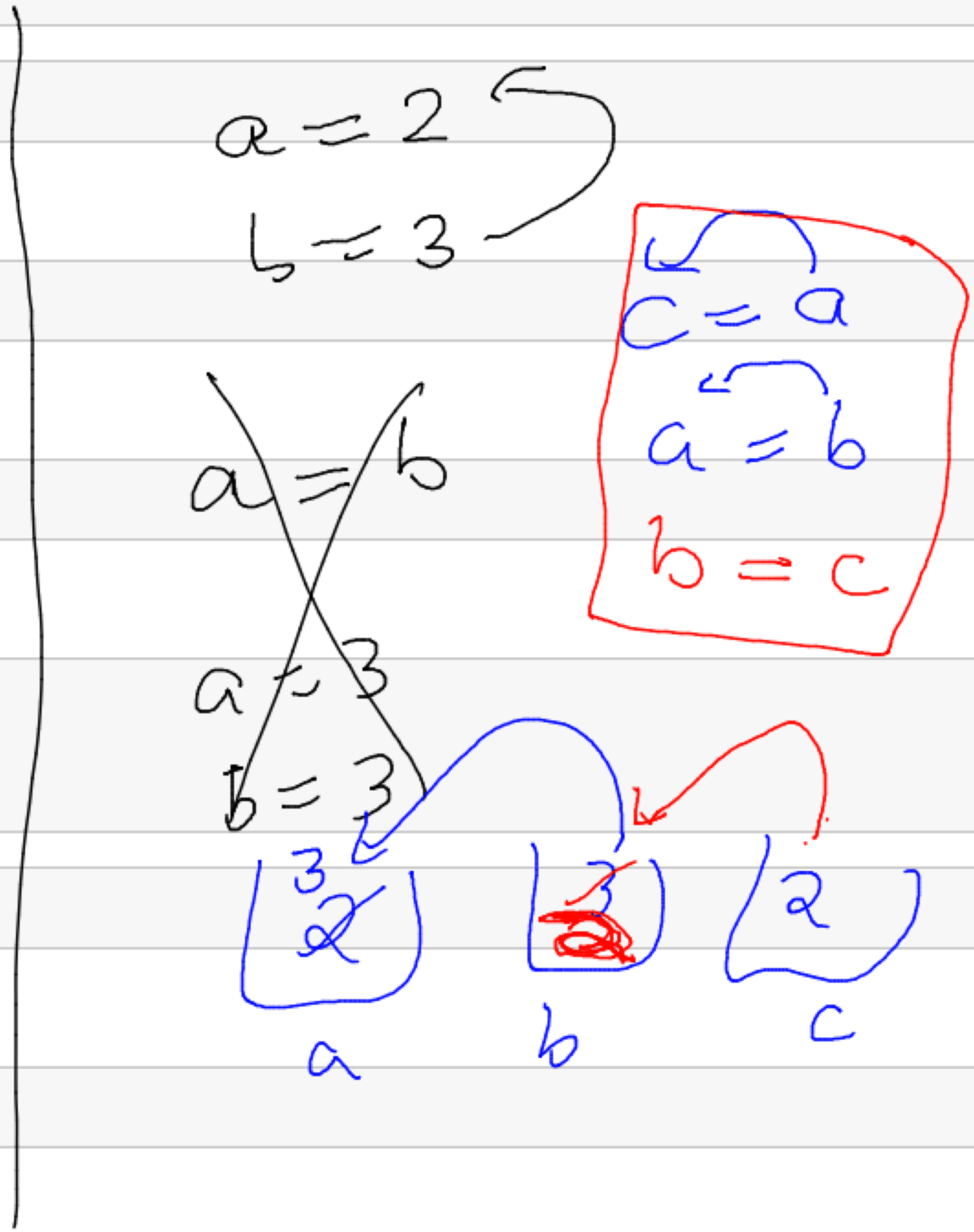
In [266]:
```
# step 2: unpacking from tuple

a, b = x
```

In [267]:
```
print(a, id(a))
```

```
2999 140538590154000
```

In [268]:
```
print(b, id(b))
```

```
1999 140538590154544
```



```
['0', '', '']
```

In [ ]:

In [71]:
```
board = [['', '', ''], ['', '', ''], ['', '', '']]
print(board)
```

```
[['', '', ''], ['', '', ''], ['', '', '']]
```

In [72]:
```
board[0]
```

Out[72]:
```
['', '', '']
```

6 9 => same

In [74]:
```
print(id(board[0]))
print(id(board[1]))
print(id(board[2]))
```

```
140539129311360
140538585723200
140538585723904
```

In [75]:
```
id(board[0][0]) == id(board[1][1]) == id(board[2][2])
```

Out[75]:
```
True
```

In [78]:
```
for i in range(len(board)):
    for j in range(len(board[0])):
        print(id(board[i][j]))
```

```
print(id(a))
```

```
140538588074944
140538588074944
```

deep diff

In [292]:
```
l = [2] * 3

print(l)
```

```
[2, 2, 2]
```

shallow copy

In [295]:
```
l = [[2] * 3] * 3
print(l)
```

```
[[2, 2, 2], [2, 2, 2], [2, 2, 2]]
```

same

In [296]:
```
id(l[0]) == id(l[1]) == id(l[2])
```

Out[296]: True

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
print(id(a))
```

```
140538588074944
140538588074944
```

In [292]:
```
l = [2] * 3

print(l)
```

```
[2, 2, 2]
```

In [308]:
```
l = [[2] * 3] * 3
# [2] * 3 => shallow copy of the list
print(l)
```

```
[[2, 2, 2], [2, 2, 2], [2, 2, 2]]
```

In [309]:
```
l[0][0] = 5
```

In [310]:
```
print(l)
```

```
[[5, 2, 2], [5, 2, 2], [5, 2, 2]]
```

In [ ]:

In [ ]:

In [300]:
```
id(l[0]) == id(l[1]) == id(l[2])
```

```
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
```

In [312]:
```
l = [[5, 6, 7], [8, 9,  10]]
```

In [313]:
```
id(l)
```

Out[313]: 140538588078016

In [314]:
```
id(l[0])
```

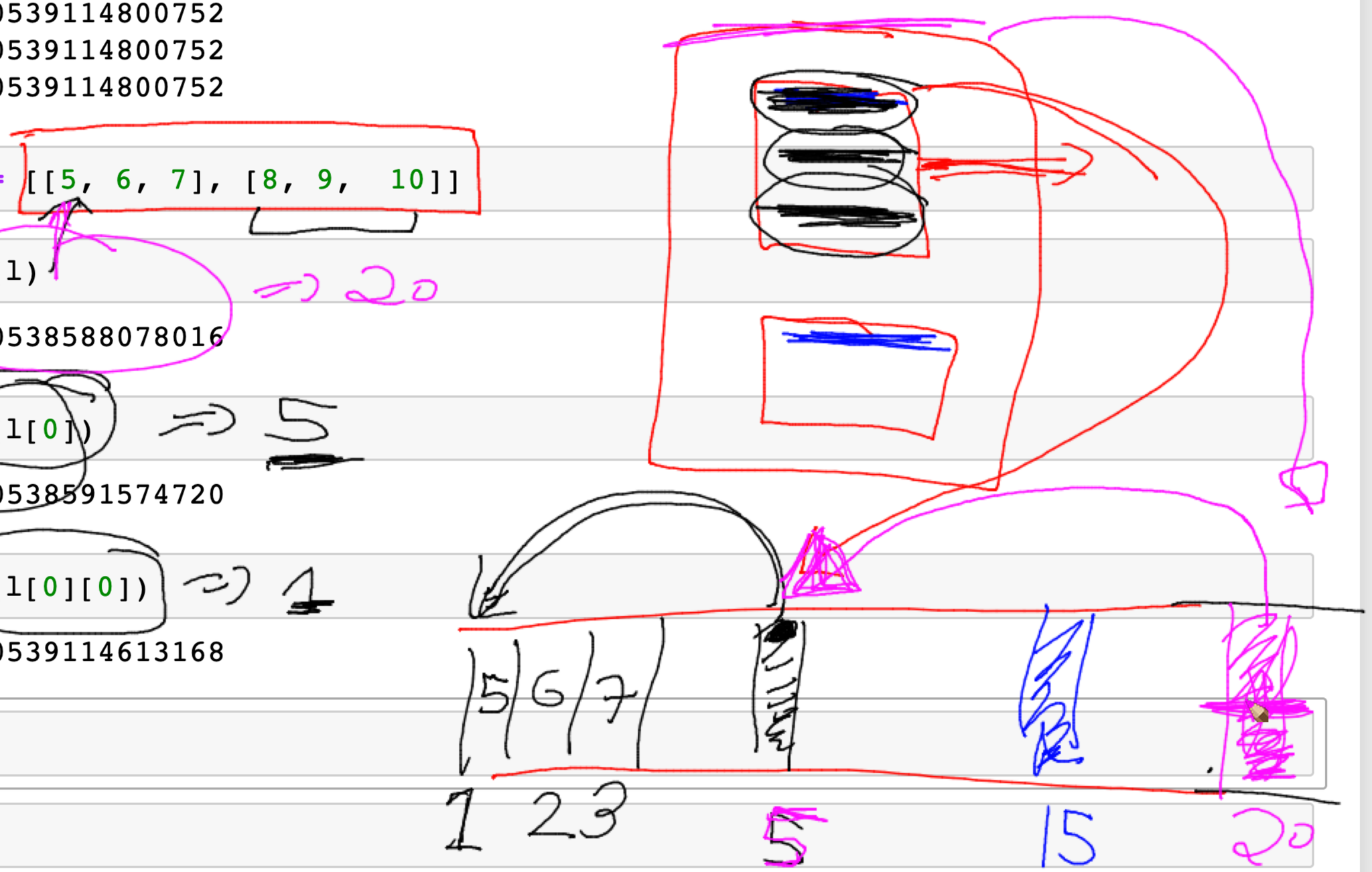
Out[314]: 140538591574720

In [317]:
```
id(l[0][0])
```

Out[317]: 140539114613168

In [ ]:

In [ ]:



Out[72]:
```
['', '', '']
```

In [74]:
```
print(id(board[0]))
print(id(board[1]))
print(id(board[2]))
```

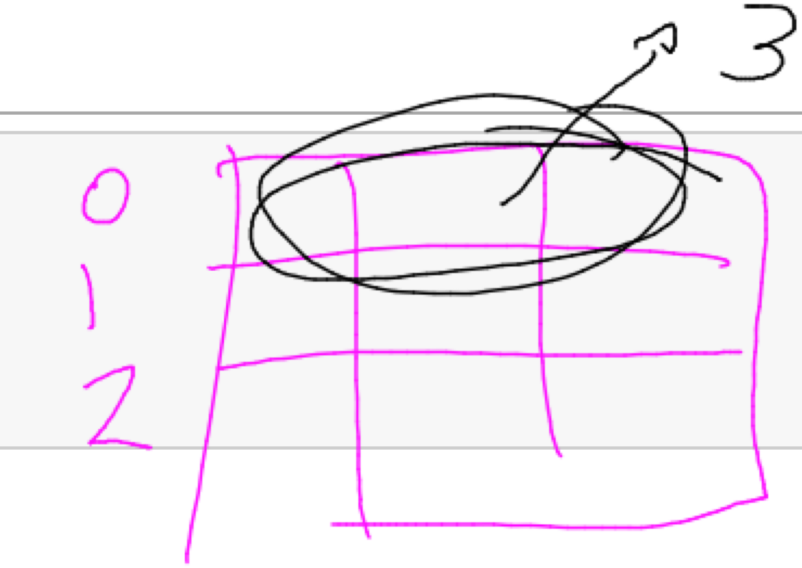
```
140539129311360
140538585723200
140538585723904
```

In [75]:
```
id(board[0][0]) == id(board[1][1]) == id(board[2][2])
```

Out[75]:
```
True
```

In [78]:
```
R = len(board)

for i in range(len(board)):
    for j in range(len(board[0])):
        print(id(board[i][j]))
```

```
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
140539114800752
```

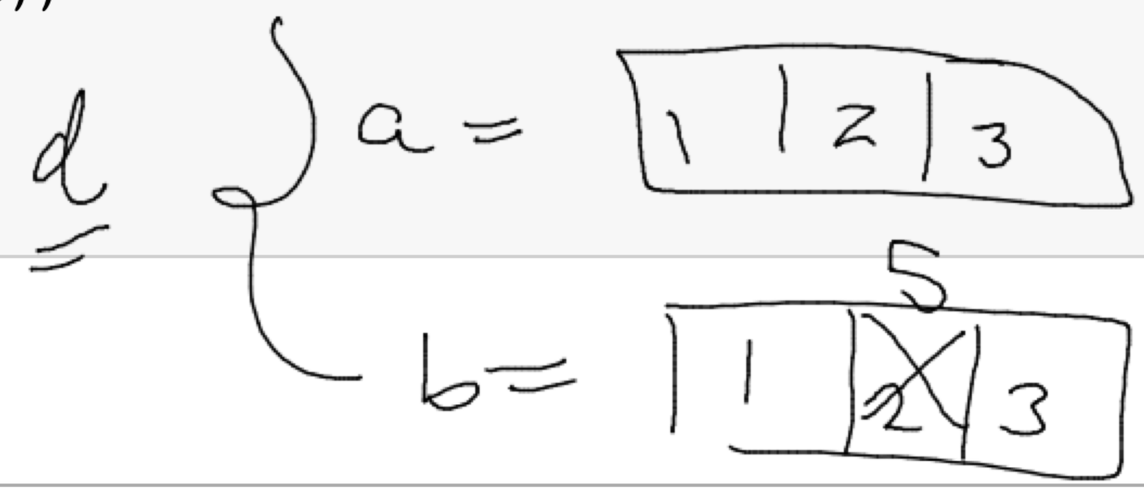


In [80]: `id(board[0]) == id(board[1]) == id(board[2])`

Out[80]: True

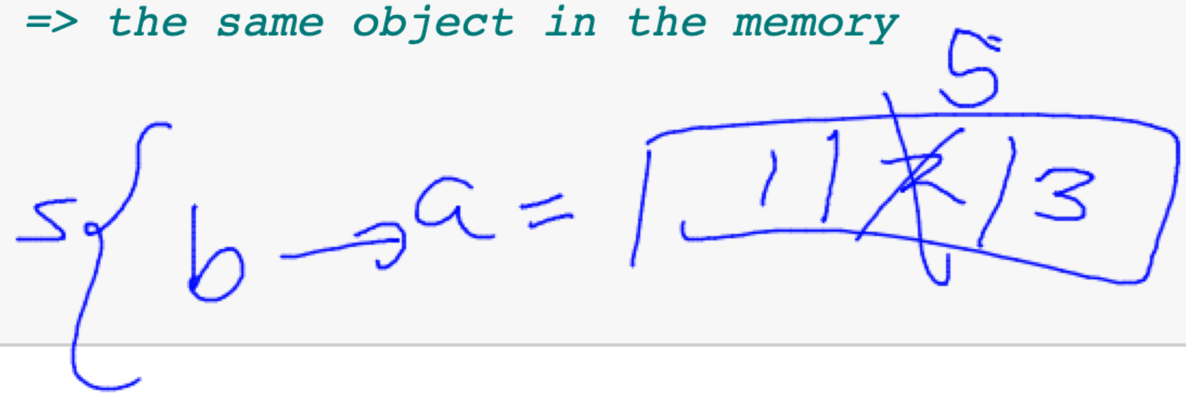
In [275]:
```
a = [1, 2, 3]
b = [1, 2, 3] # deep copy => creating a new list => different object
print(id(a) == id(b))

b[1] = 5
print(a)
print(b)
```

```
False
[1, 2, 3]
[1, 5, 3]
```



In [276]:
```
a = [1, 2, 3]
b = a    # shallow copy => the same object in the memory
print(id(a) == id(b))

b[1] = 5
print(a)
print(b)
```

```
True
[1, 5, 3]
[1, 5, 3]
```



In [ ]:

```
[2, 2, 2]
```

In [308]:
```
l = [[2] * 3] * 3
# [2] * 3 => shallow copy of the list
print(l)
```

```
[[2, 2, 2], [2, 2, 2], [2, 2, 2]]
```

In [309]:
```
l[0][0] = 5
```

In [310]:
```
print(l)
```

```
[[5, 2, 2], [5, 2, 2], [5, 2, 2]]
```

In [300]:
```
id(l[0]) == id(l[1]) == id(l[2])
```

Out[300]: True

In [301]:
```
l = [[2, 2, 2], [2, 2, 2], [2, 2, 2]]
```

In [302]:
```
id(l[0]) == id(l[1]) == id(l[2])
```

Out[302]: False

In [311]:
```
nl=[['','',''],['','',''],['','','']]

for i in range(len(nl)):
```