Углубленный Python

Лекция 3

Латкин Игорь

“

Не забудьте отметиться на занятии!

Цитата великих

Повестка дня

1. Garbage Collection
2. Устройство аллокатора памяти в CPython
3. CPython – внутреннее устройство
4. Процесс интерпретации кода
5. PyObject – из чего состоит
6. Устройство list, dict, tuple, …

“

The only reliable way to free memory is to terminate the process.

CPython

Первая и основная реализация Python

Другие существующие реализации:

1. PyPy
2. IronPython – .net CLR
3. Jython – JVM

https://wiki.python.org/moin/PythonImplementations

Garbage Collection

Garbage Collection

- ⊕ Не нужно думать об очистке памяти
- ⊕ Никаких double free ошибок
- ⊕ Решение проблем утечки памяти

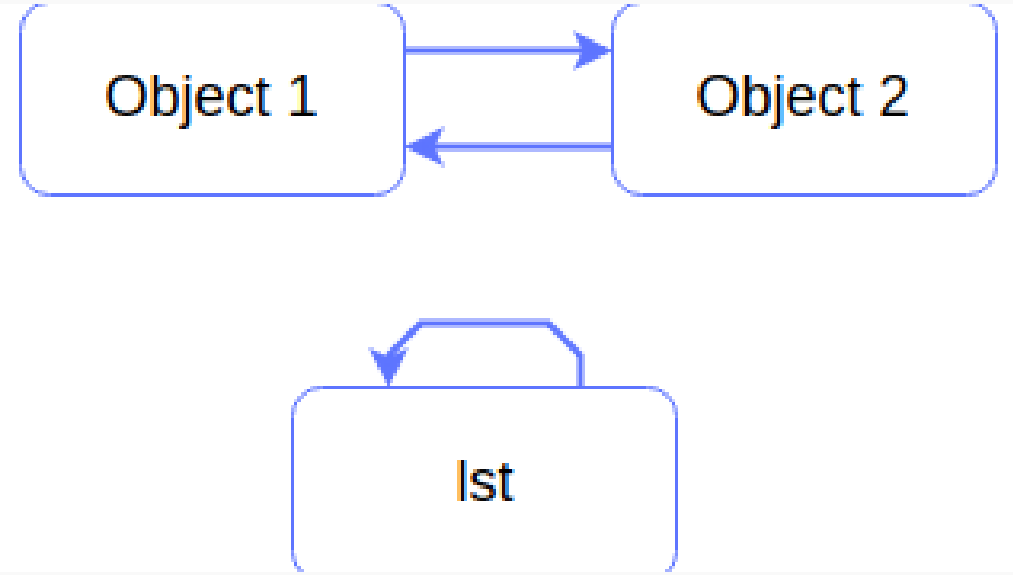
- ⊖ Дополнительное использование CPU и RAM
- ⊖ Момент сборки мусора непредсказуем

Reference Counting (REFCOUNT)

Что выведет программа?

```
1.  import sys

2.  foo = []

3.  print(sys.getrefcount(foo))

4.  def bar(a):
5.      print(sys.getrefcount(a))

6.  bar(foo)
7.  print(sys.getrefcount(foo))
```

REFCOUNT

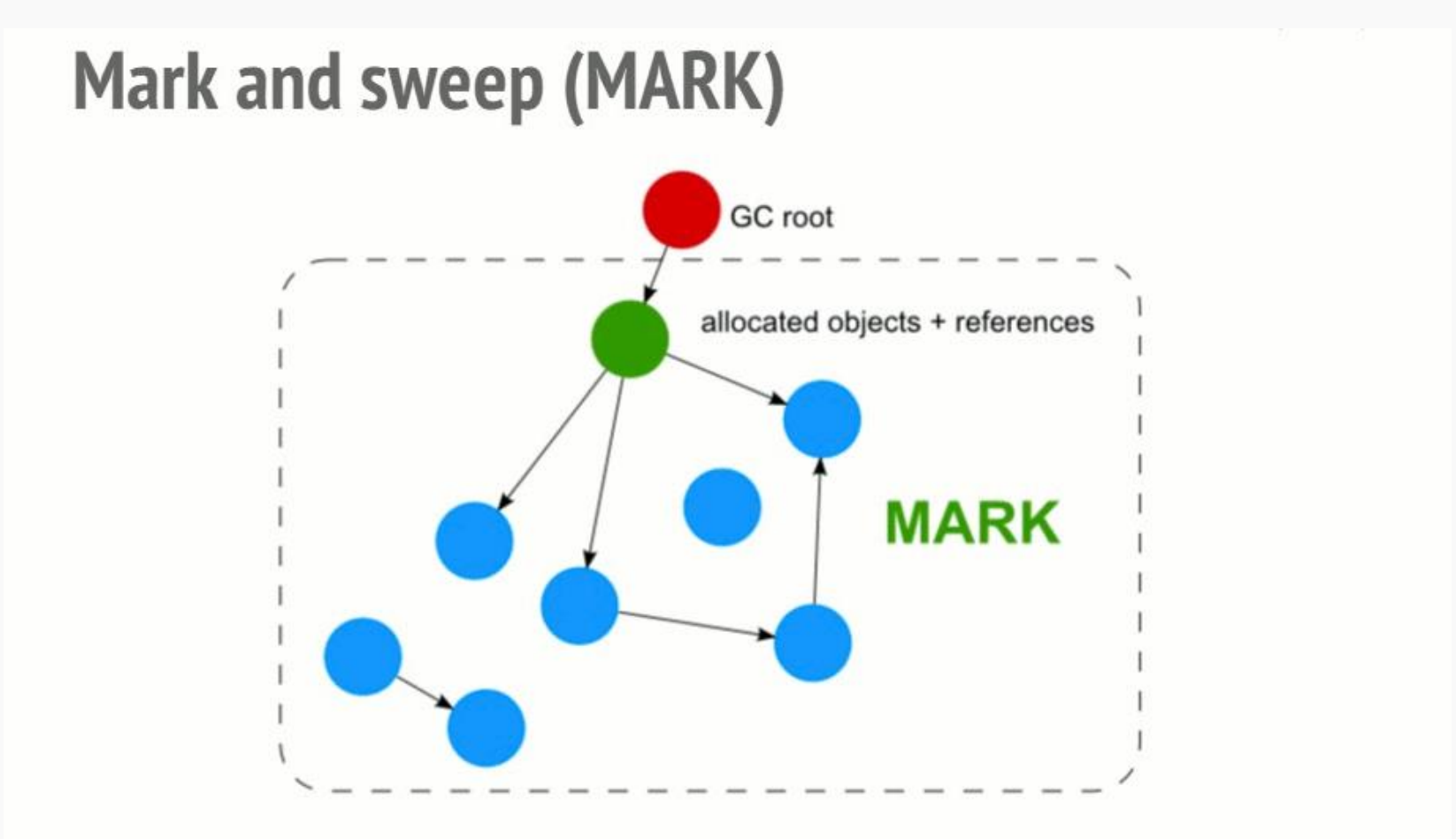


Недостатки REFCOUNT

- (+) Память сразу можно очистить

- (-) Circular references
- (-) Thread-lock
- (-) Performance overhead

Mark & Sweep GC



Garbage Collection

Generational GC (from Python 1.5)

GC следит только за объектами-контейнерами
(если они содержат тоже объекты-контейнеры)

1. List
2. Dict
3. Tuples
4. Classes
5. ….

https://docs.python.org/3/library/gc.html

DEMO

Garbage Collection

Generational GC (from Python 1.5)

http://arctrix.com/nas/python/gc
https://rushter.com/blog/python-garbage-collector
https://docs.python.org/3/library/weakref.html
https://www.python.org/dev/peps/pep-0442/

Memory Management

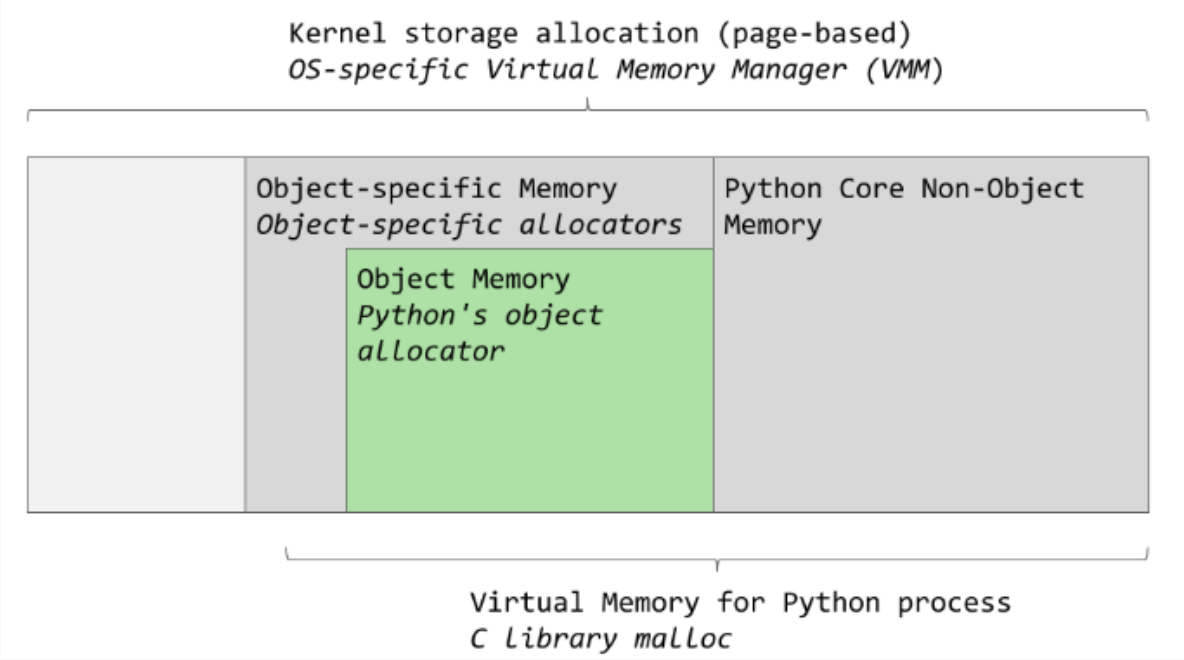
PyObject

```
109  typedef struct _object {
110      _PyObject_HEAD_EXTRA
111      Py_ssize_t ob_refcnt;
112      struct _typeobject *ob_type;
113  } PyObject;
```

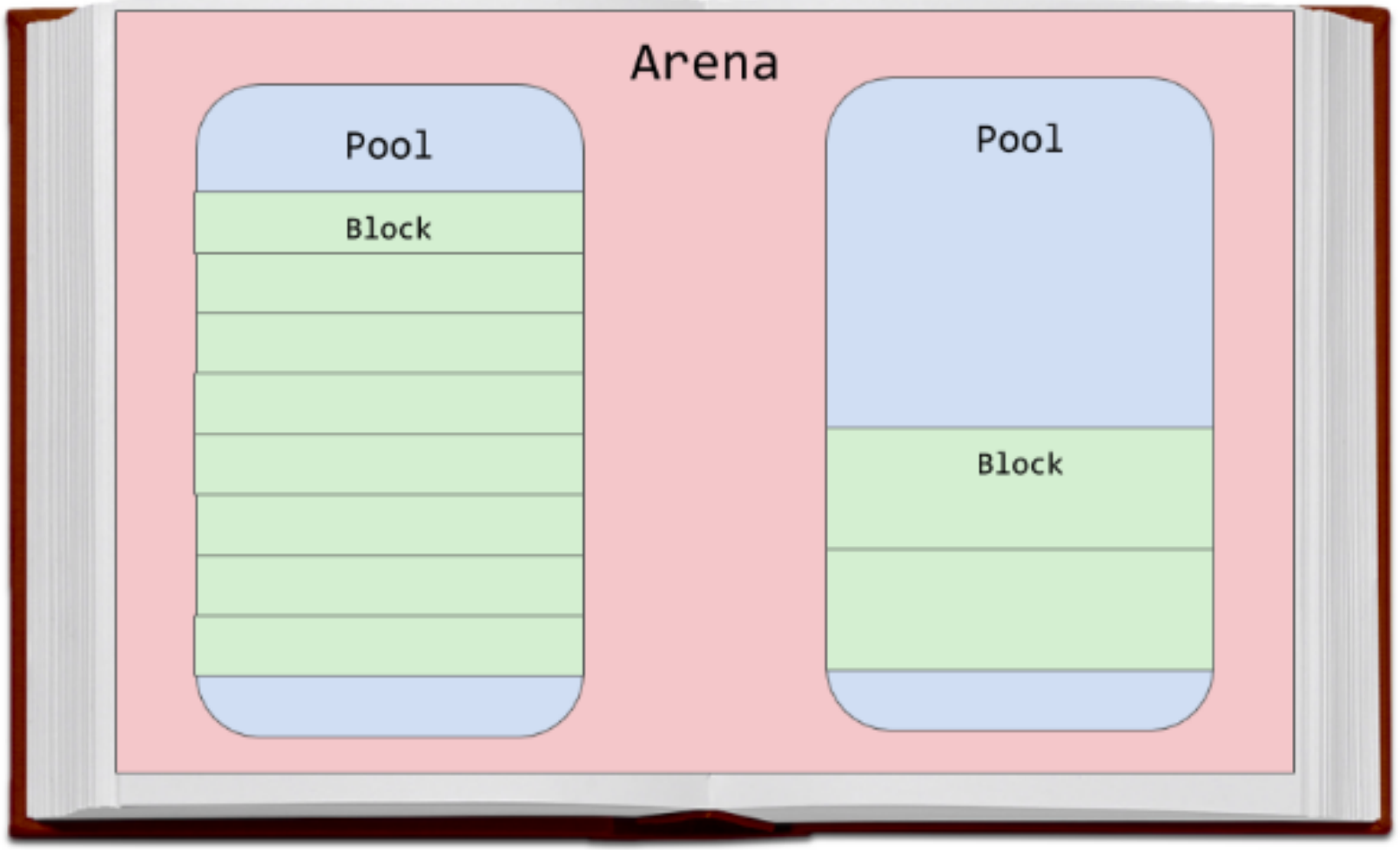
Memory Management

Где все происходит?

Objects/obmalloc.c



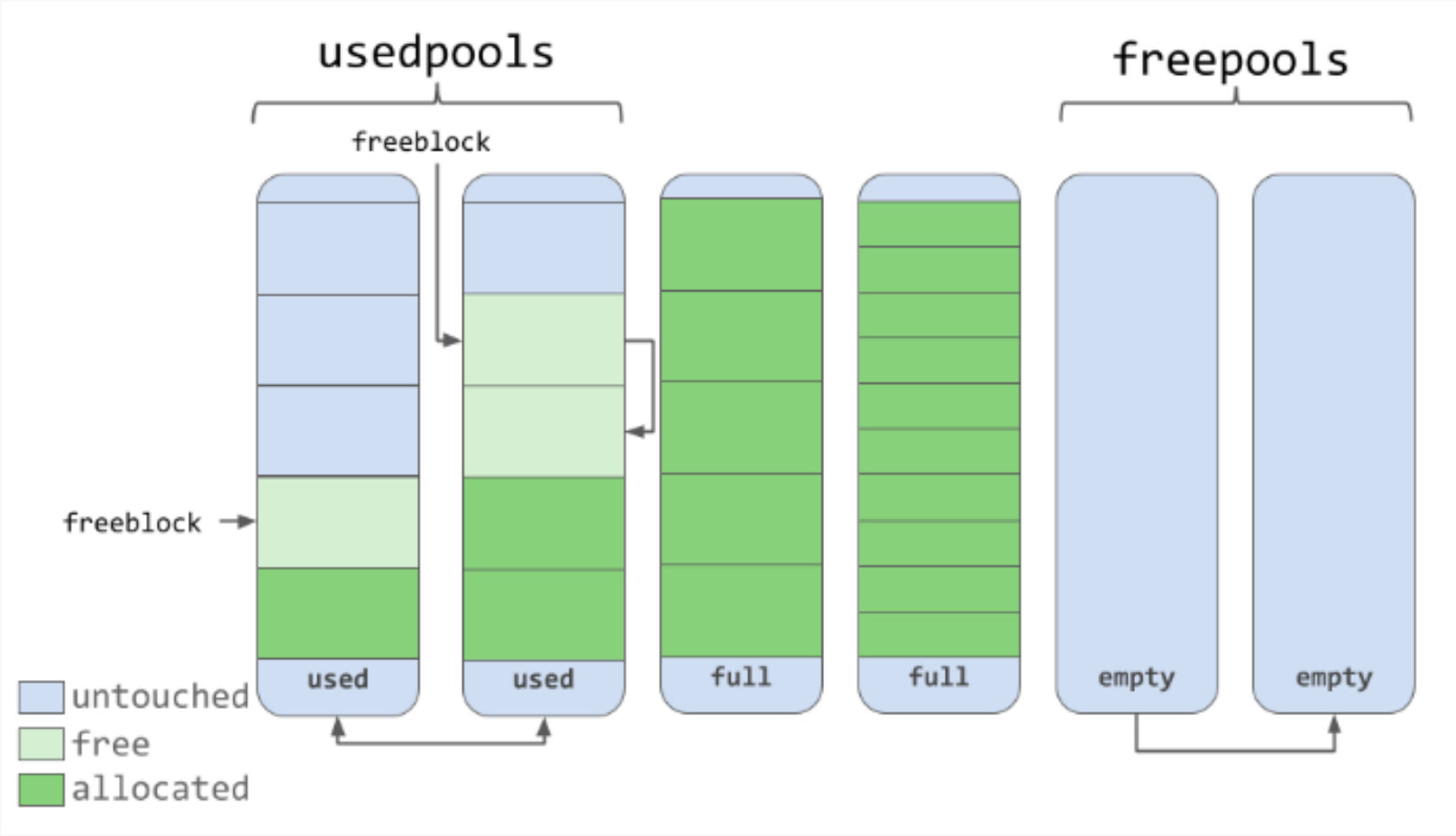
Arena & Pool



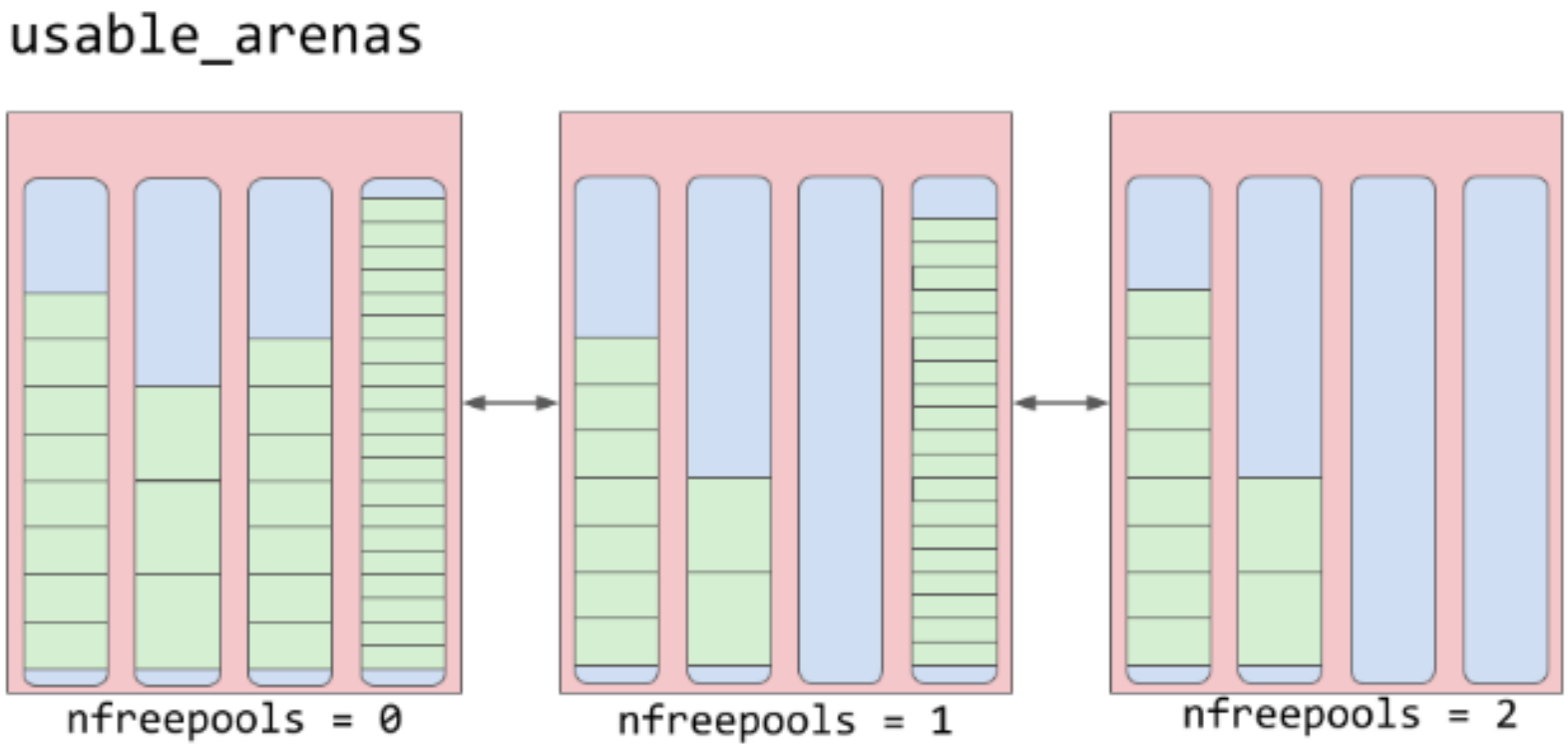
Blocks


```
 * 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
 *       65-72                  72                       8
 *         ...                 ...                     ...
 *      497-504                504                      62
 *      505-512                512                      63
 *
 *      0, SMALL_REQUEST_THRESHOLD + 1 and up: routed to the underlying
 *      allocator.
 */
```

Blocks


usedpools
freepools
freeblock
freeblock
used
used
full
full
empty
empty
untouched
free
allocated

Arenas



В следующей лекции…

1. Процесс интерпретации кода
2. PyObject – из чего состоит
3. Устройство list, dict, tuple, …

telegram: alexopryshko
email: alexopryshko@gmail.com

telegram: igorcoding
email: igor.latkin@outlook.com

Спасибо за внимание!