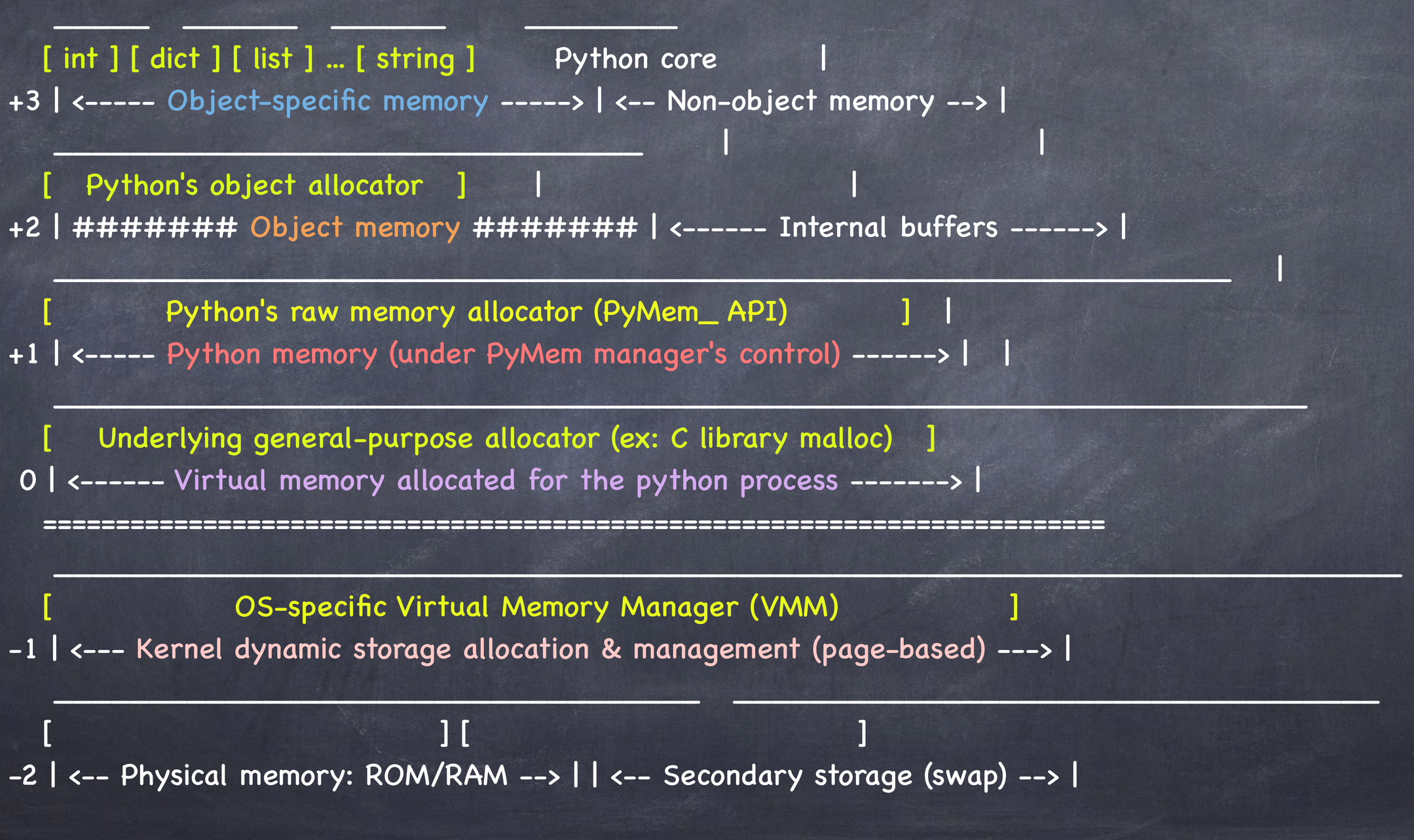
Python高级内存管理

- xiaorui.cc

Object-specific allocators



```
* Request in bytes     Size of allocated block     Size class idx
* ----------------------------------------------------------------
*        1-8                     8                       0
*        9-16                   16                       1
*       17-24                   24                       2
*       25-32                   32                       3
*       33-40                   40                       4
*       41-48                   48                       5
*       49-56                   56                       6
*       57-64                   64                       7
*         ...                   ...                    ...
*      497-504                  504                      62
*      505-512                  512                      63
*
*     小于SMALL_REQUEST_THRESHOLD 从PyObject_Malloc，大于退化到malloc行为
*/
```

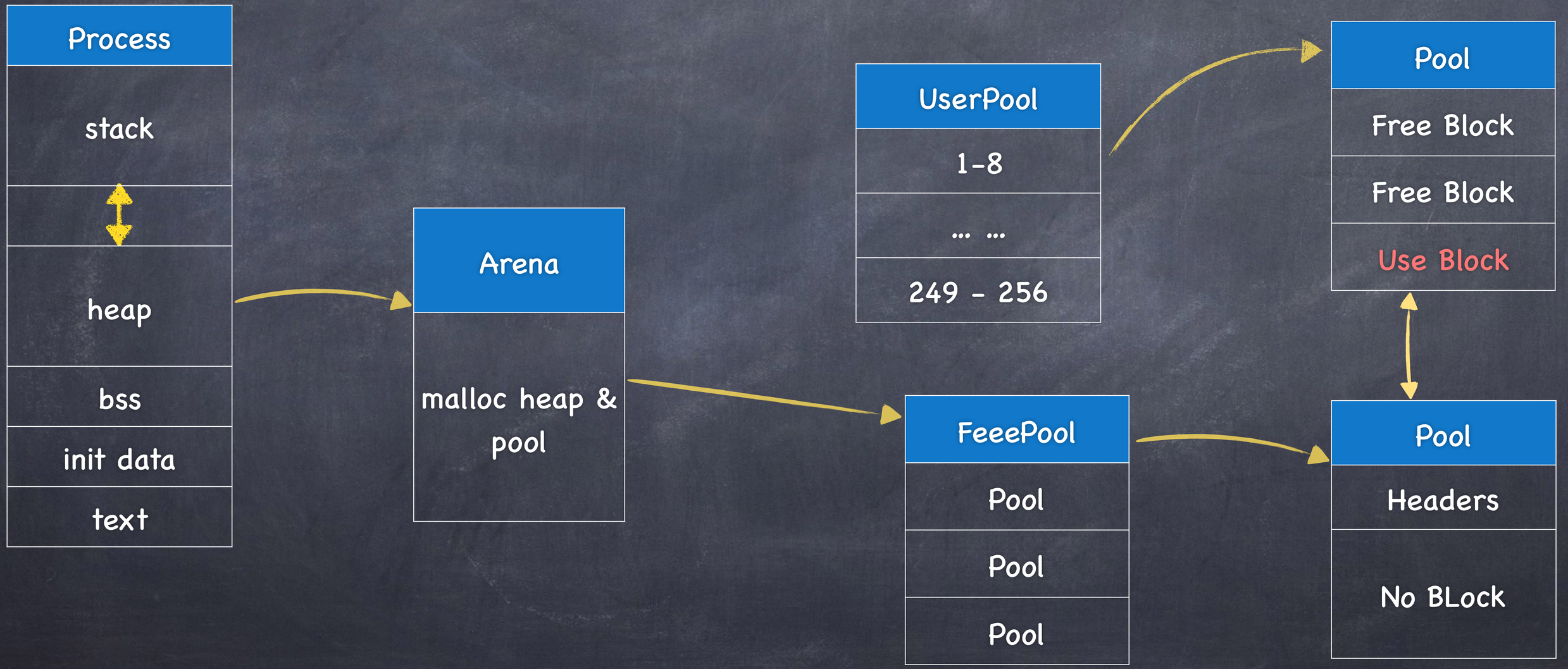
名词解释

- process heap
- Arenas
- Pool
- UsedPools
- FreePools

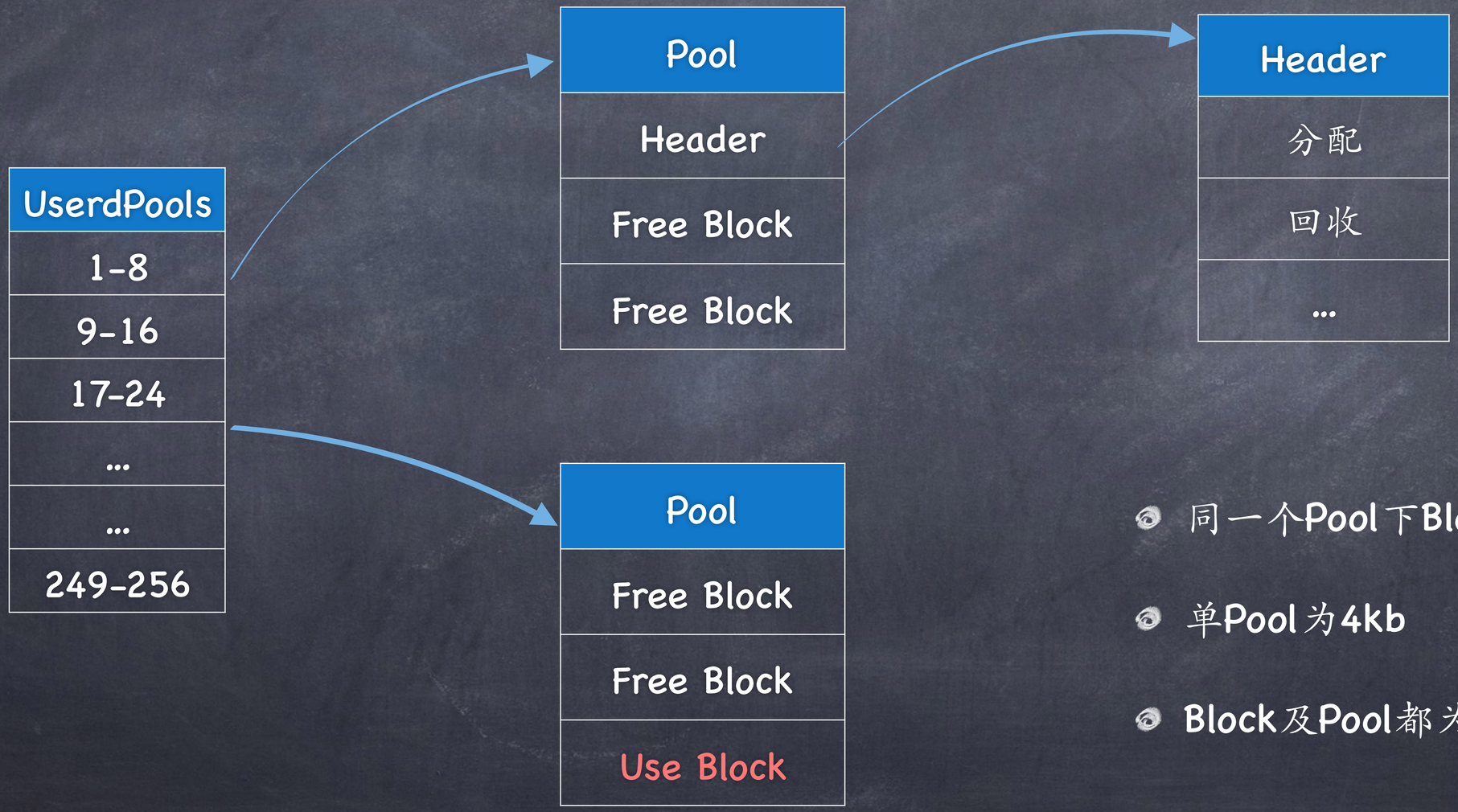
method

- posix malloc
- python memory pool
- object buffer pool

Arena

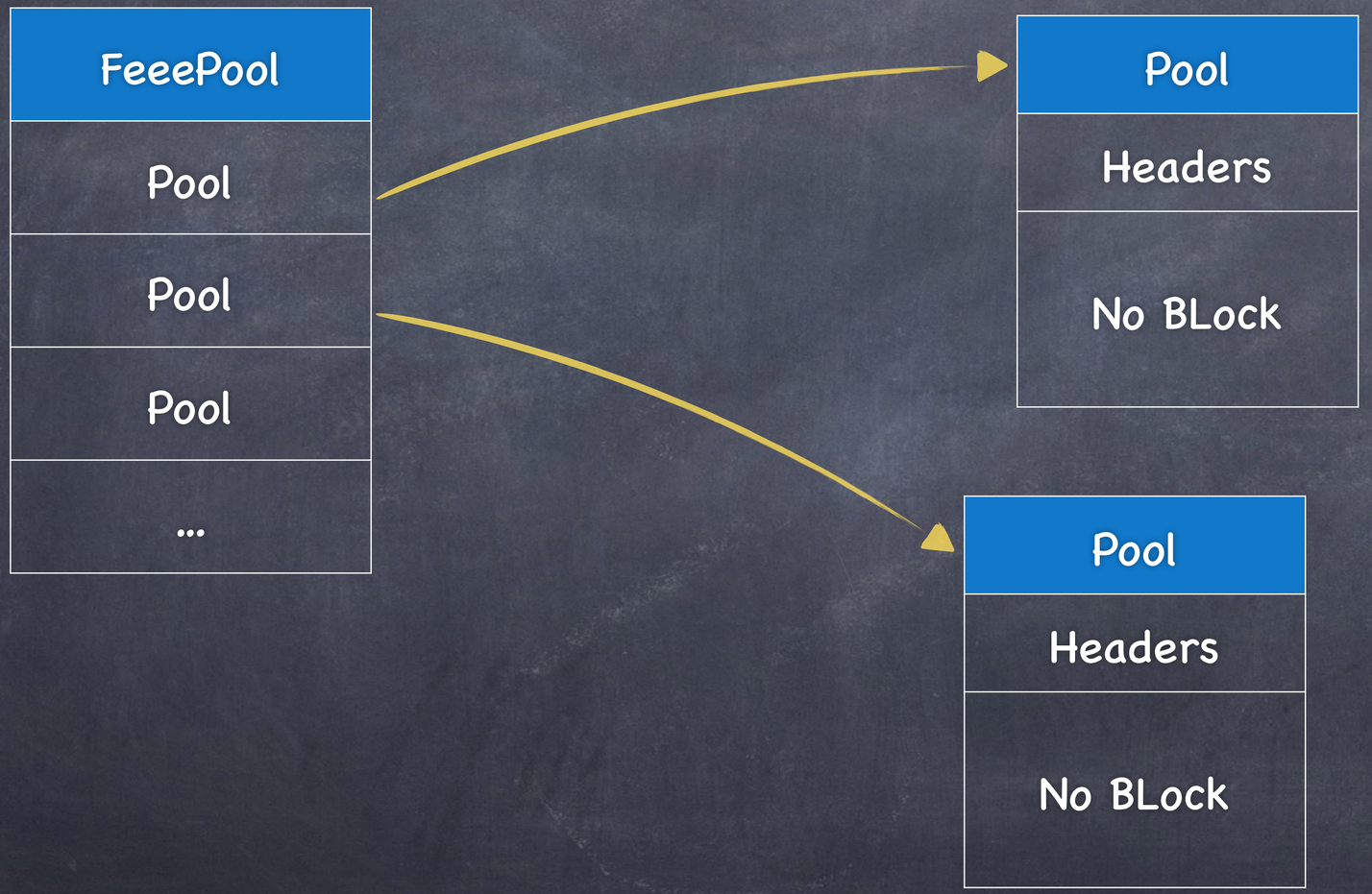

Process
stack
heap
bss
init data
text
Arena
malloc heap & pool
UserPool
1-8
... ...
249 - 256
FeeePool
Pool
Pool
Pool
Pool
Free Block
Free Block
Use Block
Pool
Headers
No BLock

userdpool design



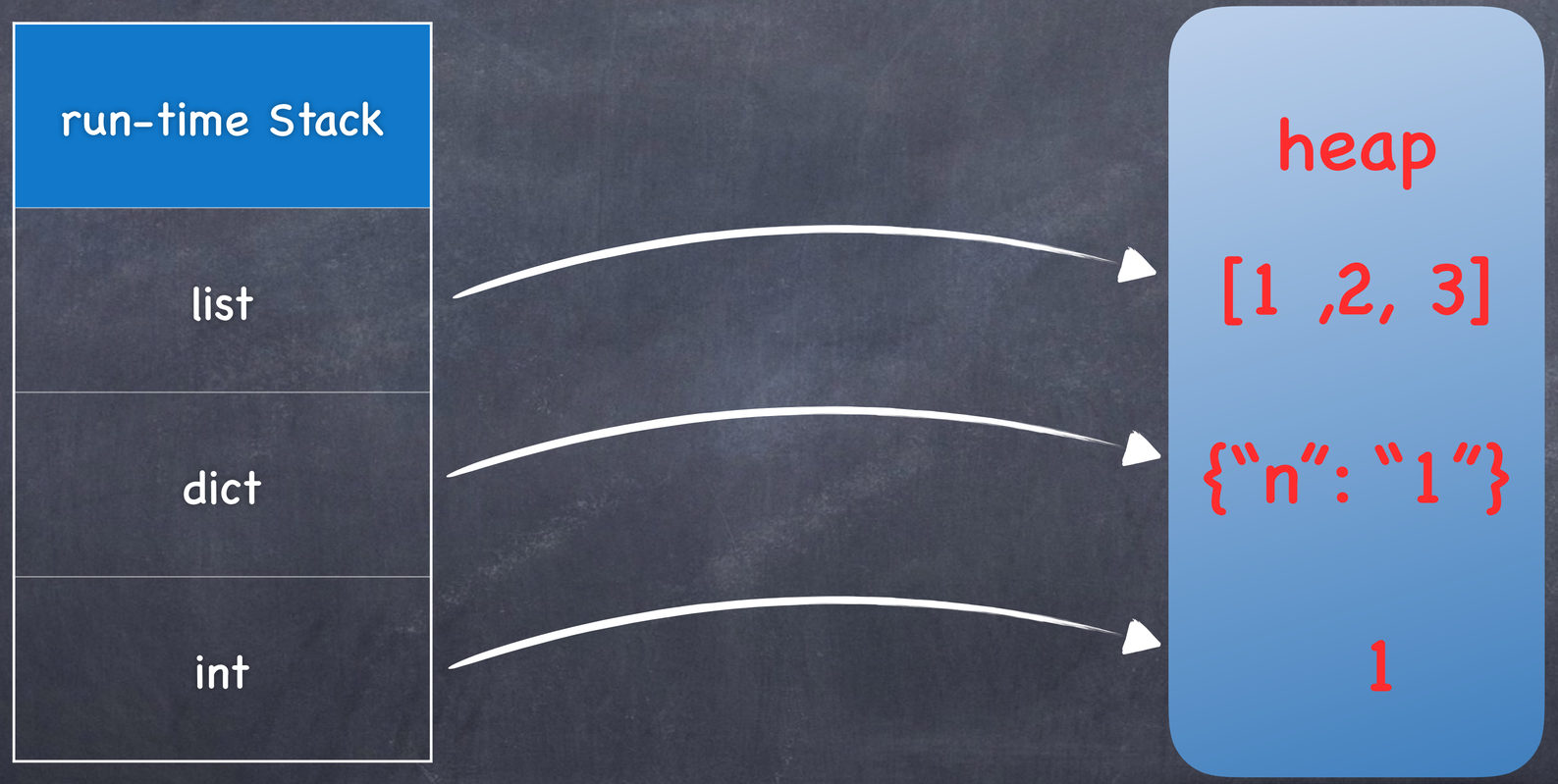
- 同一个Pool下Block一样长
- 单Pool为4kb
- Block及Pool都为单链表

free pool desgin



- Pool为4kb大小
- Pool清理Headers

where store variable ?



why ?

```
In [1]: a = 123

In [2]: b = 123

In [3]: a is b
Out[3]: True
```

```
In [4]: a = 1000

In [5]: b = 1000

In [6]: a is b
Out[6]: False
```

```
In [7]: a = 'n'

In [8]: b = 'n'

In [9]: a is b
Out[9]: True
```

```
In [10]: a = "python"

In [11]: b = "python"

In [12]: a is b
Out[12]: True
```

why ?

```
In [10]: a = b = 'nima'

In [11]: b = a

In [12]: a is b
Out[12]: True

In [13]: b = 'hehe'

In [14]: a is b
Out[14]: False
```

只有引用 ?

```
In [1]: def go(var):
   ...:     print id(var)
   ...:

In [2]: id(a)
Out[2]: 4401335072

In [3]: go(a)
4401335072
```

python objects stored in memory?

Python Has Names, Not Variables ! ! !

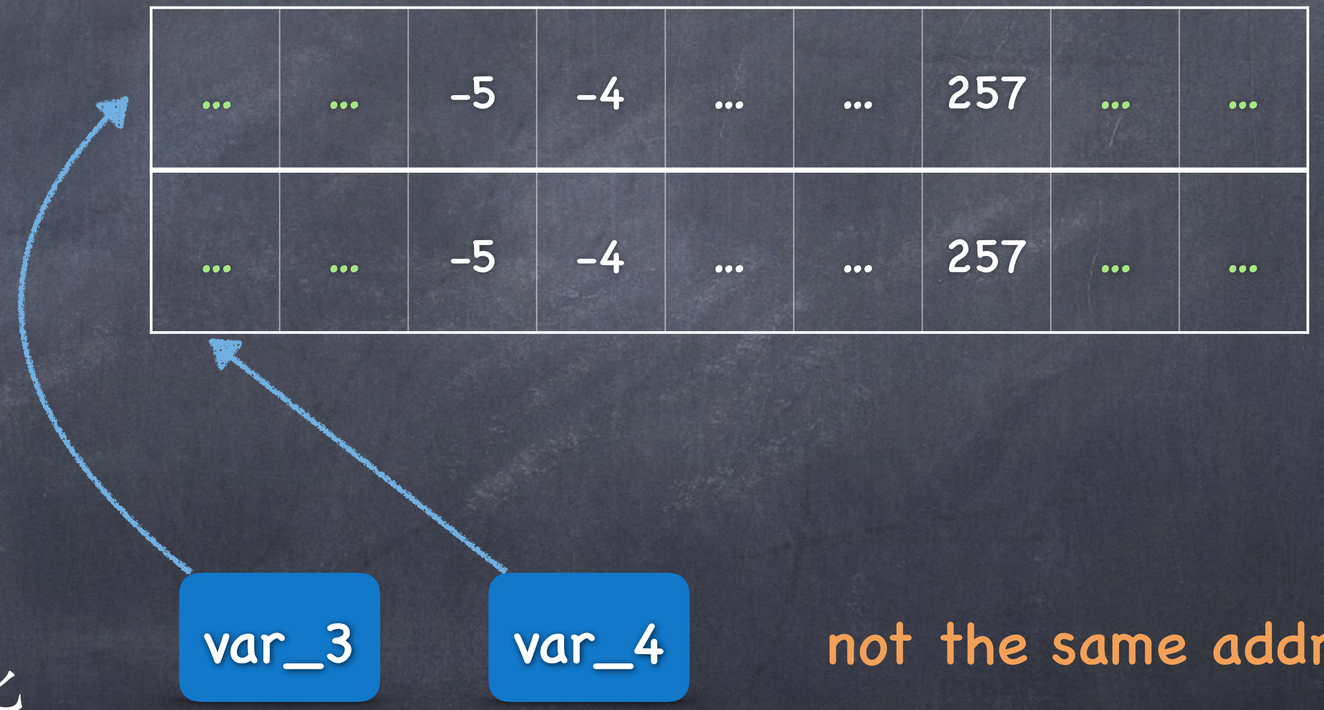
整数对象池

小整数

- 28 bytes
- 解释器初始化

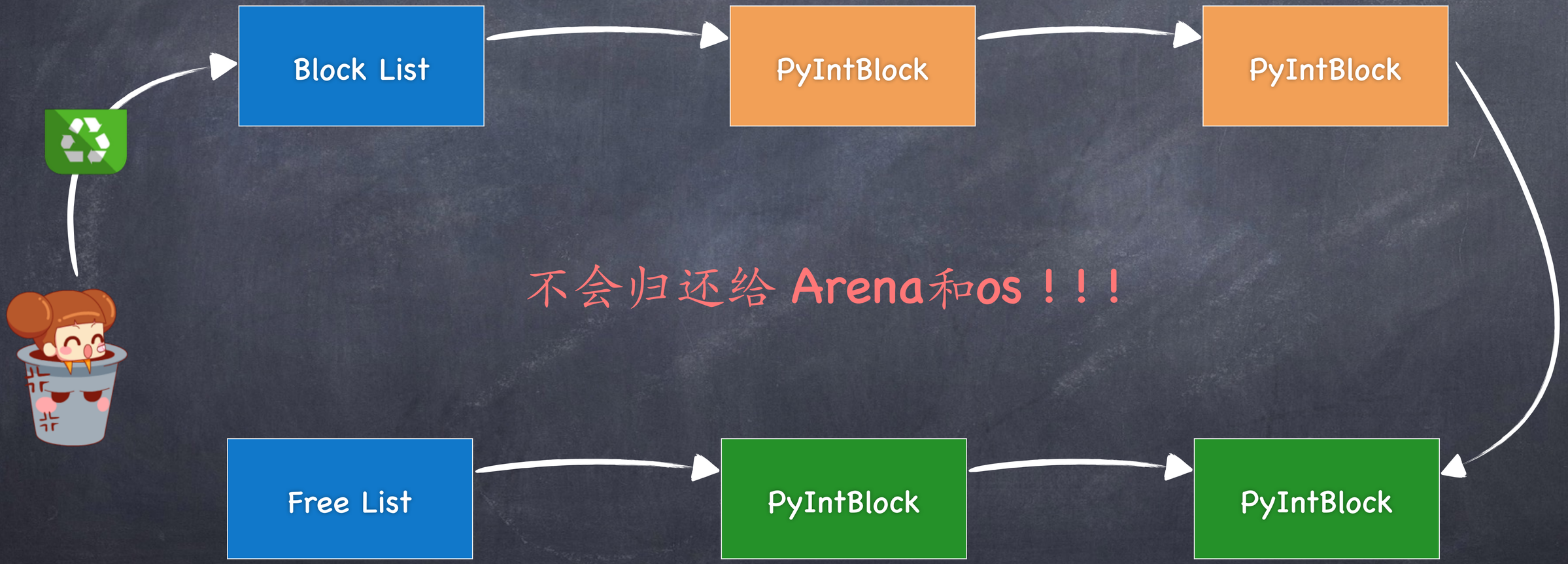
the same addr !

大整数

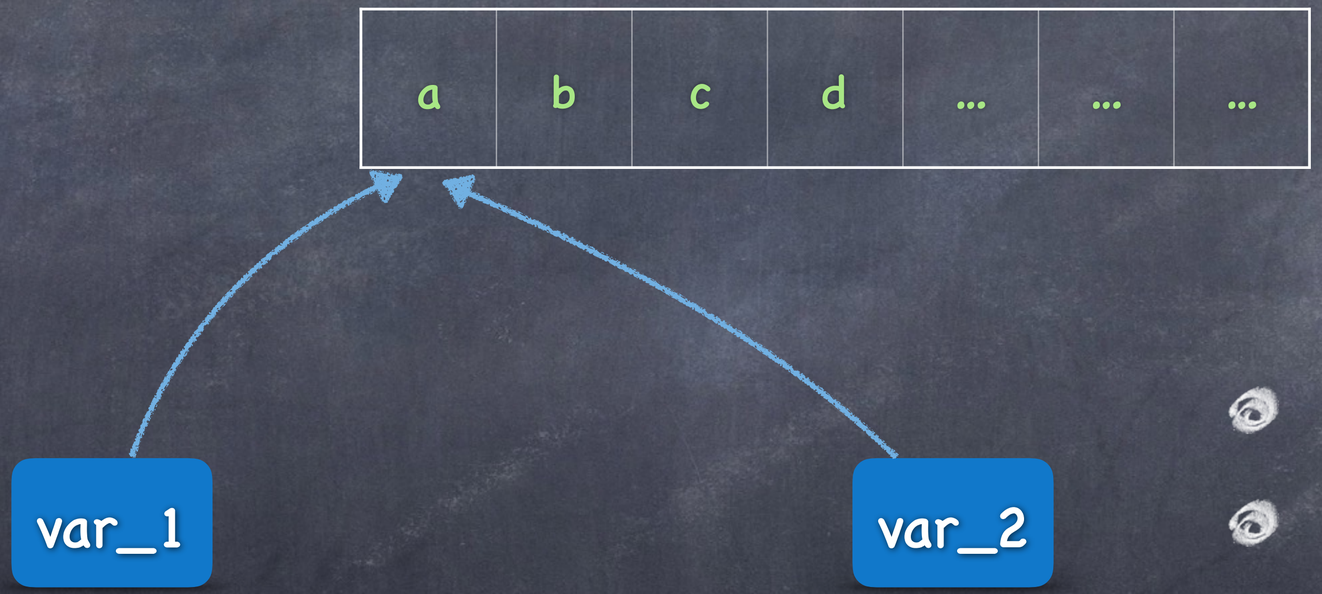


not the same addr

整数对象池

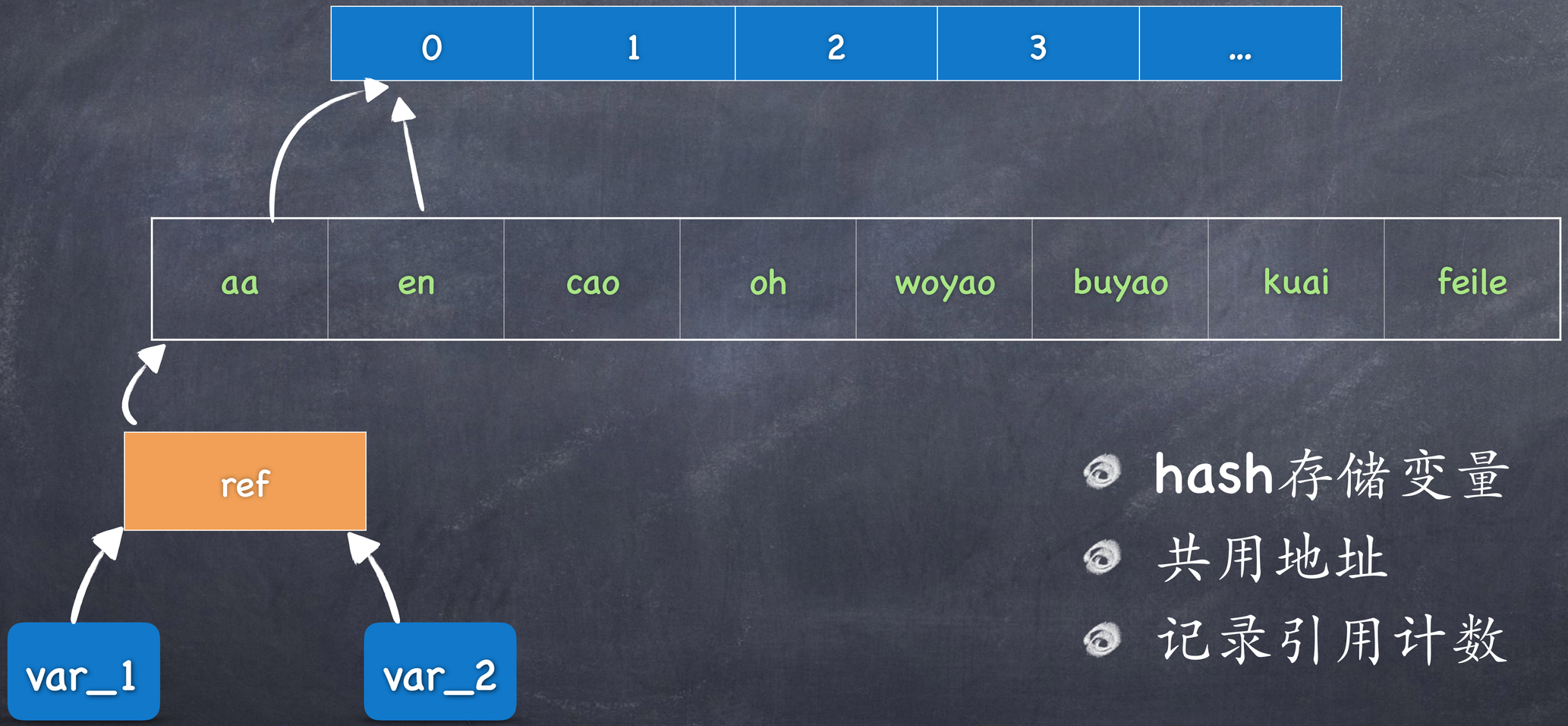

Block List
PyIntBlock
PyIntBlock
不会归还给 Arena和os ！！！
Free List
PyIntBlock
PyIntBlock

字符对象池



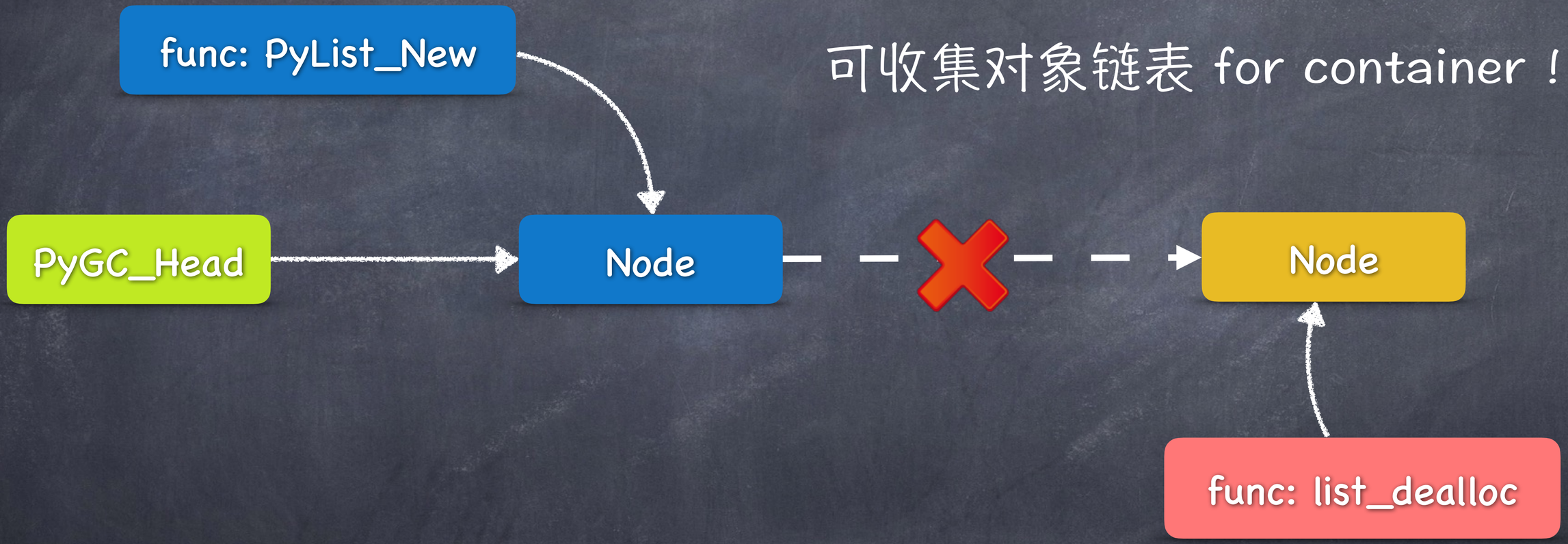
- 单个字符38 bytes
- 由解释器初始化

字符串对象池



- hash存储变量
- 共用地址
- 记录引用计数

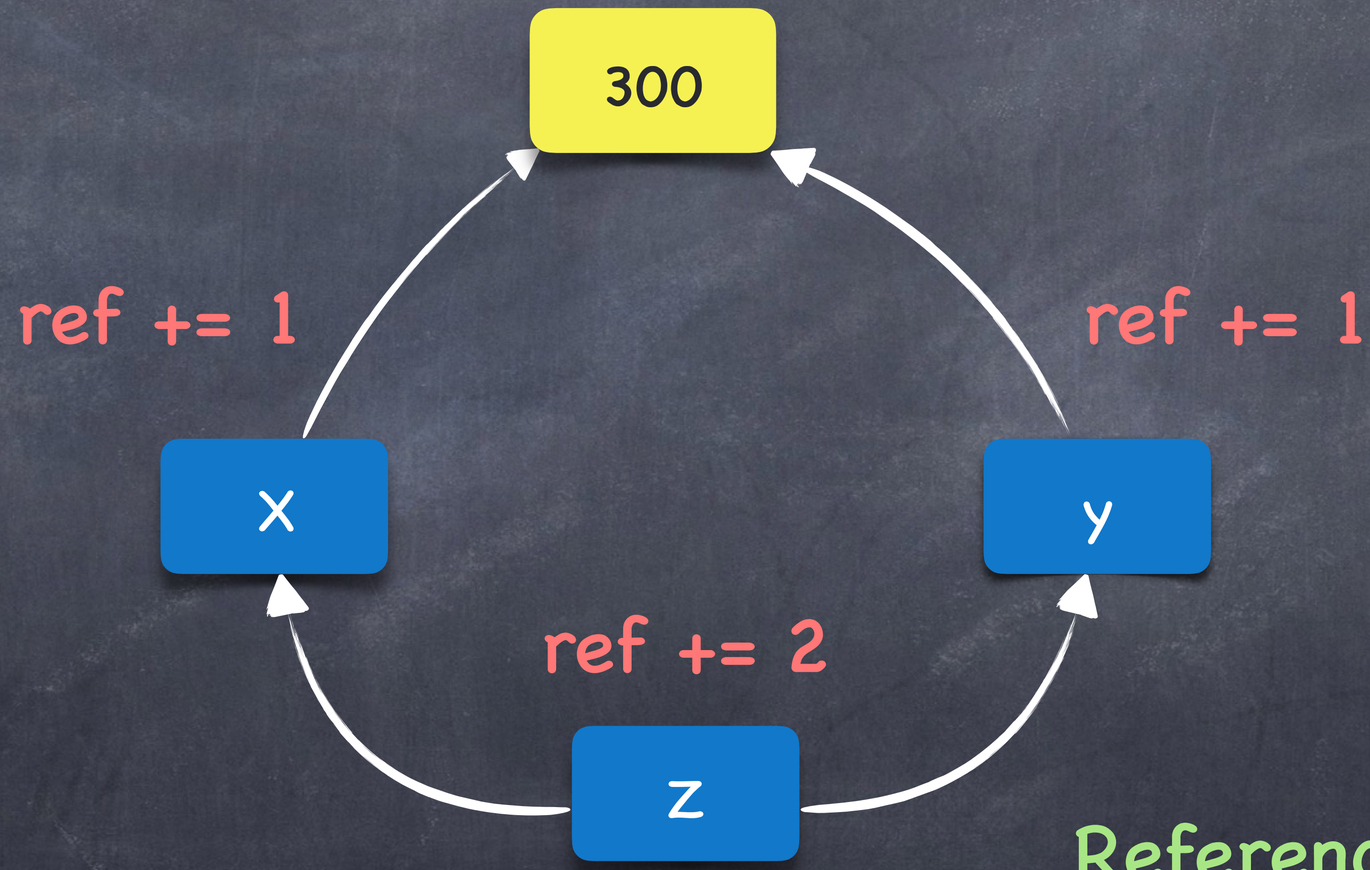
PyObject_GC_TRACK



ref: https://svn.python.org/projects/python/trunk/Objects/listobject.c

ref count

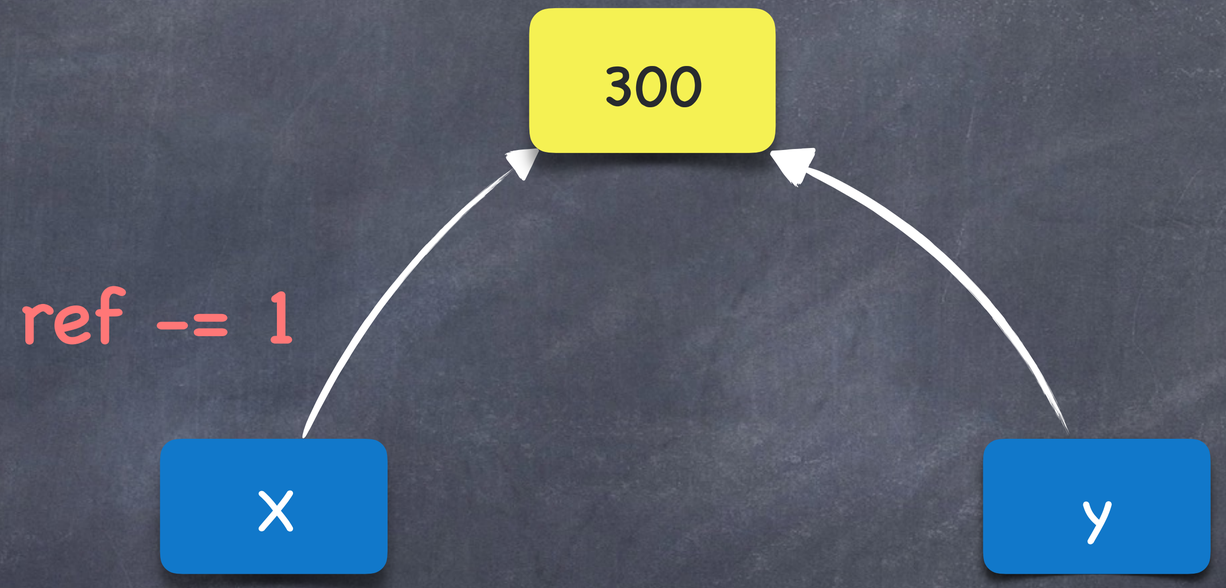
```
x = 300
y = x
z = [x, y]
```



References -> 4 !

What does del do?

```
x = 300
y = x
del x
```



References -> 1 !

The del statement doesn't delete objects.

- removes that name as a reference to that object
- reduces the ref count by 1

ref count case

```
def go():
    w = 300
```

ref count +1

```
go()
```

w is out of scope; ref count -1

```
a = "fuc . "
del a
```

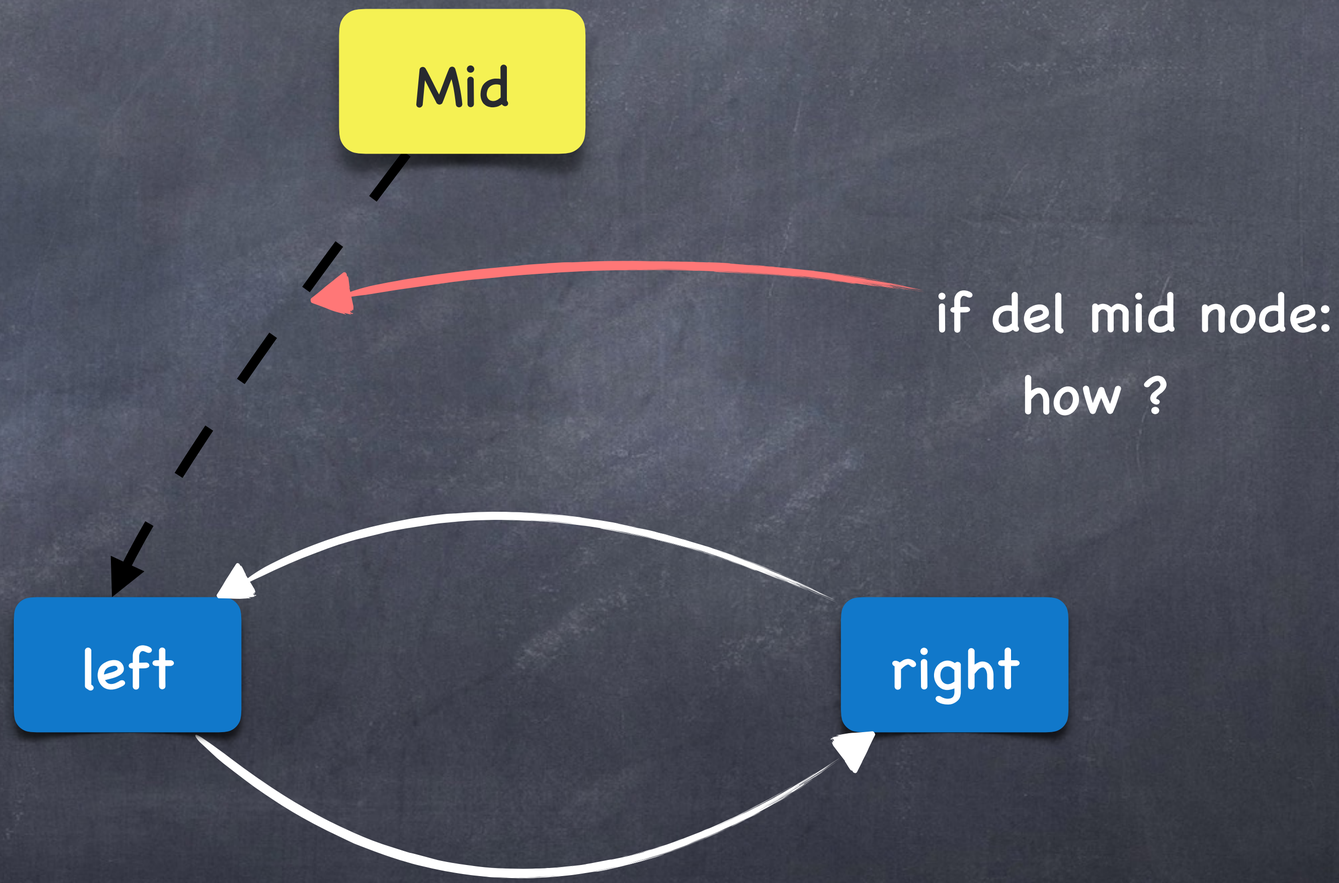
del a; ref count -1

```
b = "en, a"
b = None
```

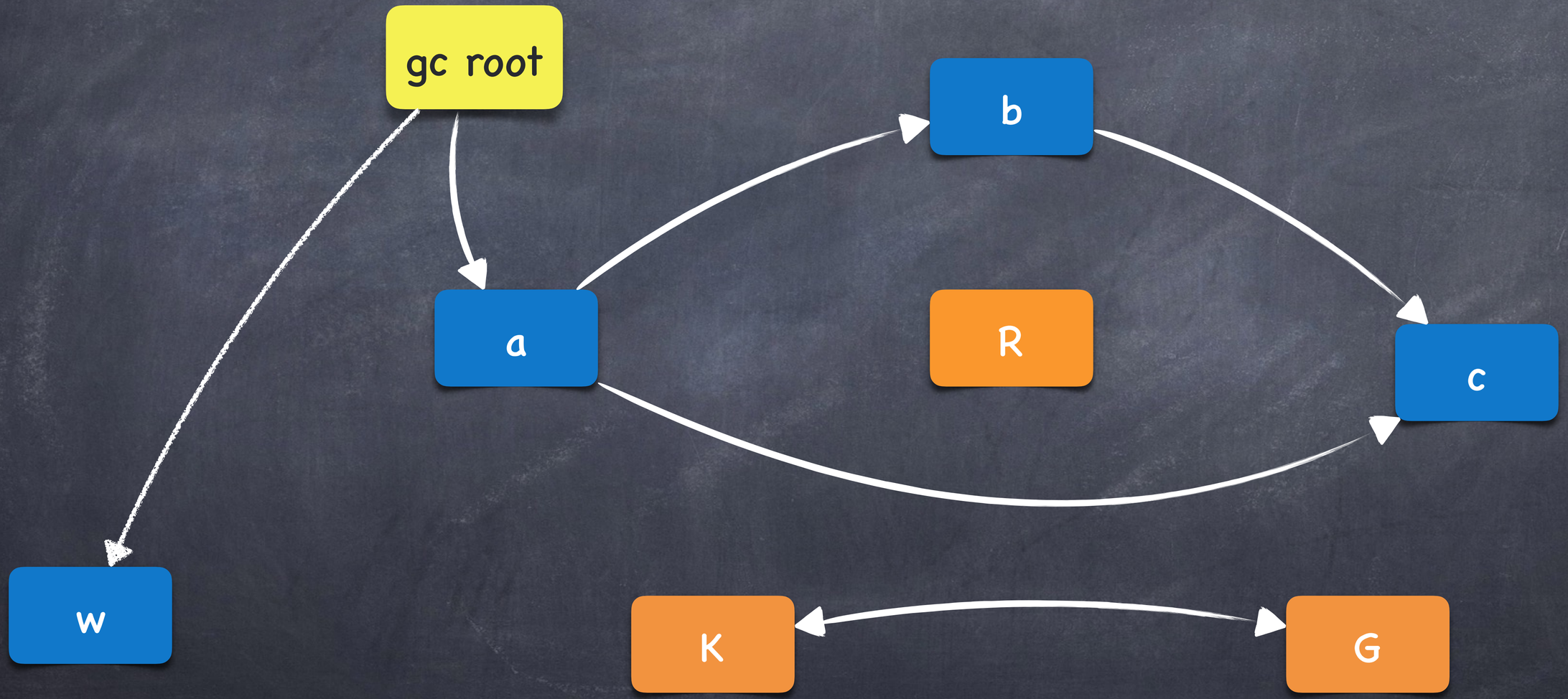
重新赋值; ref count -1

cyclical ref

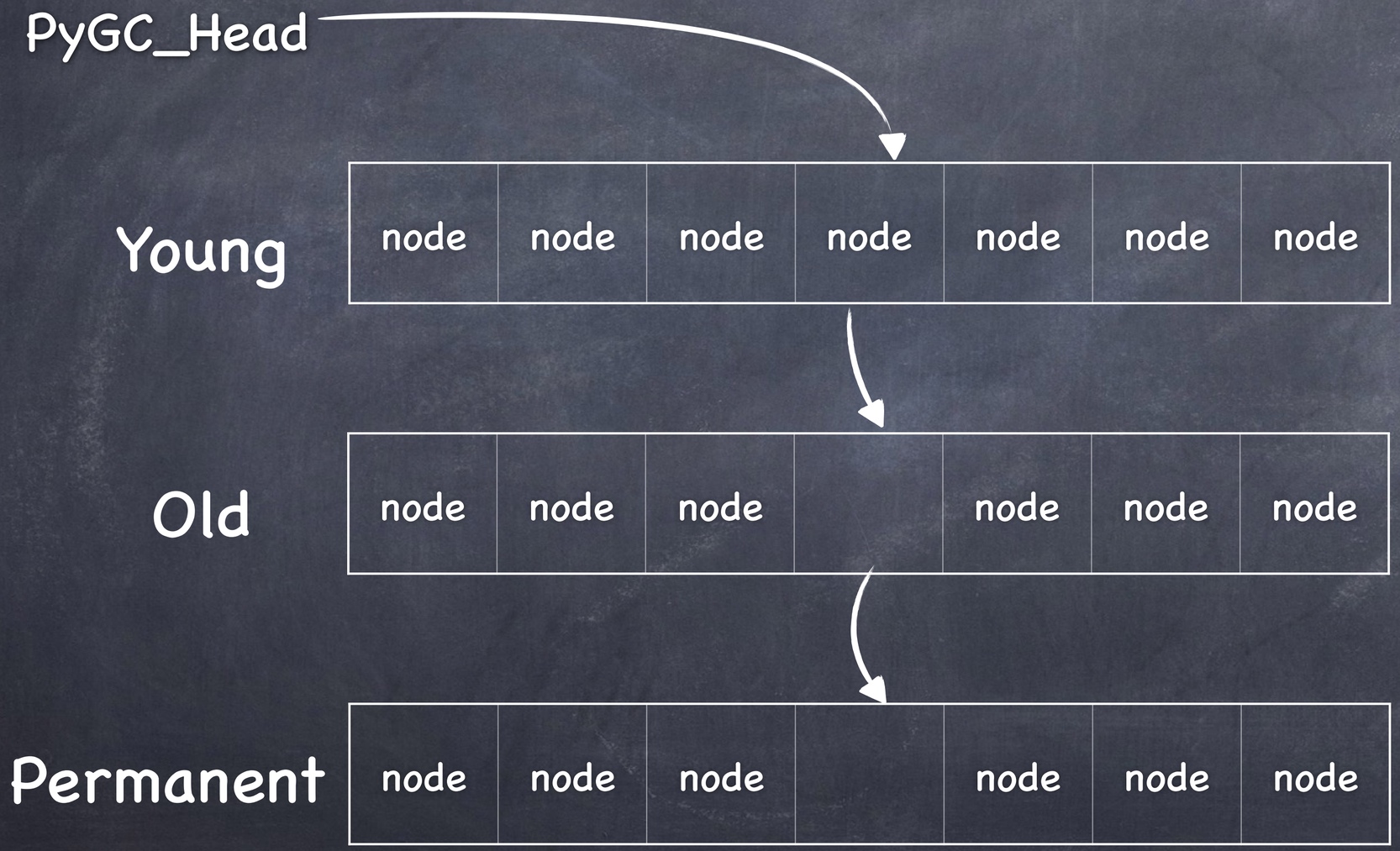
```
class Node:
    def __init__(self, va):
        self.va = va
    def next(self, next):
        self.next = next

mid = Node('root')
left = Node('left')
right = Node('right')

mid(left)
left.next(right)
right.next(left)
```



mark & sweep



分代回收

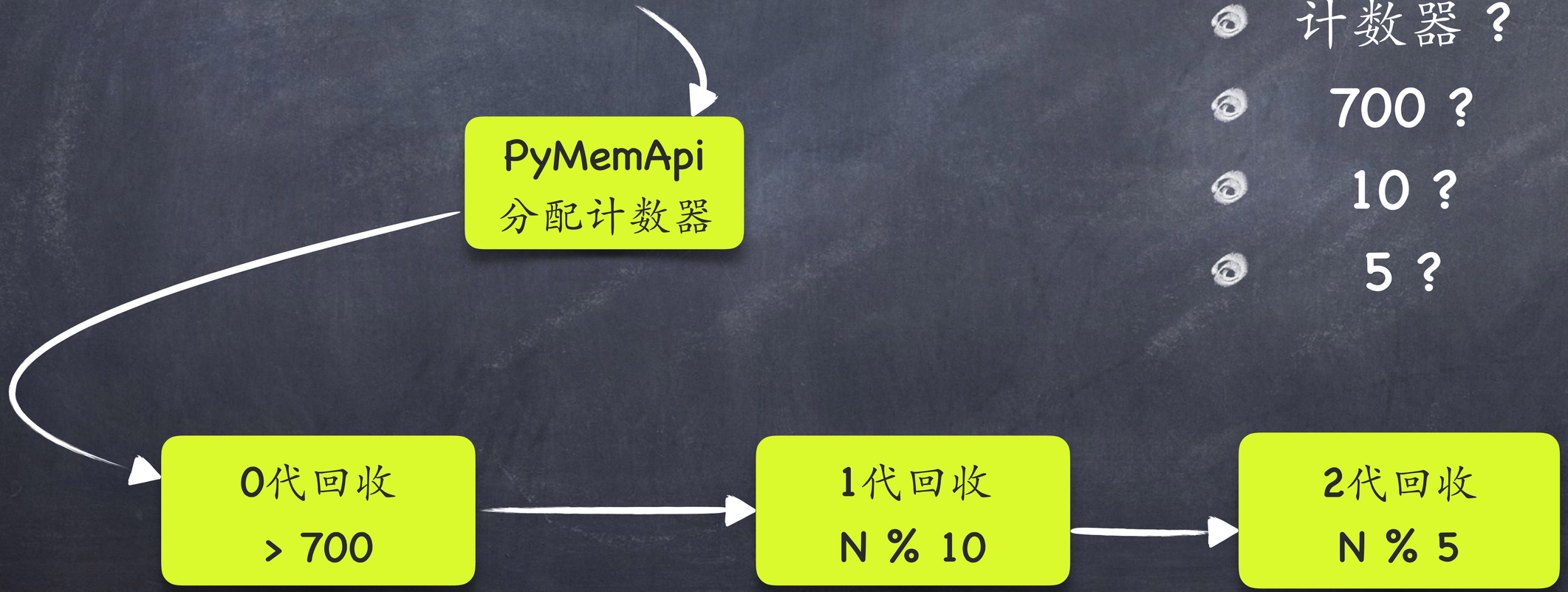


- 分而治之
- 提高效率
- 生命周期
- 空间换时间

when gc

```
import gc
gc.set_threshold(700, 10, 5)
```

- 计数器？
- 700？
- 10？
- 5？



summery

分配内存

-> 发现超过阈值了

-> 触发垃圾回收

-> 将所有可收集对象链表放到一起

-> 遍历，计算有效引用计数

-> 分成 有效引用计数=0 和 有效引用计数 > 0 两个集合

-> 大于0的，放入到更老一代

-> =0的，执行回收

-> 回收遍历容器内的各个元素，减掉对应元素引用计数(破掉循环引用)

-> 执行-1的逻辑，若发现对象引用计数=0，触发内存回收

-> python底层内存管理机制回收内存

weakref 弱引用

```
class  Expensive(object):
    def __del__(self):
        print '(Deleting %s)' % self

obj = Expensive()
r = weakref.ref(obj)
del obj
print 'r():', r()
```

- 不参与引用计数
- 解决循环引用

```
class Parent(object):
    def __init__(self):
        self.children = [ Child(self) ]

class Child(object):
    def __init__(self, parent):
        self.parent = weakref.proxy(parent)
```

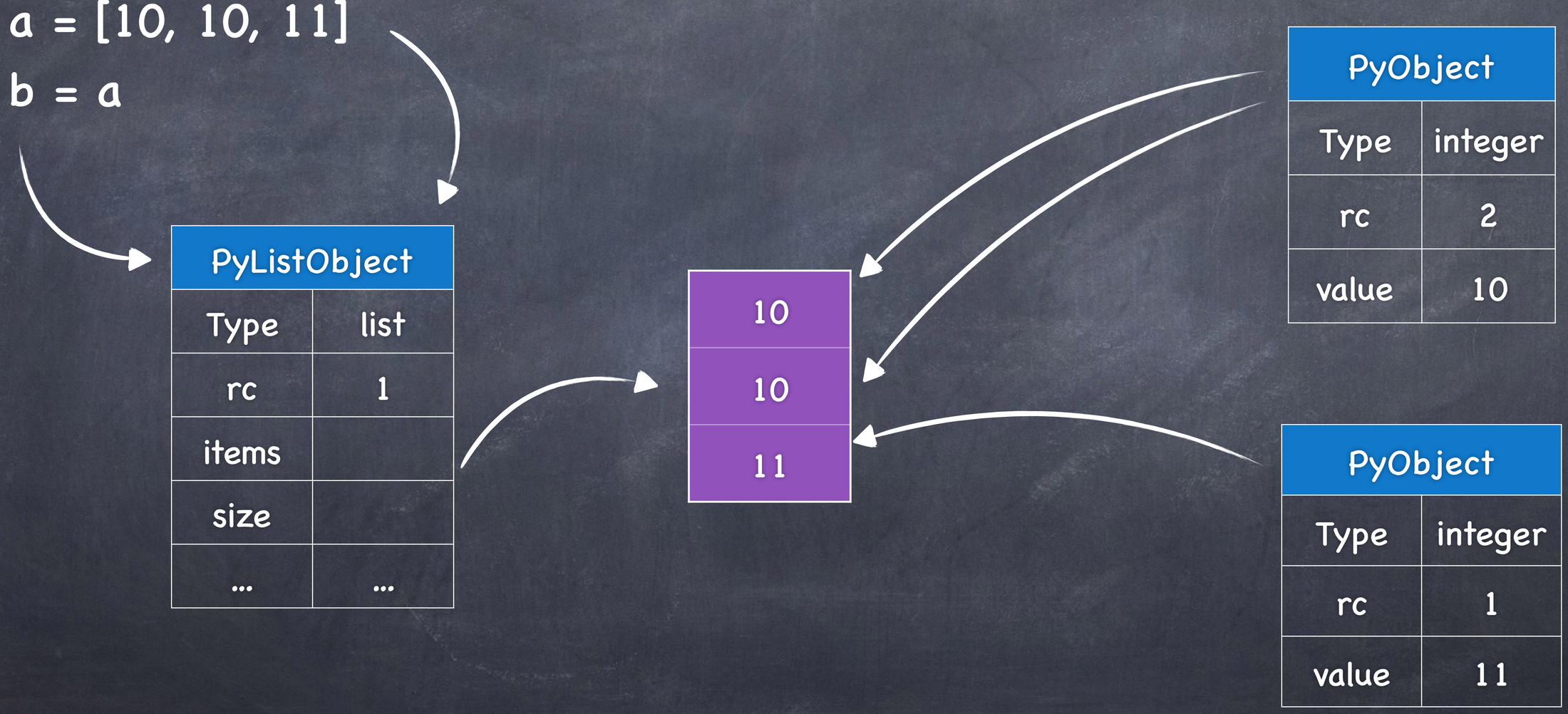
可变 vs 不可变 (obj)

- list
- dict

- string
- int
- tuple

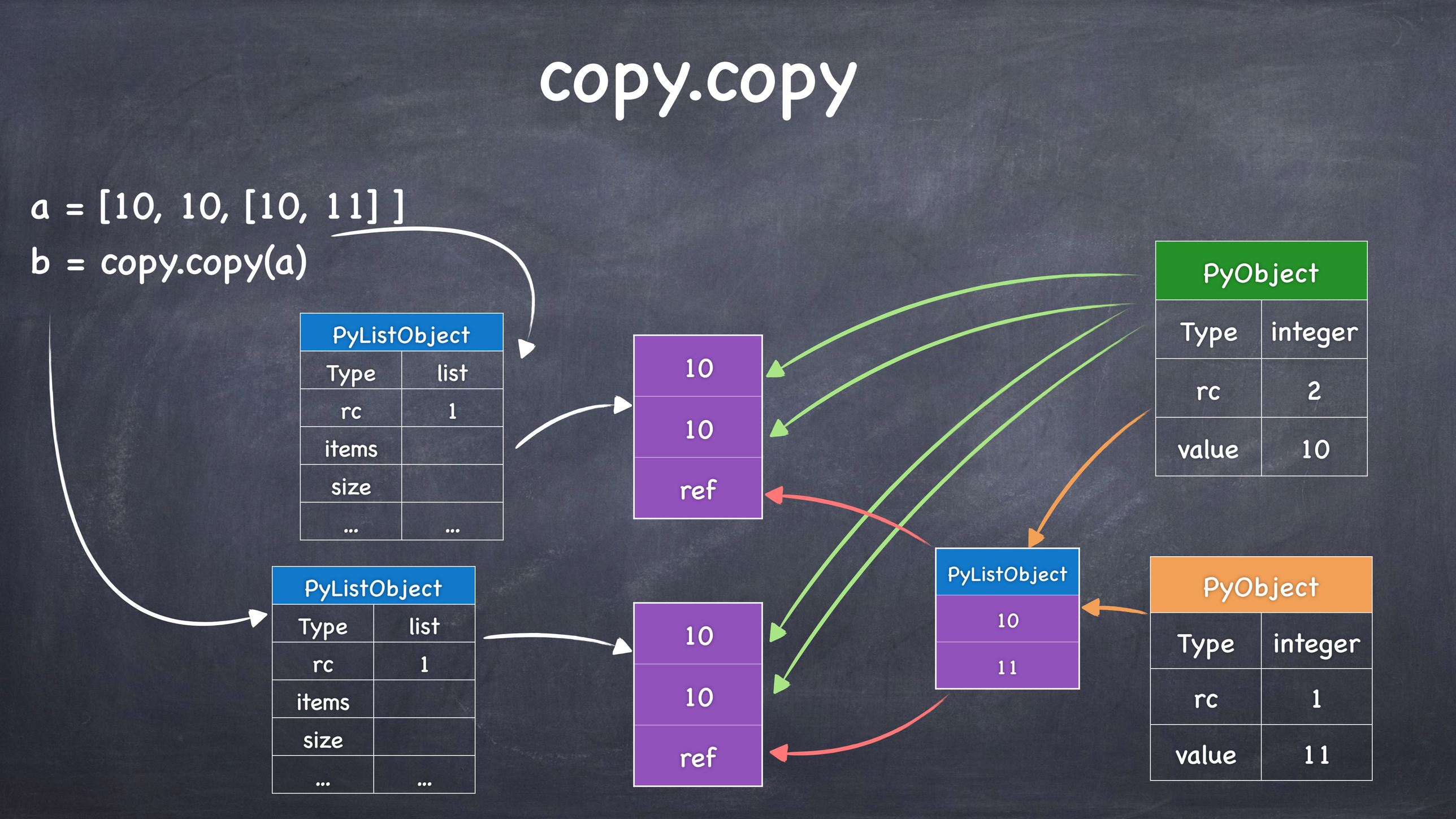
container objects

```
a = [10, 10, 11]
b = a
```



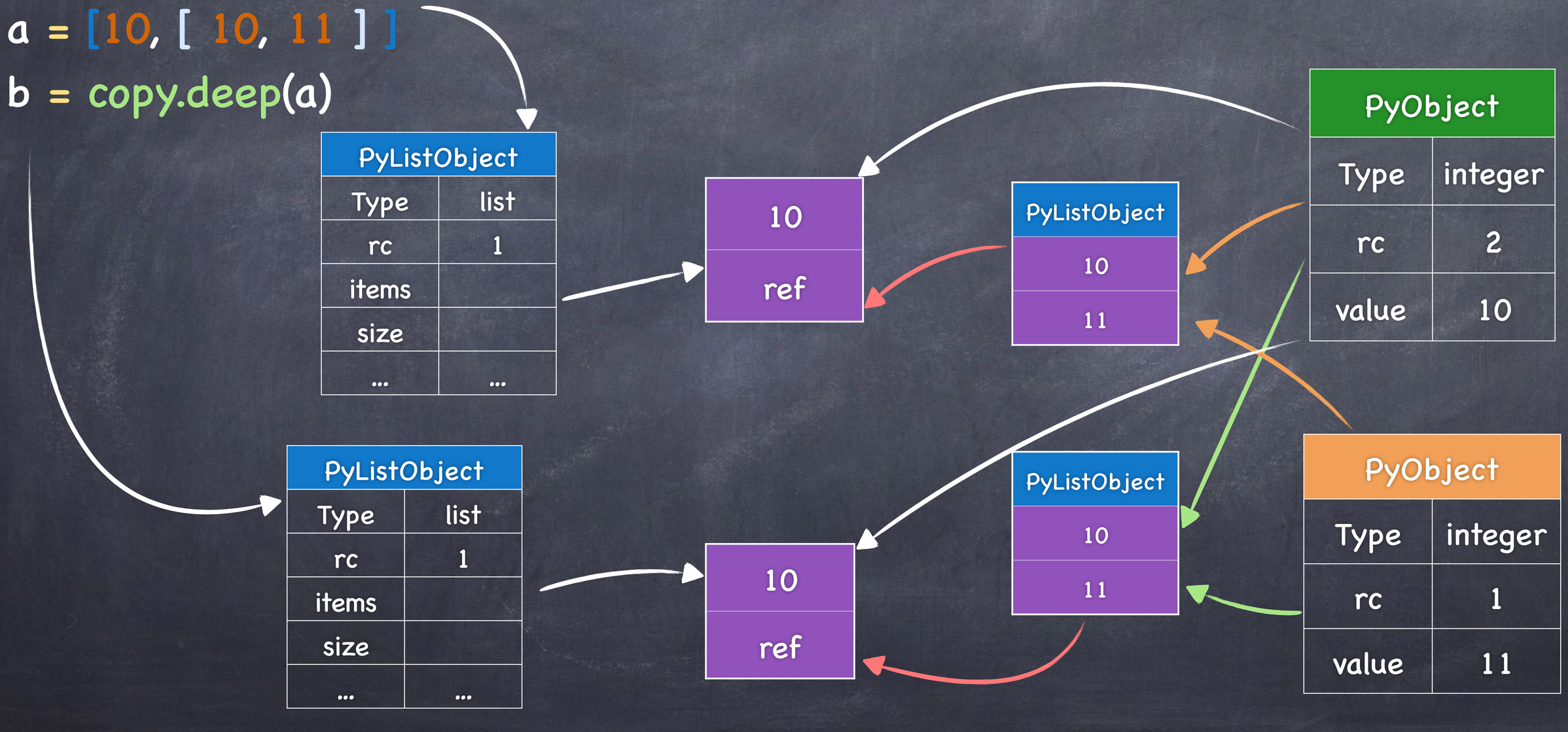
copy.copy

```
a = [10, 10, [10, 11] ]
b = copy.copy(a)
```



copy.deepcopy

```
a = [10, [ 10, 11 ] ]
b = copy.deep(a)
```



diy gc

```
import gc
import sys
gc.set_debug(gc.DEBUG_STATS|gc.DEBUG_LEAK)
a=[]
b=[]
a.append(b)
print 'a refcount:',sys.getrefcount(a)  # 2
print 'b refcount:',sys.getrefcount(b)  # 3

del a
del b
print gc.collect()  # 0
```

Garbage Collector Optimize

- memory bound
  - 可以降低threshold来时间换空间
- cpu bound
  - 提高threshold来空间换时间
  - 暂停gc，引入master worker设计

Q & A

- 引用计数 跟 gil 的影响？
- gc 是否是原子？
- gc的 stop the world现象？
- ...

“ END ”

– xiaorui.cc