ТЕХНОАТОМ

Углубленный Python

Лекция 1: работа с памятью

Кандауров Геннадий

Не забудьте отметиться на занятии!

Преподаватели

ТЕХНОАТОМ

Алёна Елизарова

Антон Кухтичев

Геннадий Кандауров

Программа курса

ТЕХНОАТОМ

1. Работа с памятью
2. Объектная модель
3. c-extensions
4. Метаклассы
5. Типы и структуры данных
6. Профилирование, тестирование, логирование
7. Сетевое взаимодействие
8. Потоки GIL, Threads, multiprocessing, IPC
9. Асинхронное программирование
10. Работа с файлами и каталогами
11. Публикация приложений
12. Хакатон

Повестка дня

ТЕХНОАТОМ

1. Garbage Collection
2. Устройство аллокатора памяти в CPython
3. CPython - внутреннее устройство
4. Процесс интерпретации кода
5. Устройство PyObject
6. Устройство list, dict, tuple, etc

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

CPython

ТЕХНОАТОМ

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

ТЕХНОАТОМ

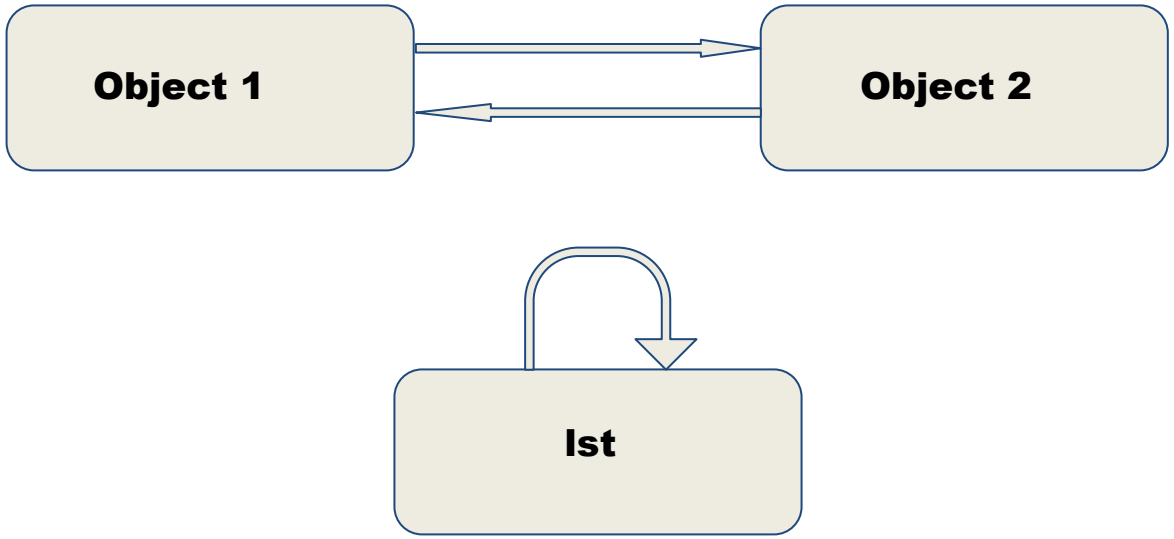
- \+ Не нужно думать об очистке памяти
- \+ Никаких double free ошибок
- \+ Решение проблем утечек памяти
- − Дополнительное использование CPU и RAM
- − Момент сборки мусора непредсказуем

Reference Counting (REFCOUNT)

ТЕХНОАТОМ

Что выведет программа**?**

```
>>> import sys
>>> foo = []
>>> print(sys.getrefcount(foo))
>>> def bar(a):
        print(sys.getrefcount(a))
>>> bar(foo)
>>> print(sys.getrefcount(foo))
```


Object 1
Object 2
lst

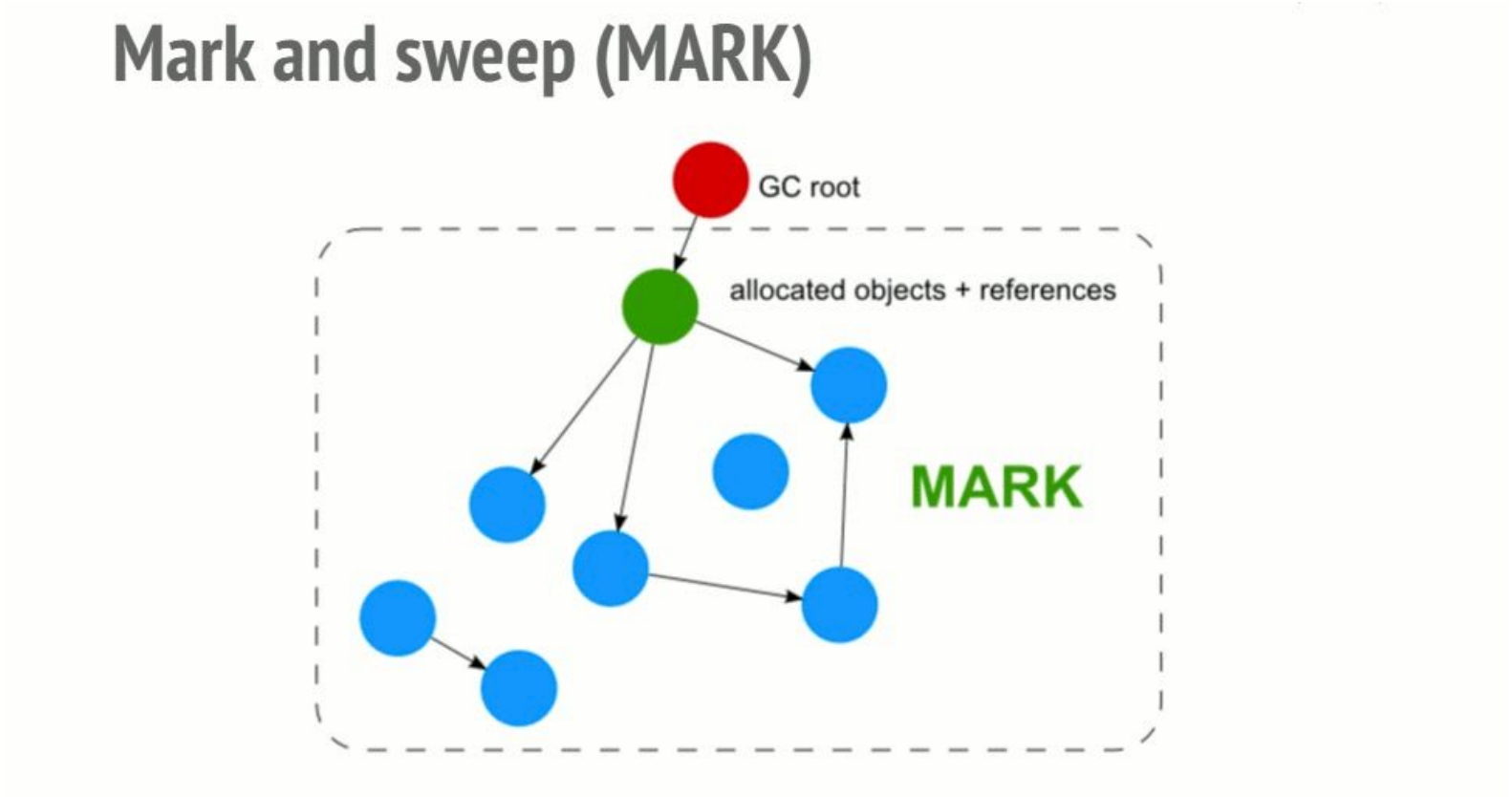
REFCOUNT

ТЕХНОАТОМ

- \+ Память сразу можно очистить
- − Циклические ссылки
- − Блокирование потоков
- − Доп расход CPU и RAM

Mark & Sweep GC

TEXHOATOM



Generational GC (from Python 1.5)

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

1. list
2. dict
3. tuple
4. class
5. etc

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

Возможные оптимизации

ТЕХНОАТОМ

1. Отключение gc

   ```
   gc.disable()
   gc.collect()
   ```

2. `weakref` (https://docs.python.org/3/library/weakref.html)

   `weakref.ref`

   `WeakKeyDictionary, WeakValueDictionary, WeakSet,`

   `WeakMethod`

   `finalize`

   `не работает с list, dict, object`

3. избегать циклических ссылок

TEXHOATOM

```
>>> import weakref
>>> class Object:
...     pass
...
>>> o = Object()
>>> r = weakref.ref(o)
>>> o2 = r()
>>> o is o2
True
>>> del o, o2
>>> print(r())
None
```

__slots__

ТЕХНОАТОМ

1. Экономия памяти, побочный эффект - ускорение
2. Объект не хранит `__dict__`
3. Допускается объявление `__dict__` в `__slots__`(может свести экономию памяти на нет)
4. Не наследуется
5. `__weakref__`

“Бездумная оптимизация еще хуже преждевременной”

Лучано Рамальо, “Python. К вершинам мастерства”

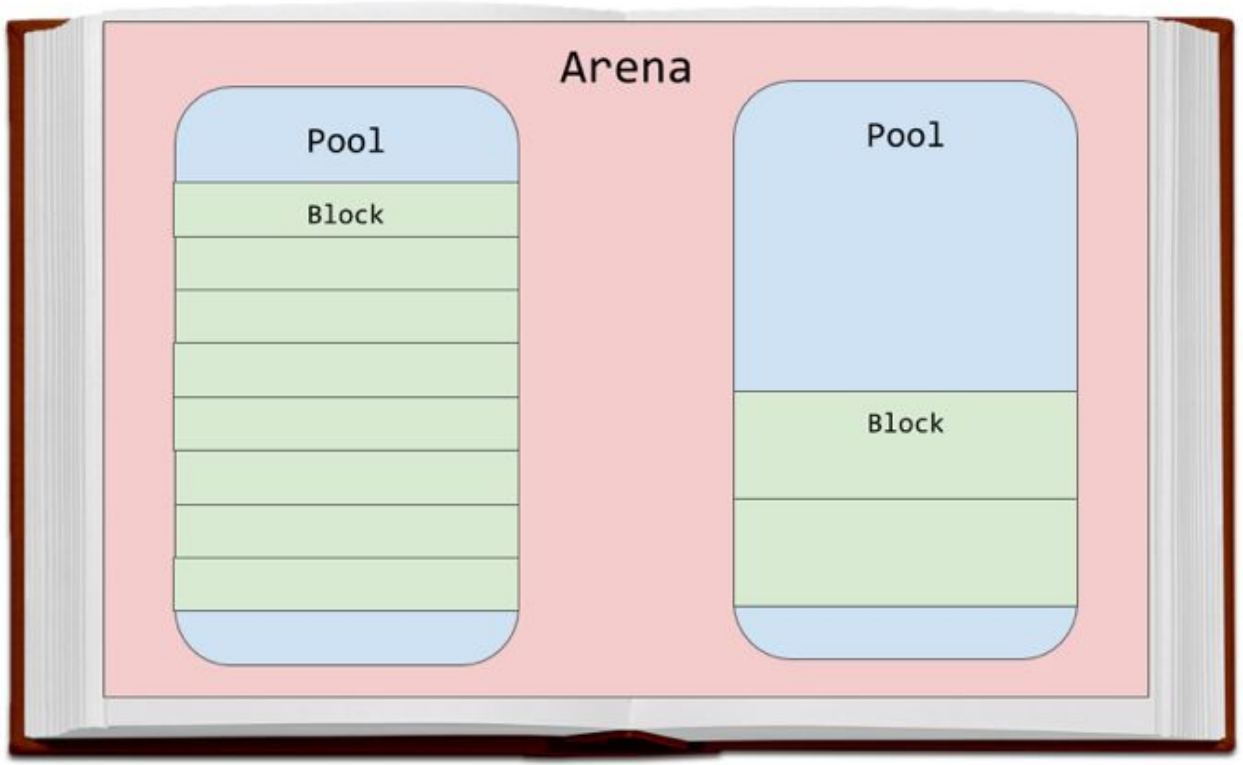
PyObject

TEXHOATOM

```
typedef struct _object {
    _PyObject_HEAD_EXTRA
    Py_ssize_t ob_refcnt;
    PyTypeObject *ob_type;
} PyObject;
```

Управление памятью

ТЕХНОАТОМ

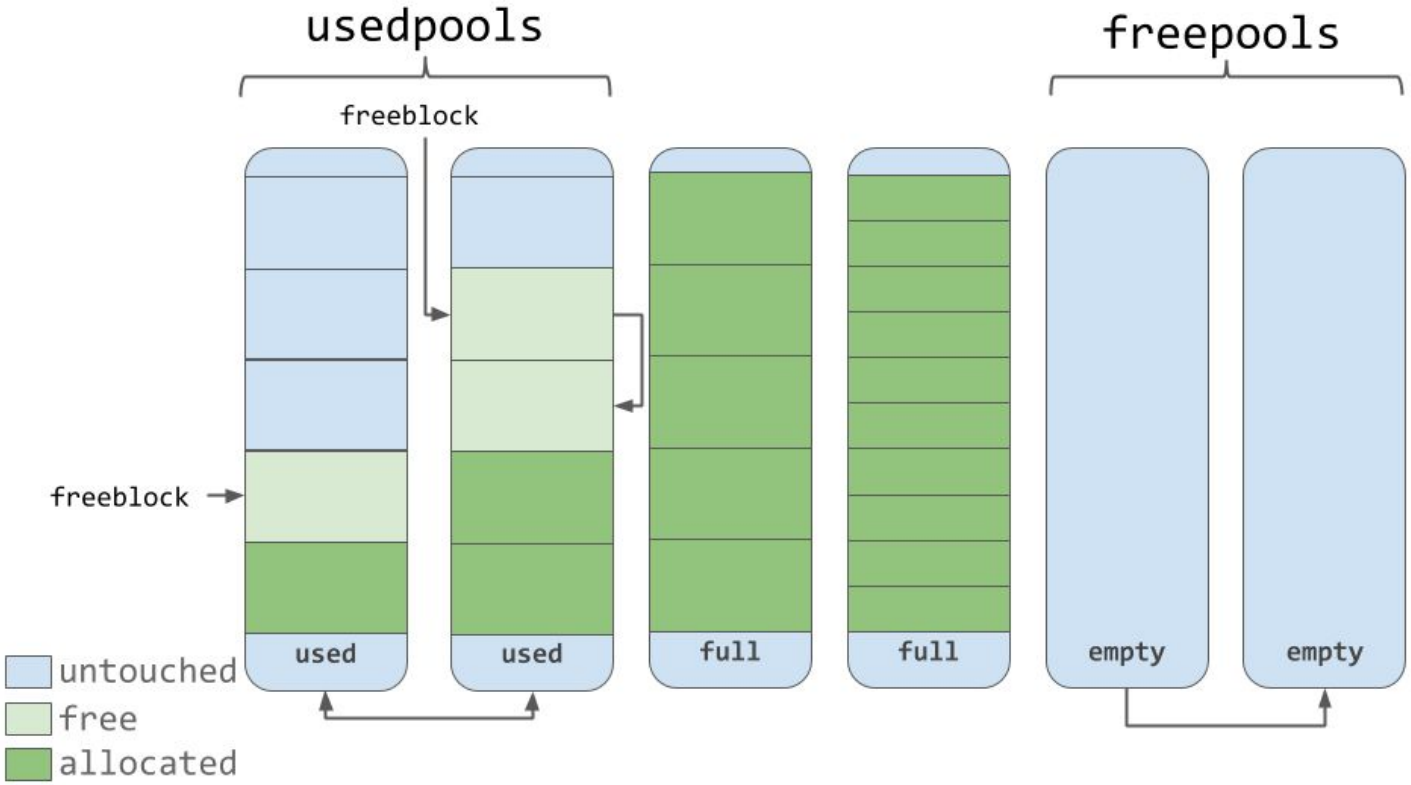


Управление памятью

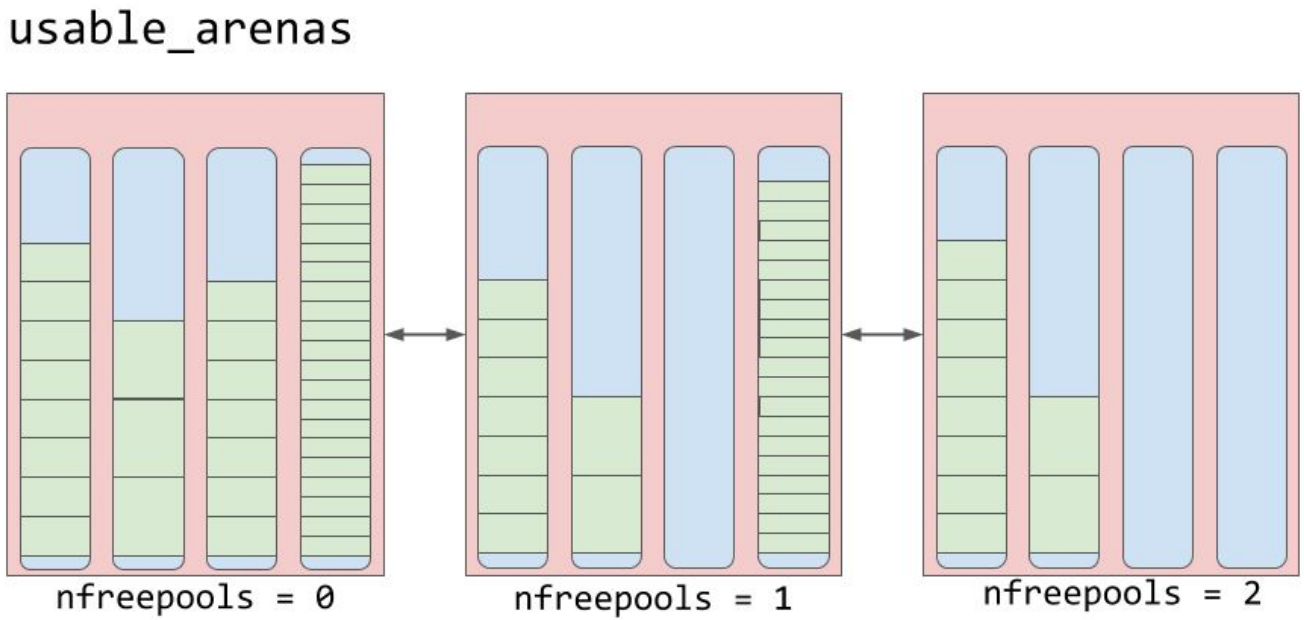
ТЕХНОАТОМ

```
* 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.
```

Управление памятью



Управление памятью



ТЕХНОАТОМ

Спасибо за внимание!

Кандауров Геннадий

g.kandaurov@corp.mail.ru