目录

1 介绍

Cpython 是什么

CPython

The canonical implementation of the Python programming language, as distributed on python.org. The term “CPython” is used when necessary to distinguish this implementation from others such as Jython or IronPython.

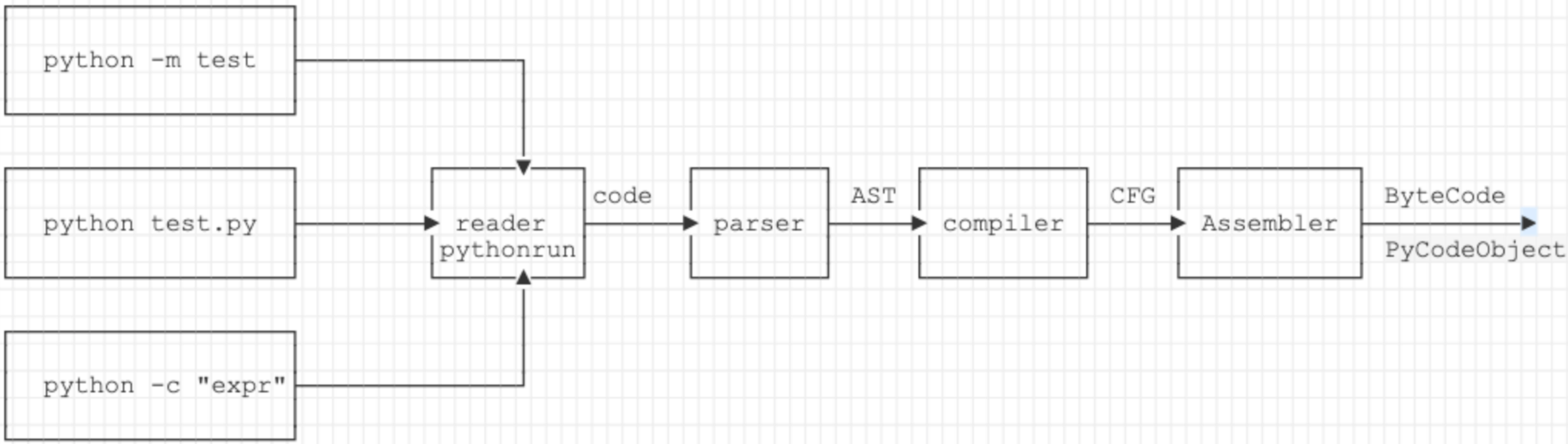
- 对 Python 语言的官方 C 版本实现
- Compiler 和 Interpreter
- REPL

目录结构

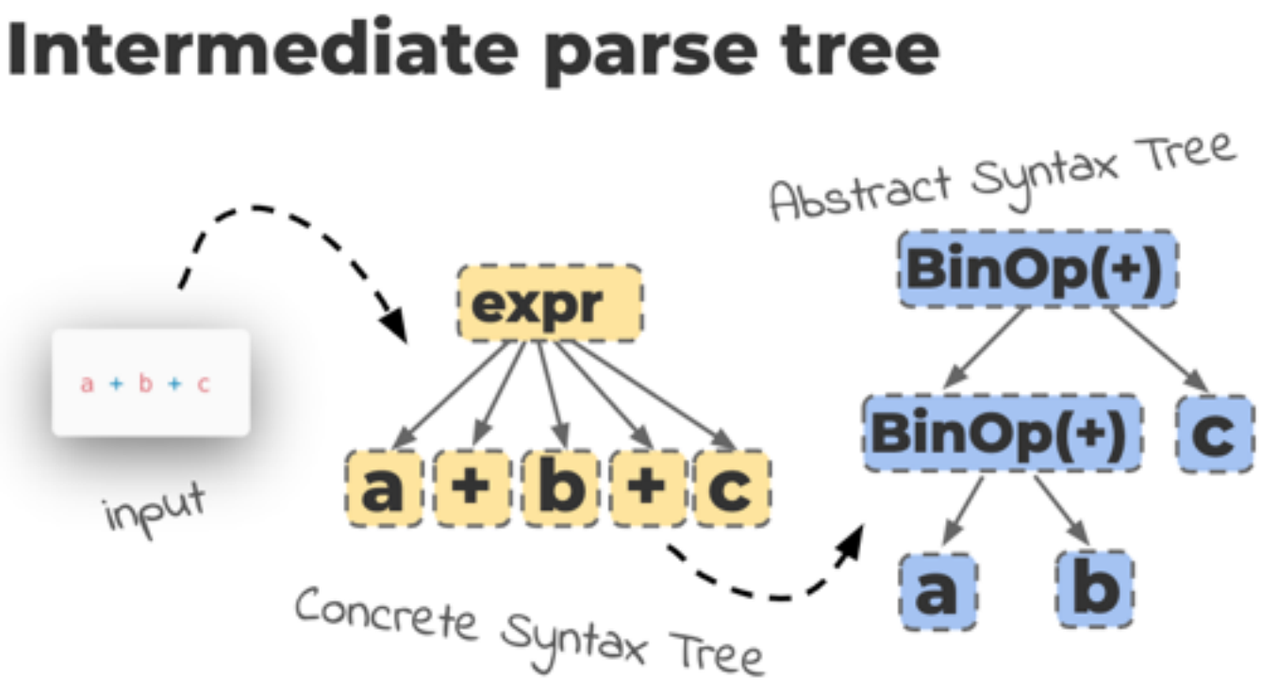
```
cpython/
│
├── Doc      ← Source for the documentation
├── Grammar  ← The computer-readable language definition
├── Include  ← The C header files
├── Lib      ← Standard library modules written in Python
├── Mac      ← macOS support files
├── Misc     ← Miscellaneous files
├── Modules  ← Standard Library Modules written in C
├── Objects  ← Core types and the object model
├── Parser   ← The Python parser source code
├── PC       ← Windows build support files
├── PCbuild  ← Windows build support files for older Windows versions
├── Programs ← Source code for the python executable and other binaries
├── Python   ← The CPython interpreter source code
└── Tools    ← Standalone tools useful for building or extending Python
```

2 代码的执行流程

执行流程图



* pyc 缓存




```
>>> from dis import dis
>>> dis("a**b")
  1           0 LOAD_NAME                0 (a)
              2 LOAD_NAME                1 (b)
              4 BINARY_POWER
              6 RETURN_VALUE
>>>
```

代码行号　偏移量　bytecode　参数

bytecode execution

1、调用run_eval_code_obj() 进入到 evaluation loop （python > ceval.c)

2、准备工作，生成frameobject、tstate等。

3、每个 OP_CODE 都有对应的实现函数，调用它！

```
Switch OP_CODE
case TARGET(BINARY_ADD): {
    PyObject *right = POP();
    PyObject *left = TOP();
    PyObject *sum;
    /* NOTE(vstinner): Please don't try to mic
       CPython using bytecode, it is simply wo
       See http://bugs.python.org/issue21955 a
       http://bugs.python.org/issue10044 for t
       no patch shown any impact on a realistic
       speedup on microbenchmarks. */
    if (PyUnicode_CheckExact(left) &&
                PyUnicode_CheckExact(right)) {…
    else {
        sum = PyNumber_Add(left, right);
        Py_DECREF(left);
```

Stack frame

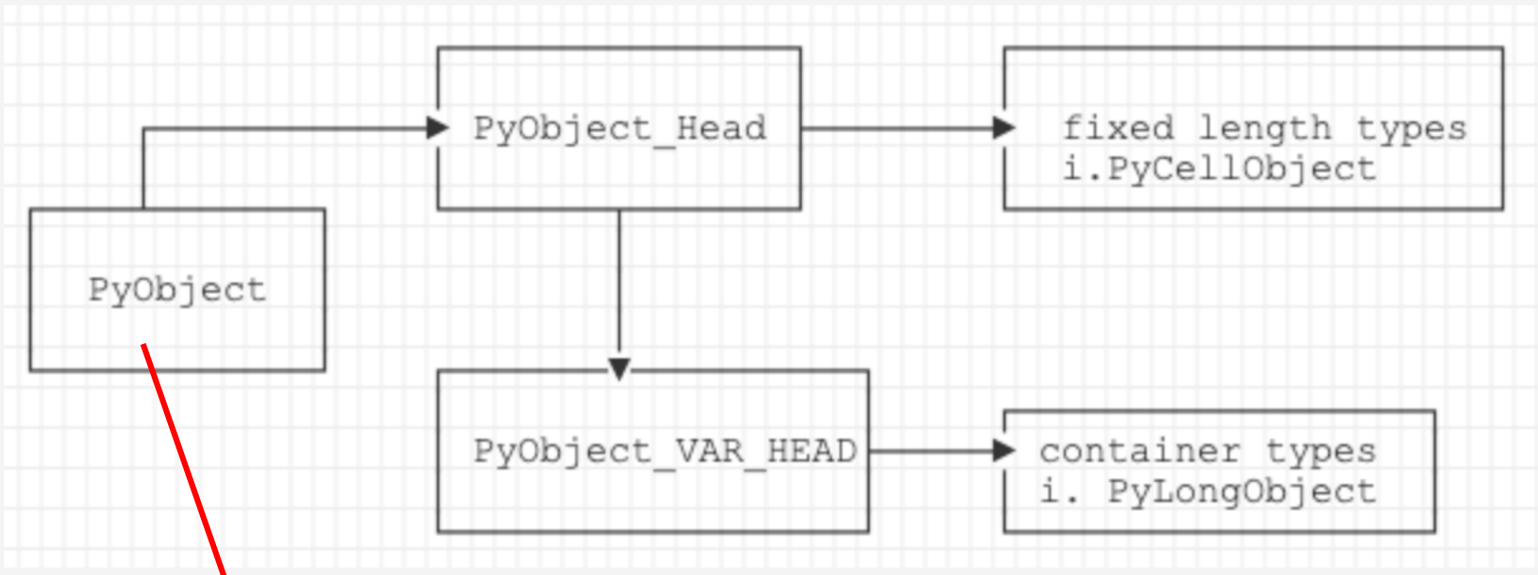
_PyFrame_New_NoTrack() 创建 FrameObject对象

```
f->f_back = (PyFrameObject*)Py_XNewRef(tstate->frame);
f->f_code = (PyCodeObject *)Py_NewRef(con->fc_code);
f->f_builtins = Py_NewRef(con->fc_builtins);
f->f_globals = Py_NewRef(con->fc_globals);
f->f_locals = Py_XNewRef(locals);
// f_valuestack initialized by frame_alloc()
f->f_trace = NULL;
f->f_stackdepth = 0;
f->f_trace_lines = 1;
f->f_trace_opcodes = 0;
f->f_gen = NULL;
f->f_lasti = -1;
f->f_lineno = 0;
f->f_iblock = 0;
f->f_state = FRAME_CREATED;
// f_blockstack and f_localsplus initialized by frame_alloc()
return f;
```

3 数据类型

一切皆为对象

0、一切都是 PyObject的子类(扩展)，都能转成 PyObject *
1、对应 struct 的继承




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

```c
struct _typeobject {
    PyObject_VAR_HEAD
    const char *tp_name; /* For print
    Py_ssize_t tp_basicsize, tp_items

    /* Methods to implement standard

    destructor tp_dealloc;
```

tp_name -> type(x)
tp_hash -> hash()
tp_repr -> print(x)
tp_flags -> 识别类型
…

number 类型

1、int(py2)、long、 float、 complex

2、long的实现方式

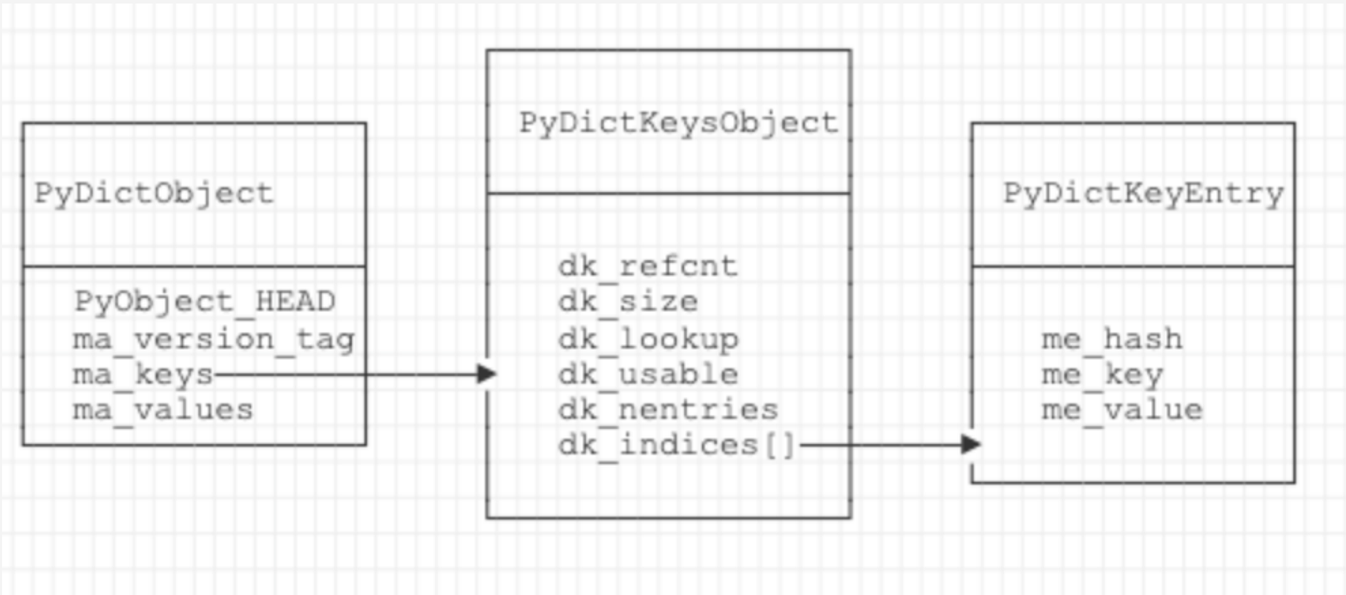
3、不同类型之间的二元操作

```
struct _longobject {
    PyObject_VAR_HEAD
    digit ob_digit[1];
};
```

```
_PyLong_New(Py_ssize_t size)
{
    PyLongObject *result;
    /* Number of bytes needed is: offsetof(PyLongObject, ob_digit) +
       sizeof(digit)*size.  Previous incarnations of this code used
       sizeof(PyVarObject) instead of the offsetof, but this risks being
       incorrect in the presence of padding between the PyVarObject heade
       and the digits. */
    if (size > (Py_ssize_t)MAX_LONG_DIGITS) {
        PyErr_SetString(PyExc_OverflowError,
                        "too many digits in integer");
        return NULL;
    }
    result = PyObject_Malloc(offsetof(PyLongObject, ob_digit) +
                             size*sizeof(digit));
    if (!result) {
```

```
if (!Py_IS_TYPE(v, Py_TYPE(w)) &&
    PyType_IsSubtype(Py_TYPE(w), Py_TYPE(v)) &&
    (f = Py_TYPE(w)->tp_richcompare) != NULL) {
    checked_reverse_op = 1;
    res = (*f)(w, v, _Py_SwappedOp[op]);
    if (res != Py_NotImplemented)
        return res;
    Py_DECREF(res);
}
```

dict 类型

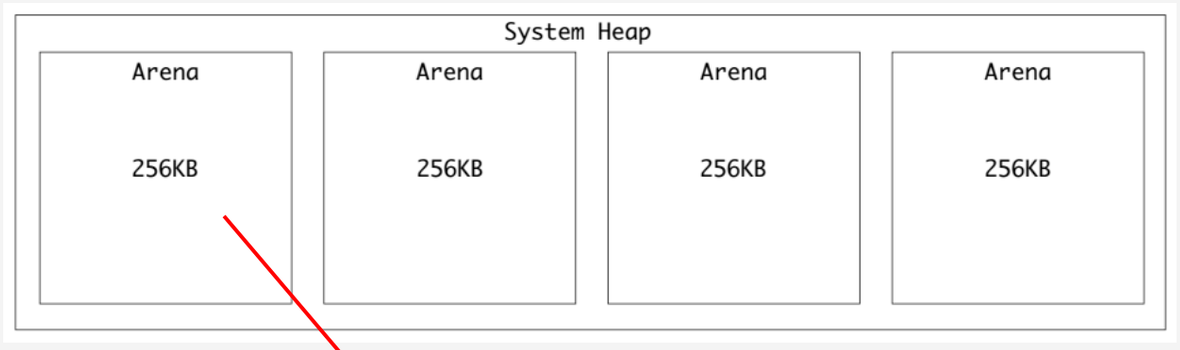


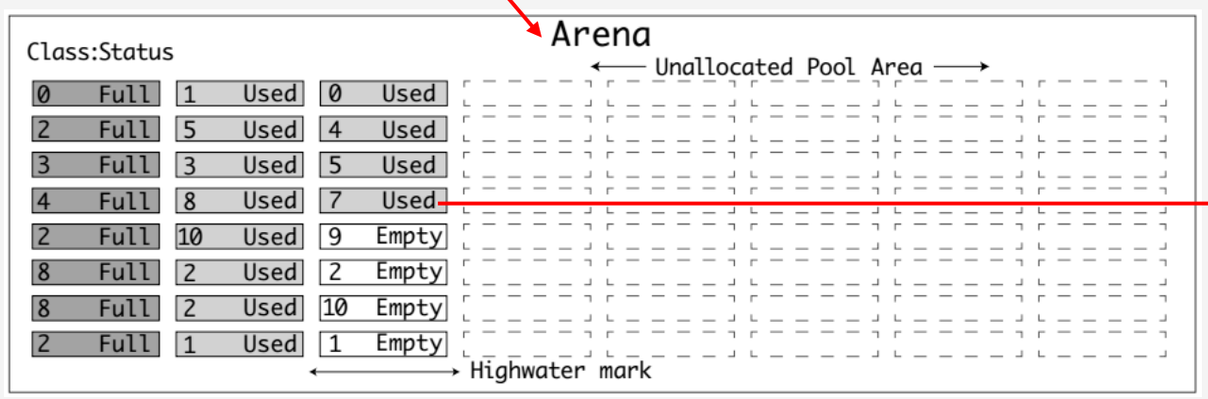
1、 查找
2、**插入**

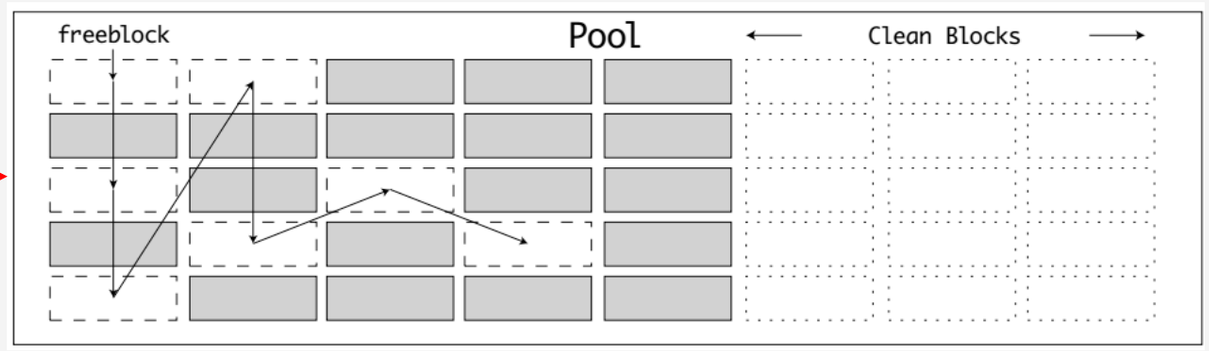
4 内存管理

内存管理

| Raw domain | 申请非Python Object 相关&>256k对象内存,通过 malloc() 申请堆内存
| Object domain | Python Object 相关的内存分配，通过pymalloc() 申请私有堆内存
| PyMem domain | 兼容旧版本api，通过pymalloc() 申请私有堆内存







引用计数

1、每个对象实例都有 ob_refcnt 字段记录被引用的次数。
2、对有新引用时 ob_refcnt++
3、当引用失效时，ob_refcnt --
4、如果 ob_refcnt=0，则会调用对应 dealloc 释放内存

引用的加减由在bytecode的执行中调用Py_INCREF、Py_DECREF

```
case TARGET(LOAD_FAST): {          Benjamin
    PyObject *value = GETLOCAL(oparg);
    if (value == NULL) {
        format_exc_check_arg(tstate, Py
                             UNBOUNDLOC
                             PyTuple_Ge
        goto error;
    }
    Py_INCREF(value);
    PUSH(value);
    FAST_DISPATCH();
}
```

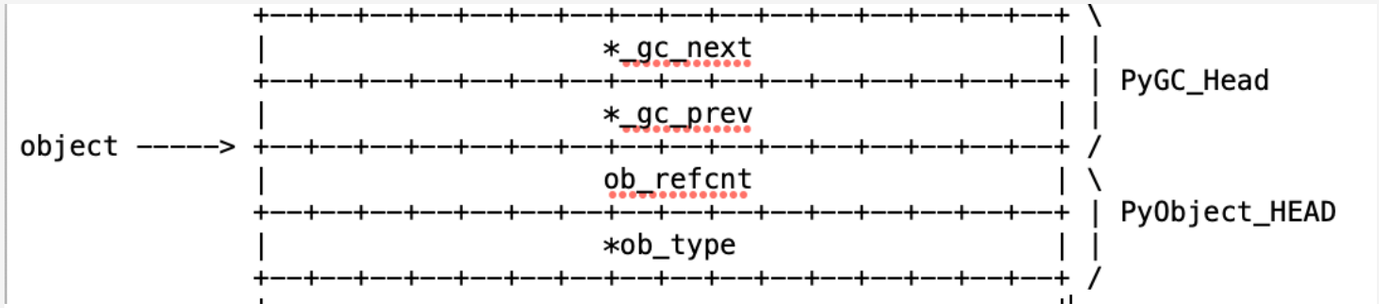
压栈引用+1

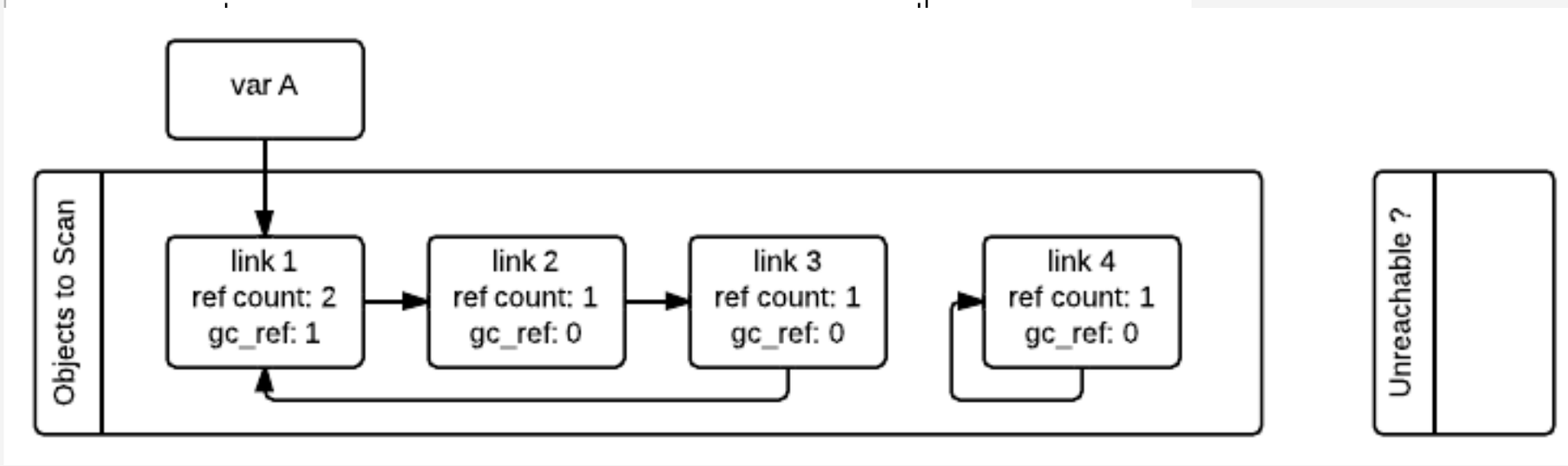
```
case TARGET(BINARY_ADD): {
    PyObject *right = POP();
    PyObject *left = TOP();
    PyObject *sum;      Benjamin Peterson, S
    /* NOTE(haypo): Please don't try to micr
    if (PyUnicode_CheckExact(left) &&
            PyUnicode_CheckExact(right)) {
    else {
        sum = PyNumber_Add(left, right);
        Py_DECREF(left);
    }
    Py_DECREF(right);
    SET_TOP(sum);
    if (sum == NULL)
        goto error;
    DISPATCH();
}
```

gc

> 作为引用计数的补充。

1、只有可变的容器类型才会被gc追踪(Py_TPFLAGS_HAVE_GC)
2、PyGC_Head(和Container Object一起存放)双向链表
3、分代回收 generation 0、1、2
4、清理方式：将refcnt 置为0触发内存释放。





总结

- 分析内部函数的执行 i. dis.dis(xx.pyc)
- 胶水语言

参考链接

https://devguide.python.org/

https://realpython.com/products/cpython-internals-book/

https://pythontutor.com/visualize.html#mode=edit

https://km.woa.com/group/51849/articles/show/486221

Thanks