```
print(board)
```

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

In [39]:
```
print(id(board[0][0]) == id(board[1][1]) == id(board[2][2]))
```

```
True
```

In [40]:
```
tup1 = (1,2)
tup2 = (1,2)

print(id(tup1) == id(tup2))
```

```
False
```

In [41]:
```
l1 = [1, 2, 3]
l2 = [1, 2, 3]
```

In [42]:
```
id(l1) == id(l2)
```

Out[42]:
```
False
```

(1,2) (1,2) } not same

l1 = [1,2,3] l2 = [1,2,3] } not same

Comprehension

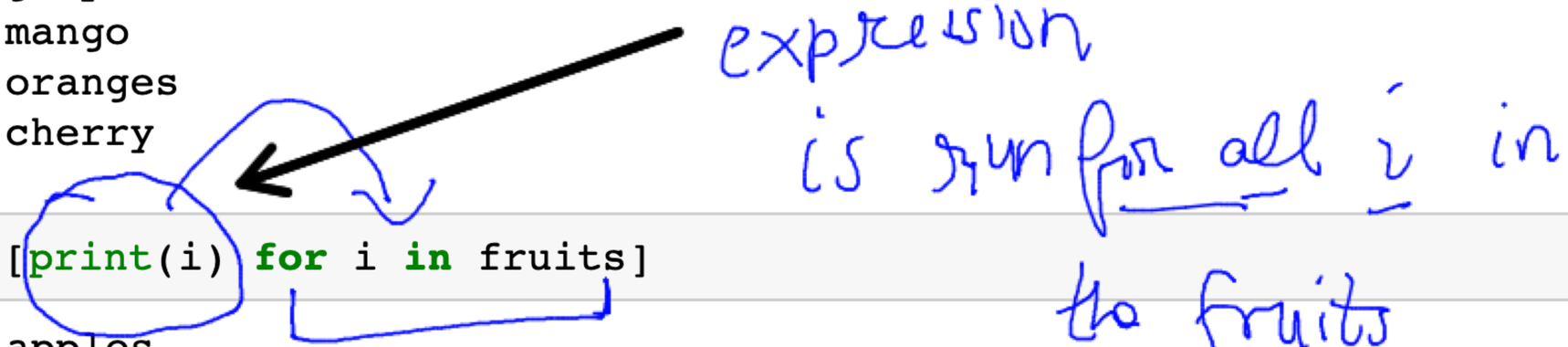
In [ ]:
```
```

Comprehension

In [43]:
```
fruits = ['apples', 'bananas', 'strawberries', 'grapes',
          'mango', 'oranges', 'cherry']
```

In [44]:
```
for i in fruits:
    print(i)
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```



In [45]:
```
[print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

Out[45]:
```
[None, None, None, None, None, None, None]
```


```
grapes
mango
oranges
cherry
```

Out[45]: [None, None, None, None, None, None, None]

In [46]:
```
x = print(5)
```

```
5
```

In [47]:
```
print(x)
```

```
None
```

In [48]:
```
res = [print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

In [49]:
```
print(res)
```

```
[None, None, None, None, None, None, None]
```

In [42]:
```
id(l1) == id(l2)
```

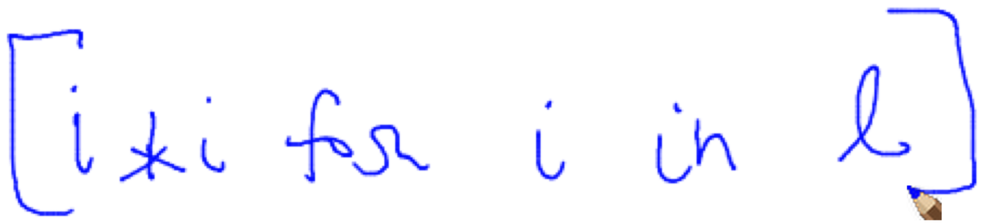
Out[42]: False

Comprehension

In [43]:
```
fruits = ['apples', 'bananas', 'strawberries', 'grapes',
          'mango', 'oranges', 'cherry']
```

In [44]:
```
for i in fruits:
    print(i)
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```



In [45]:
```
[print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
```

File Edit

5 / 6

Python 3 (ipykernel)

In [51]:
```
a = [5, 1, 2, 3, 7]
```

In [53]:
```
b = []

for i in a:
    b.append(i*i)
```

In [54]:
```
print(b)
```

```
[25, 1, 4, 9, 49]
```

In [55]:
```
b_cool = [i**2 for i in a]
```

In [56]:
```
print(b_cool)
```

```
[25, 1, 4, 9, 49]
```

In [57]:
```
c = [i for i in a]
```

In [58]:
```
print(c)
```

```
[5, 1, 2, 3, 7]
```

In [ ]:

In [ ]:

In [57]:
```
c = [i for i in a]
```

In [58]:
```
print(c)
```

```
[5, 1, 2, 3, 7]
```

Quiz

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

l = [0 for i in a]
```

l = [0, 0, 0]

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [70]:
```
a = [5, 1, 2, 3, 7, 6]

b = []

for i in a:
    if i % 2 == 0: # divisible by 2 => even
        b.append(i*i)
    else:
        b.append(i)

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

b d d → i

E: von → i²

In [71]:
```
def my_logic(i):
    if i % 2 == 0:
        return i*i
    else:
        return i
```

In [73]:
```
my_logic(5)
```

Out[73]: 5

In [ ]:

In [ ]:

In [71]:
```
def my_logic(i):
    if i % 2 == 0:
        return i*i
    else:
        return i
```

In [73]:
```
my_logic(5)
```

Out[73]: 5

In [75]:
```
b = []

for i in a:
    b.append(my_logic(i))

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [ ]:
```
b = [      for i in a]
```

In [ ]:

In [ ]:

In [ ]:

In [71]:
```
def my_logic(i):
    if i % 2 == 0:
        return i*i
    else:
        return i
```

In [73]:
```
my_logic(5)
```

Out[73]: 5

In [75]:
```
b = []

for i in a:
    b.append(my_logic(i))

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [78]:
```
b = [my_logic(i) for i in a]

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [ ]:

In [ ]:

In [71]:
```
def my_logic(i):
    if i % 2 == 0:
        return i*i
    else:
        return i
```

In [73]:
```
my_logic(5)
```

Out[73]: 5

In [75]:
```
b = []

for i in a:
    b.append(my_logic(i))

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [78]:
```
b = [my_logic(i) for i in a]

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [ ]:

In [ ]:

```
        else:
            return i
```

In [73]:
```
my_logic(5)
```

Out[73]: 5

time(A) ≈ time(B)

In [75]:
```
b = []

for i in a:
    b.append(my_logic(i))

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

A

append

Arrays

In [78]:
```
b = [my_logic(i) for i in a]

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

B

In [ ]:

In [ ]:

In [ ]:

```
print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [79]:
```
b = [i*i if i % 2 == 0 else i for i in a]

print(b)
```

```
[5, 1, 4, 3, 7, 36]
```

In [80]:
```
i = 5

if i == 5:
    print('True')
else:
    print('False')
```

```
True
```

T if cond. else F

True value

False value.

In [82]:
```
i = 5
'True' if i == 5 else 'False'
```

Out[82]:
```
'True'
```

In [ ]:

In [ ]:

In [82]:
```
i = 5
'True' if i == 5 else 'False'
```

Out[82]: 'True'

In [83]:
```
def my_print(i):
    print(i)
    return None
```

In [84]:
```
res = [my_print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

expression is run & return value added to the list!

In [85]:
```
print(res)
```

```
[None, None, None, None, None, None, None]
```

In [ ]:

In [ ]:

In [ ]:

In [82]:
```
i = 5
'True' if i == 5 else 'False'
```

Out[82]:
```
'True'
```

In [83]:
```
def my_print(i):
    print(i)
    return None
```

In [84]:
```
res = [my_print(i) for i in fruits]
```

```
apples
bananas
strawberries
grapes
mango
oranges
cherry
```

In [85]:
```
print(res)
```

```
[None, None, None, None, None, None, None]
```

In [ ]:

In [ ]:

In [ ]:

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

```
Rahul
```

In [94]:
```
bits = [False, True, False, False, True, False, False, True]
bits = [1 if b==True else 0 for b in bits]
```

In [95]:
```
print(bits)
```

```
[0, 1, 0, 0, 1, 0, 0, 1]
```

In [ ]:

In [ ]:

In [ ]:

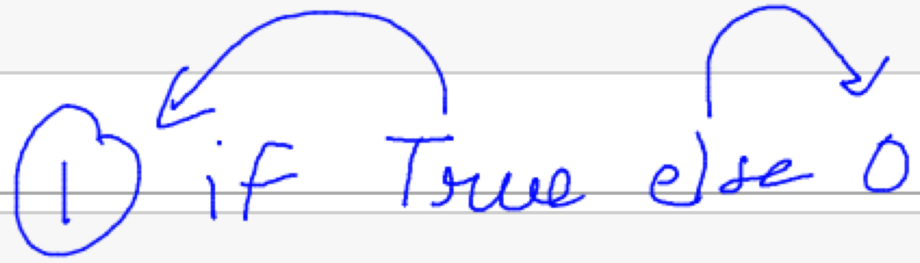
In [ ]:

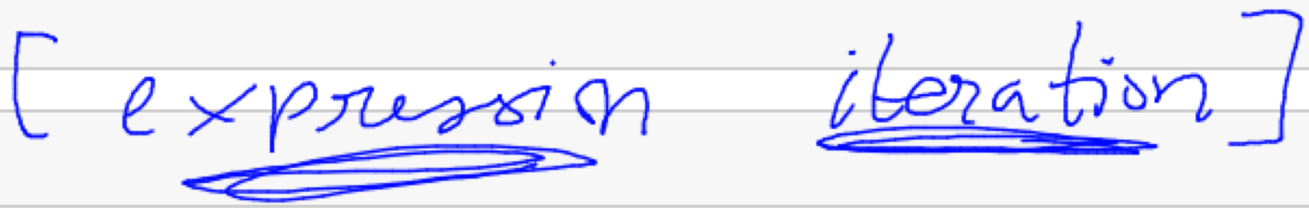
In [ ]:

In [ ]:

In [ ]:

In [ ]:





In [94]:
```
bits = [False, True, False, False, True, False, False, True]
bits = [1 if b==True else 0 for b in bits]
```

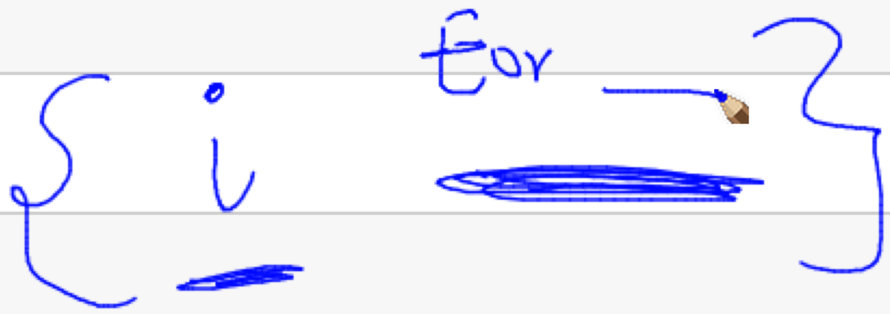
In [95]:
```
print(bits)
```

```
[0, 1, 0, 0, 1, 0, 0, 1]
```

More fun on comprehension

In [97]:
```
l = [i for i in range(1, 5)]
print(l)
```

```
[1, 2, 3, 4]
```



In [99]:
```
s = {i for i in range(1, 10)}
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
<generator object <genexpr> at 0x7fc2b90727b0>
```

In [107]:
```
tuple(t)
```

Out[107]:
```
(1, 2, 3, 4)
```

In [109]:
```
d = { i : i**2 for i in range(1, 5)}
print(d)
```

Exp => key val

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

In [110]:
```
tuple([1, 2, 3, 4])
```

Out[110]:
```
(1, 2, 3, 4)
```

In [112]:
```
d = { i : 5 for i in range(1, 3)}
print(d)
```

i => 1, 2

1 : 5

2 : 5

```
{1: 5, 2: 5}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
<generator object <genexpr> at 0x7fc2b90727b0>
```

In [107]:
```
tuple(t)
```

Out[107]:
```
(1, 2, 3, 4)
```

In [109]:
```
d = { i : i**2 for i in range(1, 5)}
print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

In [110]:
```
tuple([1, 2, 3, 4])
```

Out[110]:
```
(1, 2, 3, 4)
```

In [112]:
```
d = { i : 5 for i in range(1, 3)}
print(d)
```

```
{1: 5, 2: 5}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [109]:
```
d = { i : i**2 for i in range(1, 5)}
print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

In [110]:
```
tuple([1, 2, 3, 4])
```

Out[110]:
```
(1, 2, 3, 4)
```

In [112]:
```
d = { i : 5 for i in range(1, 3)}
print(d)
```

```
{1: 5, 2: 5}
```

In [114]:
```
d = { 5 : i for i in range(1, 3)}
print(d)
```

```
{5: 2}
```

In [115]:
```
d = { 5 : i for i in range(10, 2, -1)}
print(d)
```

```
{5: 3}
```

In [ ]:

In [ ]:

In [ ]:

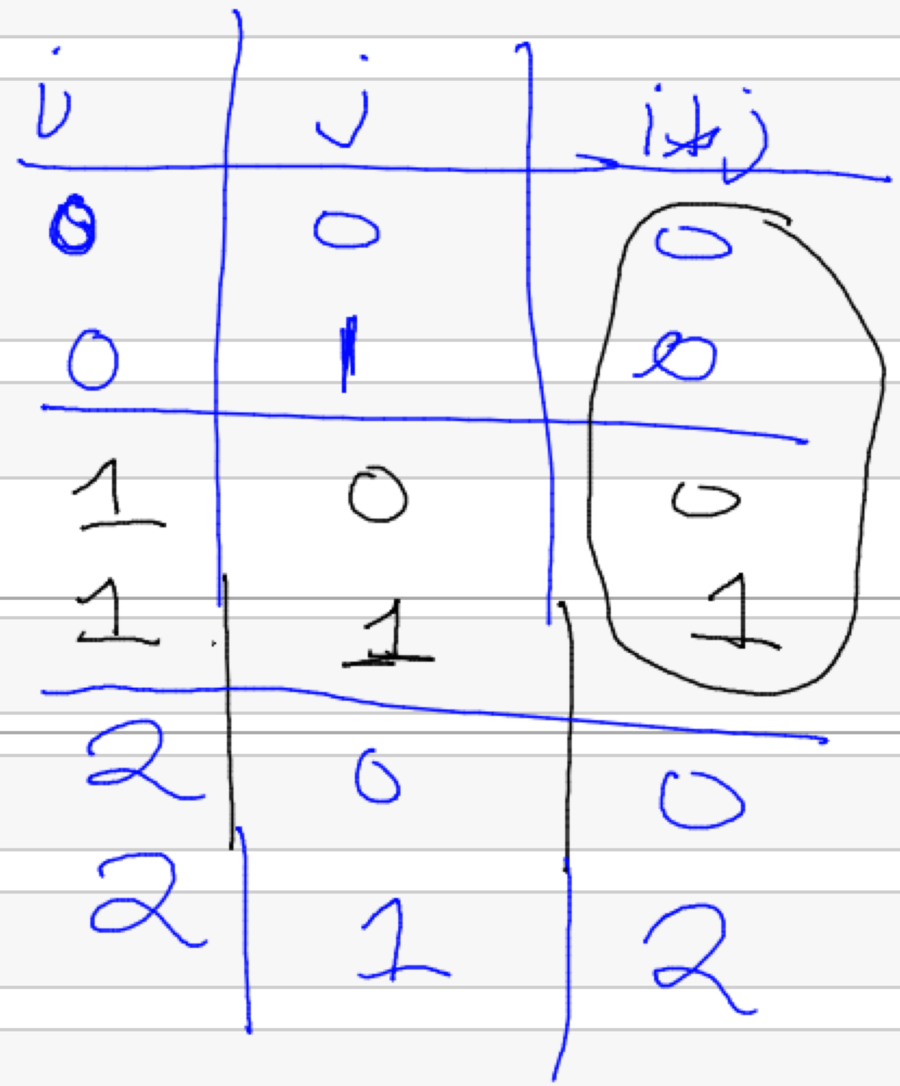
```
print(d)
```

```
{5: 3}
```

Fun part of list comprehension

In [ ]:

In [116]:
```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
```



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

```
[0, 0, 0, 1, 0, 2]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

utor: Visuali a

Python 3.6
(known limitations)

```
1 l = []
2 for i in range(3):
3     for j in range(2):
4         l.append(i * j)
```

Edit this code

st executed
execute

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

Done running (20 steps)

ization

Frames

Global frame

l	
i	2
j	1

Objects

list

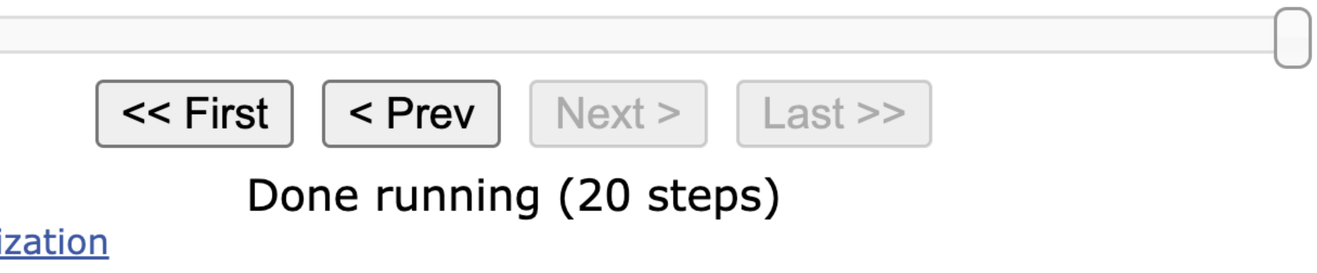
0	1	2	3	4	5
0	0	0	1	0	2

Python 3.6
(known limitations)

```
1  l = []
2  for i in range(3):
3      for j in range(2):
4          l.append(i * j)
```

Edit this code

st executed
execute



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

Done running (20 steps)

ization

Frames

Global frame

l

i 2

j 1

Objects

list

0	1	2	3	4	5
0	0	0	1	0	2

```
for i in range(3):
    for j in range(2):
        l.append(i * j)
```

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

```
[0, 0, 0, 1, 0, 2]
```

In [120]:
```
res = [i for i in range(3)]

print(res)
```

```
[0, 1, 2]
```

In [121]:
```
res2 = [j for j in range(2)]

print(res2)
```

```
[0, 1]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [116]:
```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
```

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

```
[0, 0, 0, 1, 0, 2]
```

In [120]:
```
res = [i for i in range(3)]

print(res)
```

```
[0, 1, 2]
```

In [121]:
```
res2 = [j for j in range(2)]
print(res2)
```

```
[0, 1]
```

In [128]:
```
cool_res = [i*j for j in range(2) for i in range(3)]

print(cool_res)
```

```
[0, 0, 0, 0, 1, 2]
```

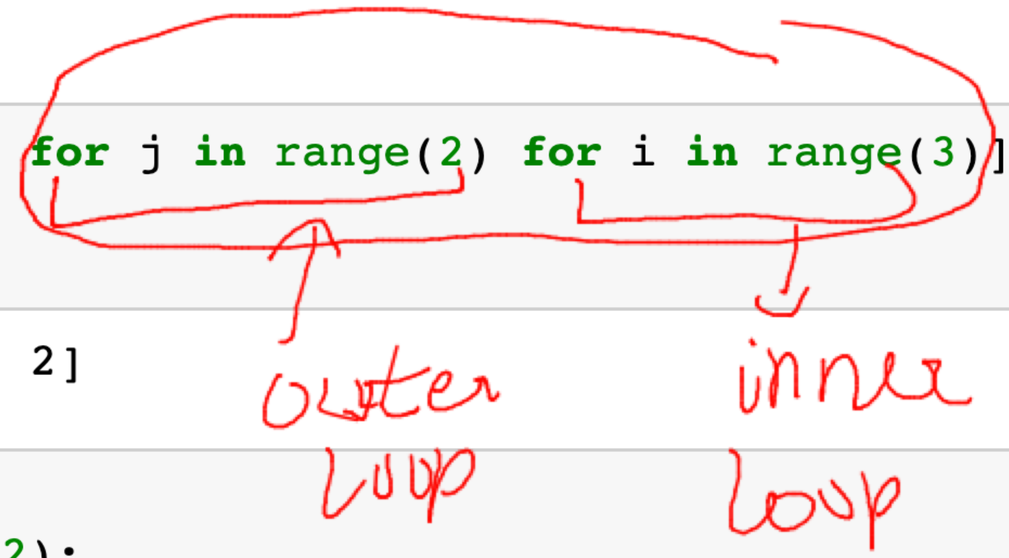
In [ ]:

In [121]:
```
res2 = [j for j in range(2)]
print(res2)
```

```
[0, 1]
```

In [128]:
```
cool_res = [i*j for j in range(2) for i in range(3)]

print(cool_res)
```

```
[0, 0, 0, 0, 1, 2]
```



In [130]:
```
res = []
for j in range(2):
    for i in range(3):
        res.append(i*j)

print(res)
```

```
[0, 0, 0, 0, 1, 2]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [121]:
```
res2 = [j for j in range(2)]
print(res2)
```

```
[0, 1]
```

In [128]:
```
cool_res = [i*j for j in range(2) for i in range(3)]

print(cool_res)
```

```
[0, 0, 0, 0, 1, 2]
```

In [130]:
```
res = []
for j in range(2):
    for i in range(3):
        res.append(i*j)

print(res)
```

```
[0, 0, 0, 0, 1, 2]
```

In [ ]:

In [ ]:

In [ ]:

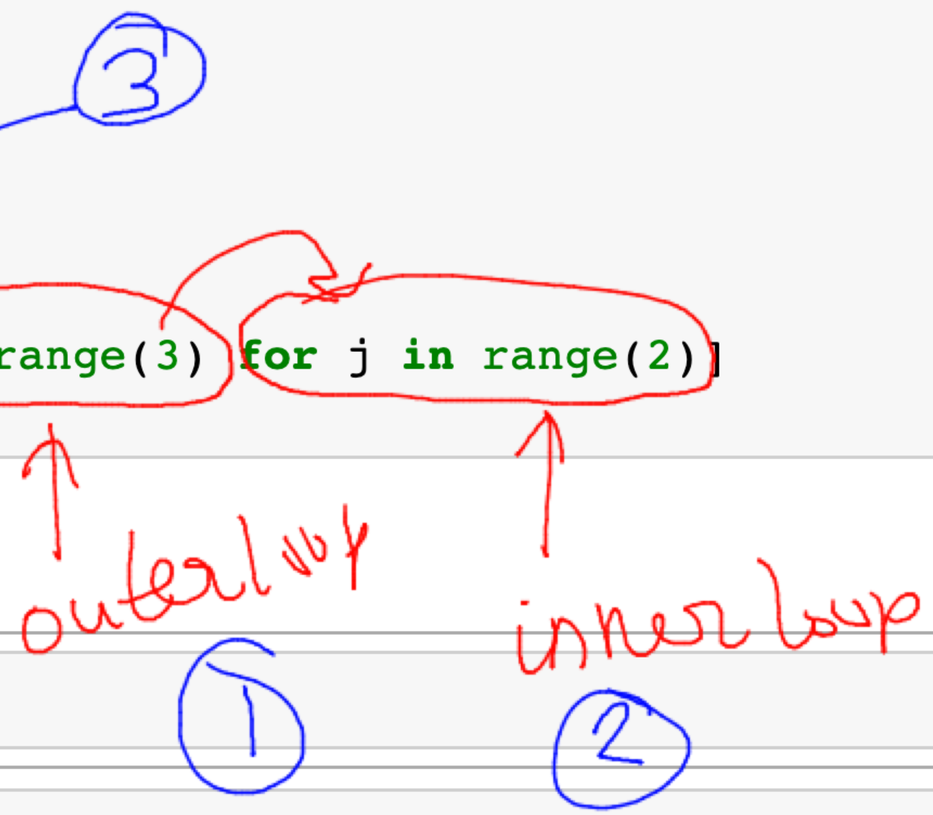
In [ ]:

In [130]:
```
res = []
for j in range(2):
    for i in range(3):
        res.append(i*j)

print(res)
```

```
[0, 0, 0, 0, 1, 2]
```

In [132]:
```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
print(l)

res_cool = [i*j for i in range(3) for j in range(2)]
print(res_cool)
```

```
[0, 0, 0, 1, 0, 2]
[0, 0, 0, 1, 0, 2]
```



In [ ]:

In [ ]:

In [ ]:

In [134]:
```
a = []
for i in range(3):
    b = []
    for j in range(2):
        b.append(i * j)
    a.append(b)

print(a)
```

i => 0/1/2

```
[[0, 0], [0, 1], [0, 2]]
```

List comprehension to get inner list?

In [135]:
```
[0*j for j in range(2)]
```

Out[135]:
```
[0, 0]
```

In [136]:
```
[1*j for j in range(2)]
```

Out[136]:
```
[0, 1]
```

Can we nest it?

In [137]:
```
[2*j for j in range(2)]
```

Out[137]:
```
[0, 2]
```

In [ ]:

In [ ]:

In [ ]:


```
    a.append(b)

print(a)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [135]:
```
[0*j for j in range(2)]
```

Out[135]:
```
[0, 0]
```

In [136]:
```
[1*j for j in range(2)]
```

Out[136]:
```
[0, 1]
```

In [137]:
```
[2*j for j in range(2)]
```

Out[137]:
```
[0, 2]
```

In [138]:
```
super_cool = [[i*j for j in range(2)] for i in range(3)]
```

In [139]:
```
print(super_cool)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [ ]:

In [ ]:

Inner loop

outer loop

```
    a.append(b)

print(a)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [135]:
```
[0*j for j in range(2)]
```

Out[135]:
```
[0, 0]
```

In [136]:
```
[1*j for j in range(2)]
```

Out[136]:
```
[0, 1]
```

In [137]:
```
[2*j for j in range(2)]
```

Out[137]:
```
[0, 2]
```

In [138]:
```
super_cool = [[i*j for j in range(2)] for i in range(3)]
```

In [139]:
```
print(super_cool)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [ ]:

In [ ]:

```
    b = []
    for j in range(2):
        b.append(i * j)
    a.append(b)

print(a)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [135]:
```
[0*j for j in range(2)]
```

Out[135]:
```
[0, 0]
```

In [136]:
```
[1*j for j in range(2)]
```

Out[136]:
```
[0, 1]
```

In [137]:
```
[2*j for j in range(2)]
```

Out[137]:
```
[0, 2]
```

In [138]:
```
super_cool = [[i*j for j in range(2)] for i in range(3)]
```

In [139]:
```
print(super_cool)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [ ]:

```
    b = []
    for j in range(2):
        b.append(i * j)
    a.append(b)

print(a)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [135]:
```
[0*j for j in range(2)]
```

Out[135]:
```
[0, 0]
```

In [136]:
```
[1*j for j in range(2)]
```

Out[136]:
```
[0, 1]
```

In [137]:
```
[2*j for j in range(2)]
```

Out[137]:
```
[0, 2]
```

In [138]:
```
super_cool = [[i*j for j in range(2)] for i in range(3)]
```

In [139]:
```
print(super_cool)
```

```
[[0, 0], [0, 1], [0, 2]]
```

In [ ]:

[0, 0, 0, 0, 1, 2]

In [130]:
```
res = []
for j in range(2):
    for i in range(3):
        res.append(i*j)

print(res)
```

```
[0, 0, 0, 0, 1, 2]
```

In [141]:
```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
print(l)

# single comprehension
res_cool = [i*j for i in range(3) for j in range(2)]
print(res_cool)
```



```
[0, 0, 0, 1, 0, 2]
[0, 0, 0, 1, 0, 2]
```

2d list

In [134]:
```
a = []
for i in range(3):
    b = []
```


```
[0, 0, 0, 0, 1, 2]
```

In [130]:
```
res = []
for j in range(2):
    for i in range(3):
        res.append(i*j)

print(res)
```

```
[0, 0, 0, 0, 1, 2]
```

In [141]:
```
l = []
for i in range(3):
    for j in range(2):
        l.append(i * j)
print(l)

# single comprehension
res_cool = [i*j for i in range(3) for j in range(2)]
print(res_cool)
```

```
[0, 0, 0, 1, 0, 2]
[0, 0, 0, 1, 0, 2]
```

2d list

In [134]:
```
a = []
for i in range(3):
    b = []
```

Out[136]: [0, 1]

In [137]:
```
[2*j for j in range(2)]
```

Out[137]: [0, 2]

In [140]:
```
# comprehension within comprehension

super_cool = [[i*j for j in range(2)] for i in range(3)]
```

In [139]:
```
print(super_cool)
```

[[0, 0], [0, 1], [0, 2]]

In [ ]:

In [ ]:

In [ ]:

In [ ]:

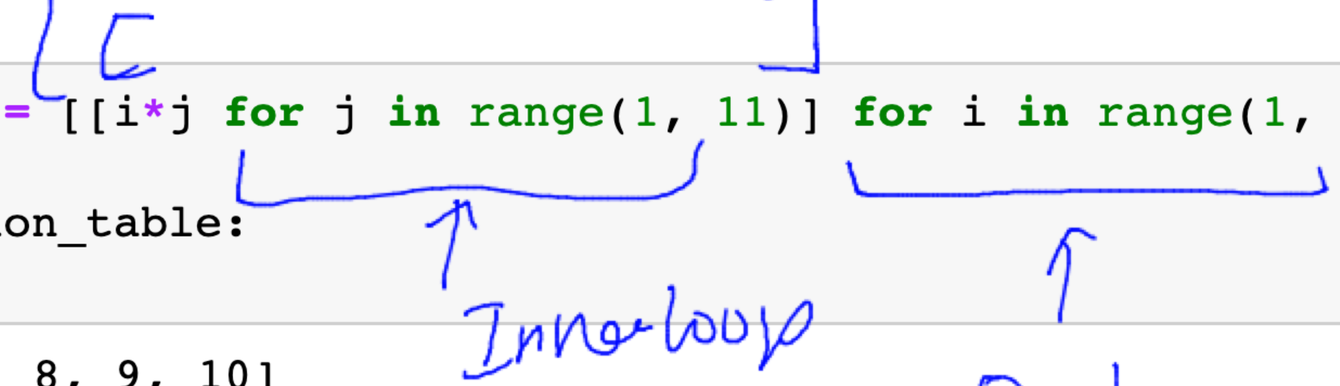
In [ ]:

In [ ]:

```
    c = []
    for j in range(1, 11):
        c.append(i * j)
    res.append(c)

for i in res:
    print(i)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40]
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
[6, 12, 18, 24, 30, 36, 42, 48, 54, 60]
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
```

In [149]:
```
multiplication_table = [[i*j for j in range(1, 11)] for i in range(1, 8)]

for i in multiplication_table:
    print(i)
```




```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40]
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
[6, 12, 18, 24, 30, 36, 42, 48, 54, 60]
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
```

In [ ]:

```
{'name': 'Numan', 'age': 5000, 'greatest_avenger': True, 'weapons': ['mjo
lnir', 'stormbreaker', 'good looks']}
```

In [160]:
```
person['weapons'].pop()
```

Out[160]:
```
'good looks'
```

In [161]:
```
print(person['weapons'])
```

```
['mjolnir', 'stormbreaker']
```

l = [1, 2, 3]

l.pop()

In [162]:
```
# update many keys or add new keys

person.update({
    'age': 100000,
    'weapons': ['stormbreaker'],
    'radom_info': ("random", "info")
})
```

In [163]:
```
print(person)
```

```
{'name': 'Numan', 'age': 100000, 'greatest_avenger': True, 'weapons': ['s
tormbreaker'], 'radom_info': ('random', 'info')}
```

In [ ]:

In [ ]:

```
{'name': 'Numan', 'age': 5000, 'greatest_avenger': True, 'weapons': ['mjo
lnir', 'stormbreaker', 'good looks']}
```

In [160]:
```
person['weapons'].pop()
```

Out[160]:
```
'good looks'
```

In [161]:
```
print(person['weapons'])
```

```
['mjolnir', 'stormbreaker']
```

In [162]:
```
# update many keys or add new keys
person.update({
    'age': 100000,
    'weapons': ['stormbreaker'],
    'radom_info': ("random", "info")
})
```

In [163]:
```
print(person)
```

```
{'name': 'Numan', 'age': 100000, 'greatest_avenger': True, 'weapons': ['s
tormbreaker'], 'radom_info': ('random', 'info')}
```

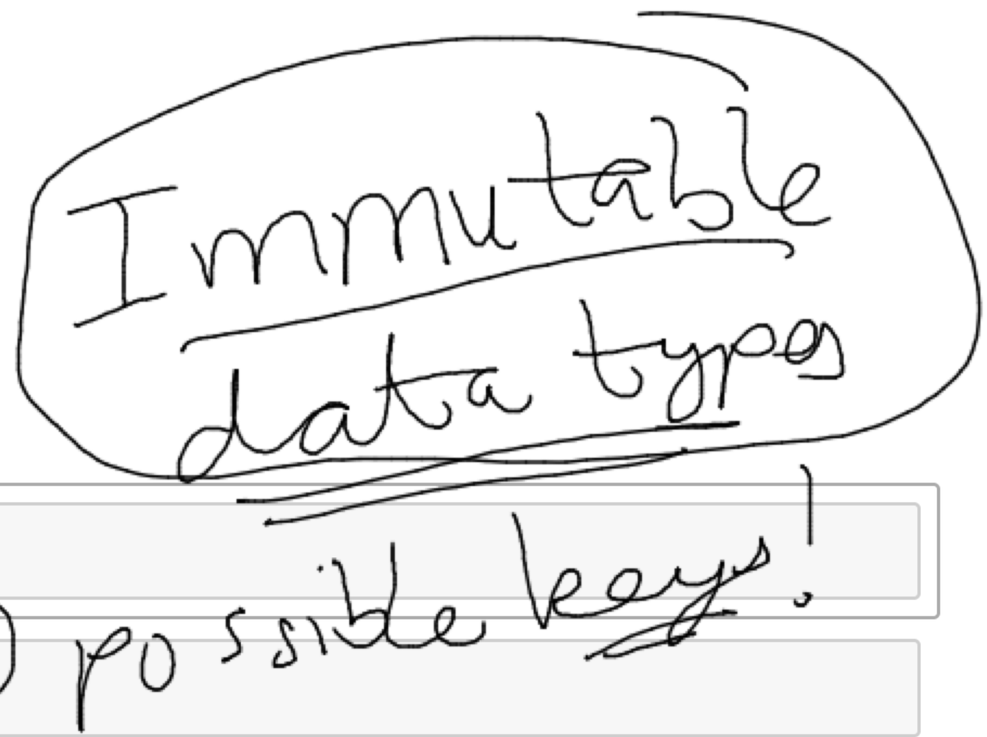
In [ ]:

In [ ]:

```
KeyError: 0
```

In [168]:
```
person.update({
    [1, 2, 3]: "list"
})
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [168], in <cell line: 1>()
----> 1 person.update({
      2     [1, 2, 3]: "list"
      3 })

TypeError: unhashable type: 'list'
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

1. **Before implementing the code =>**

- try on paper,
- take examples,
- think the logic,
- plan the solution

2. **Write the code, test it (dry run on an example) and submit it**

3. **You will get errors, how to get past those hurdles?**

Before reaching out to TAs:

- If the problem has a video solution or a hint 1 / hint 2 => try to use it
- If it's a syntax error => try to google, stackoverflow => finding / debugging code is a process which develops over time

Then still if not resolved, reach out to TAs / peers in slack / whatsapp

4. **Say you finally solved problem with helps of peers / TAs what next?**

- Don't jump to the next problem yet!!!!
- Checkout the hints (they are now free - no score deduction)
- Maybe your post solution on slack / whatsapp group and get feedback from peers
- reach out TAs to know how can you have made your own solution much better

- try on paper,
- take examples,
- think the logic,
- plan the solution

2. **Write the code, test it (dry run on an example) and submit it**

3. **You will get errors, how to get past those hurdles?**

Before reaching out to TAs:

- If the problem has a video solution or a hint 1 / hint 2 => try to use it
- If it's a syntax error => try to google, stackoverflow => finding / debugging code is a process which develops over time

Then still if not resolved, reach out to TAs / peers in slack / whatsapp

4. **Say you finally solved problem with helps of peers / TAs what next?**

- Don't jump to the next problem yet!!!!
- Checkout the hints (they are now free - no score deduction)
- Maybe your post solution on slack / whatsapp group and get feedback from peers
- reach out TAs to know how can you have made your own solution much better

- try on paper,
- take examples,
- think the logic,
- plan the solution

=> pseudocode

2. **Write the code, test it (dry run on an example) and submit it**

3. **You will get errors, how to get past those hurdles?**

Before reaching out to TAs:

- If the problem has a video solution or a hint 1 / hint 2 => try to use it
- If it's a syntax error => try to google, stackoverflow => finding / debugging code is a process which develops over time

Then still if not resolved, reach out to TAs / peers in slack / whatsapp

4. **Say you finally solved problem with helps of peers / TAs what next?**

- Don't jump to the next problem yet!!!!
- Checkout the hints (they are now free - no score deduction)
- Maybe your post solution on slack / whatsapp group and get feedback from peers
- reach out TAs to know how can you have made your own solution much better

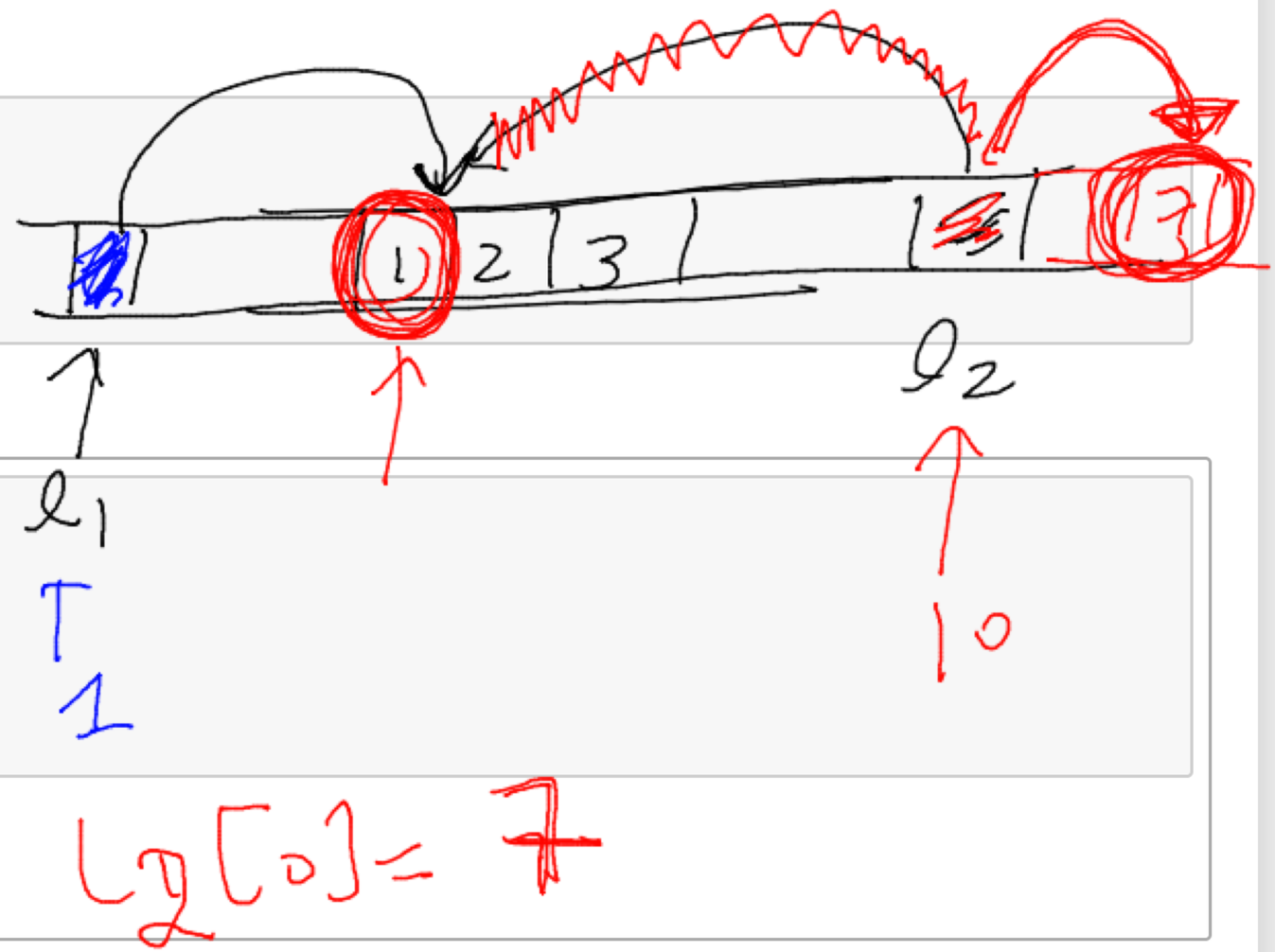
[[, ,], [, ,], [, ,]]

In [39]:
```
print(id(board[0][0]) == id(board[1][1]) == id(board[2][2]))
```

True

In [40]:
```
tup1 = (1,2)
tup2 = (1,2)

print(id(tup1) == id(tup2))
```

False



In [224]:
```
l1 = [1, 2, 3]
l2 = [1, 2, 3]

print(id(l1[0]))
print(id(l2[0]))
```

```
140474019014960
140474019014960
```

In [222]:
```
id(l1) == id(l2)
```

Out[222]: False

Comprehension

In [245]:
```
list_3d = [[[i+j+k for k in range(3)] for j in range(3)] for i in range(3)]
```

In [246]:
```
print(list_3d)
```

```
[[[0, 1, 2], [1, 2, 3], [2, 3, 4]], [[1, 2, 3], [2, 3, 4], [3, 4, 5]],
[[2, 3, 4], [3, 4, 5], [4, 5, 6]]]
```

In [247]:
```
fruits = ["apples", "bananas", "strawberries"]

res = [fruit.upper() for fruit in fruits]
```

In [248]:
```
print(fruits)
```

```
['apples', 'bananas', 'strawberries']
```

In [249]:
```
print(res)
```

```
['APPLES', 'BANANAS', 'STRAWBERRIES']
```

In [250]:
```
fruits = res
```

In [251]:
```
print(fruits)
```

```
['APPLES', 'BANANAS', 'STRAWBERRIES']
```

In [ ]:

Python 3.6
(known limitations)

```
1  fruits = ["apples", "bananas", "strawberries"]
2
3  res = [fruit.upper() for fruit in fruits]
```

Edit this code

e that just executed
xt line to execute

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

Step 5 of 8

nize visualization

Frames Objects

Global frame
fruits

list
0 "apples" | 1 "bananas" | 2 "strawberries"

<listcomp>
.0
fruit "apples"

list_iterator instance