

Pythonのメモリアロケータ

第23回NaCl社内勉強会

中村 成洋

ネットワーク応用通信研究所

Pythonとは

- 以下略
- 興味があるのはアロケータ

今

目

話

す

の

は

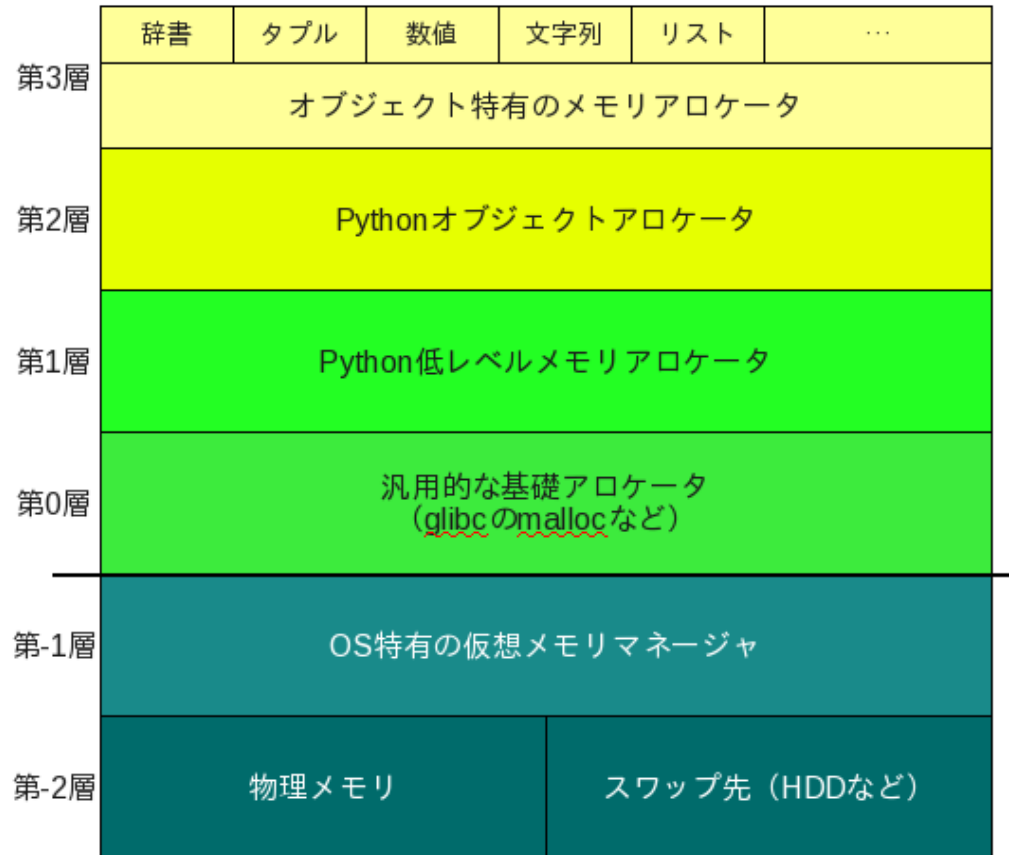
アロケータとオブジェクト生成

```
a = {"a":1, "b":10}
```

どうやってオブジェクトをメモリ割り当てしているかという話

Pythonのメモ
リアロケータは
3層になってい
る

アロケータ層

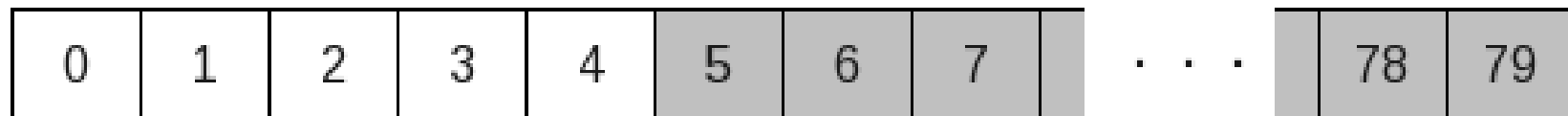


第3層 オブジェクト特有のメモリアロケータ

- オブジェクト固有で持っているメモリアロケータ
- 単純なフリーリストを持っている
 - 80個の配列とか
 - LIFO, スタックのような形でデータを取り扱う

第3層 オブジェクト特有のメモリアロケータ

free_list



使用済みオブジェクトの
ポインタを格納

第3層から第2層へ

- フリーリストが切れたとき

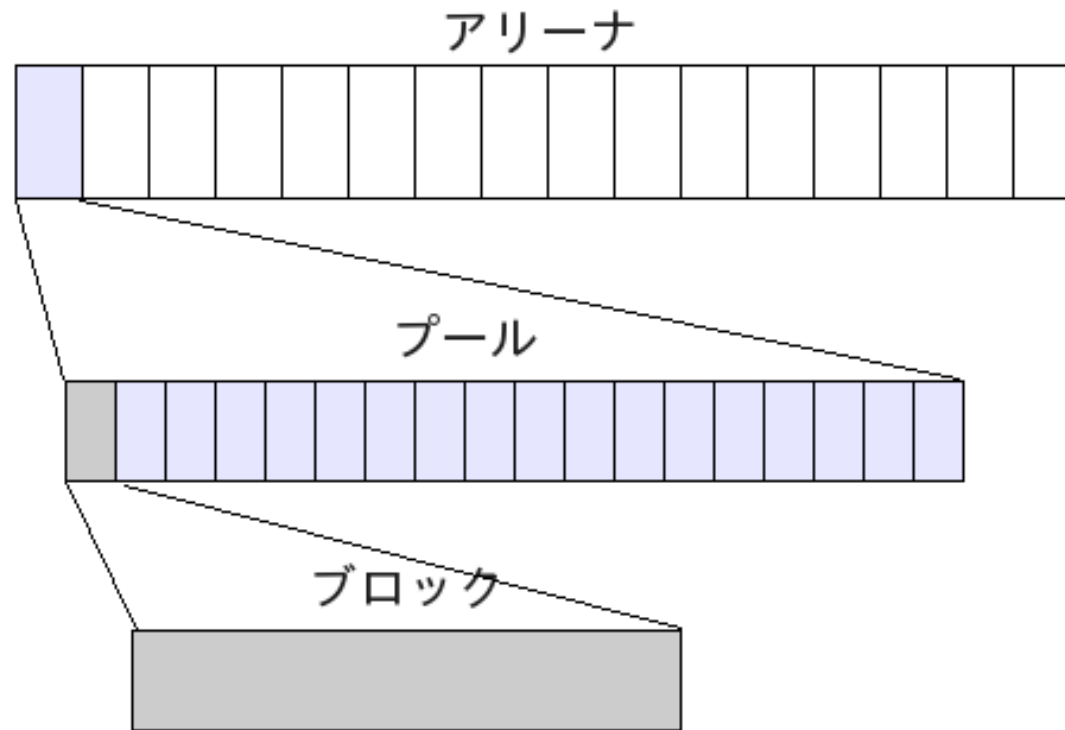
第2層と第1層

- ここがPython内のメモリアロケータの肝
- デカイし, 密に関連し合ってるので一緒に説明
- がばっと確保したメモリを細かくちぎって返すという事をやる

データ構造

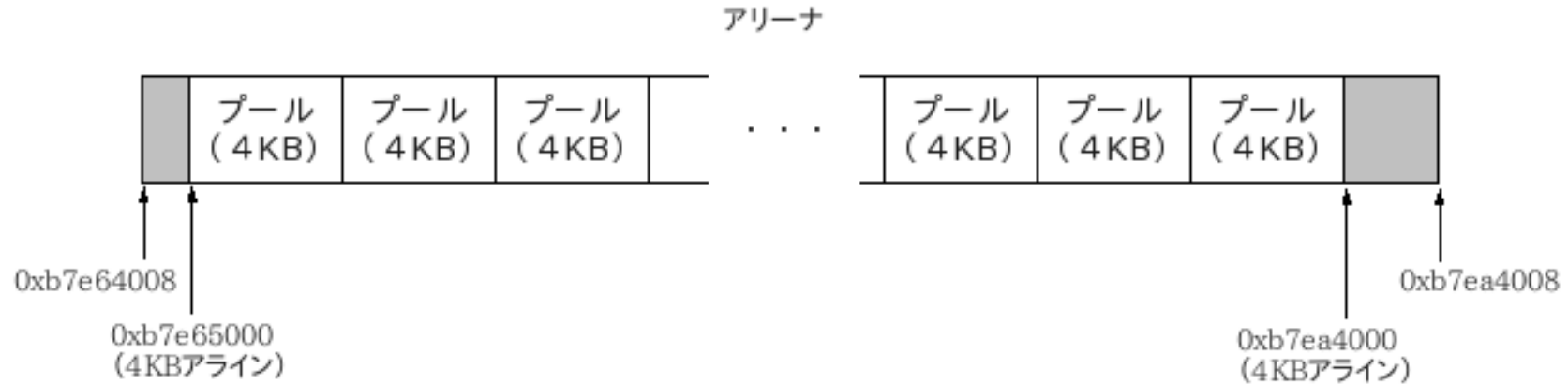
- アリーナ
- プール
- ブロック

データ構造



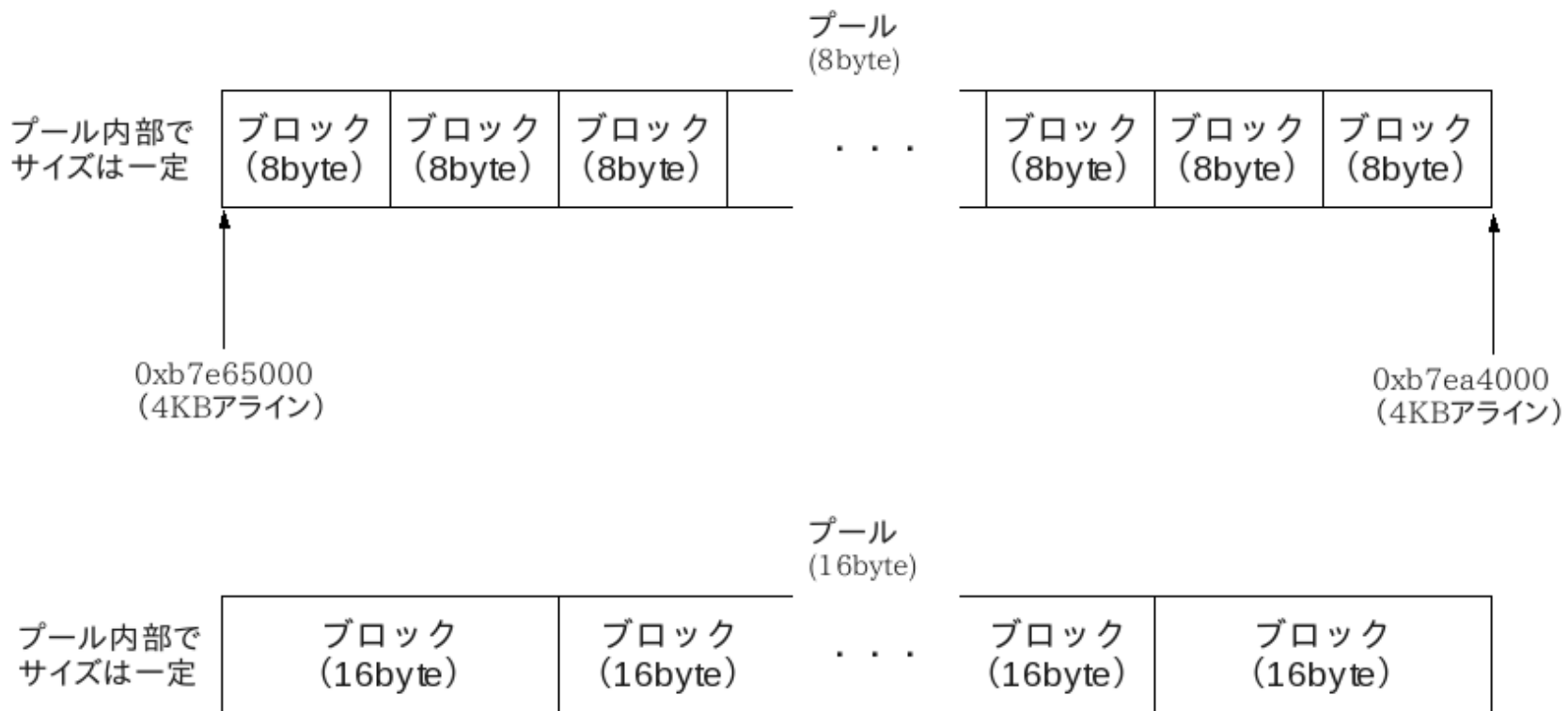
最終的にブロックが返される

アリーナ内のプール



プールは4KBでアラインメントされている(重大な意味があったりする)

プール内のブロック



ブロックはプール内で固定のサイズ
固定するサイズはプールによってまちまち

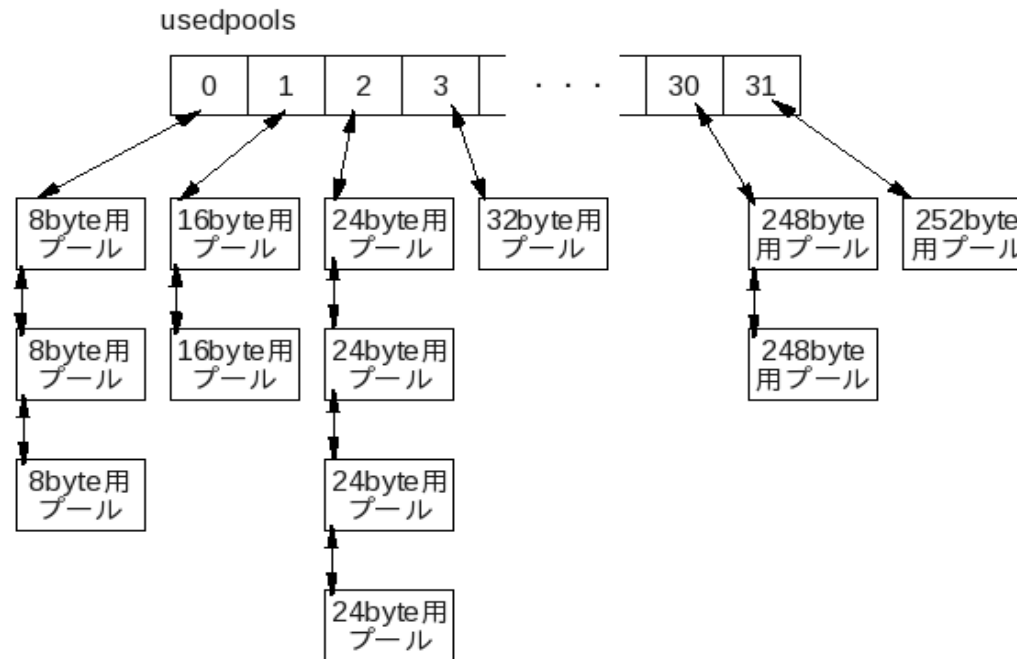
第2層 Pythonオブジェクトアロケーター

- ブロックをプールから取得する部分

要求サイズをブロックサイズへ

- 要求サイズはまちまち
 - 3byte, 10byte とか
- それをブロックのサイズに変換
 - 3byte -> 8byte とか

適したブロックサイズのプールを見つける



- 8byte -> index 0, 16byte -> index 1

取り出ししてきた
プールから
ブロックを返す

第2層から第1層へ

- プール内のブロックが全て割り当て中

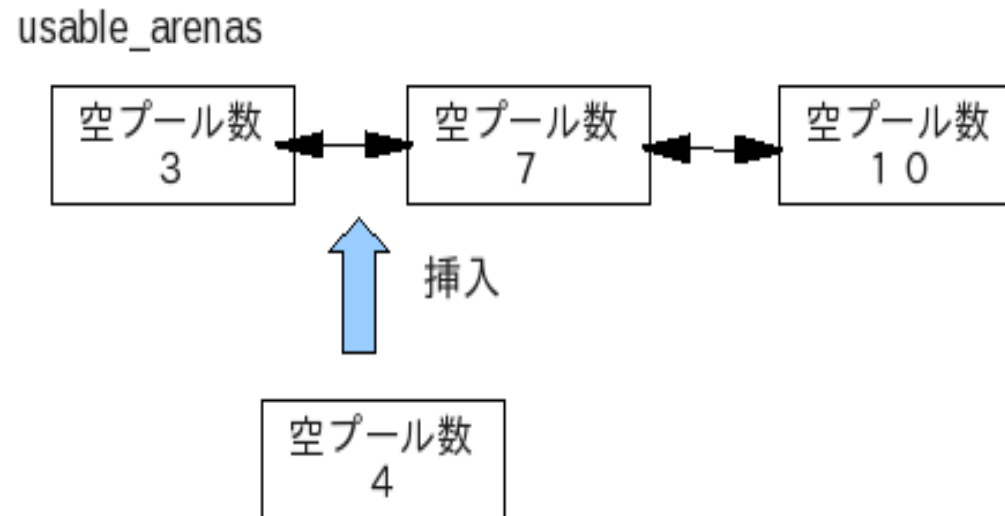
第1層 Python低レベルメモリア ロケータ

- アリーナとプールのメモリ割り当て

アリーナ取り出し

- フリーリストに繋いでおいたアリーナ取得する

フリーリスト



- 一番使われているアリーナが先頭にある
- よりアリーナを解放する為

第1層から第0層へ

- アリーナがなくなったとき

第0層 汎用的な基礎アロケータ

- Cライブラリの malloc()

おまけ

ブロックの解放

- 解放する為には
 - ポインタはブロックのアドレス
 - ブロックからプールを見つける
 - プールからアリーナを見つける

ブロックからプール探索



$0xb7e65724 \ \& \ 0xfffff000 = 0xb7e65000$

マスク処理一回で先頭アドレスが取れる

- マスク処理する
- プールの先頭がとれる