DGTL

RefCount++

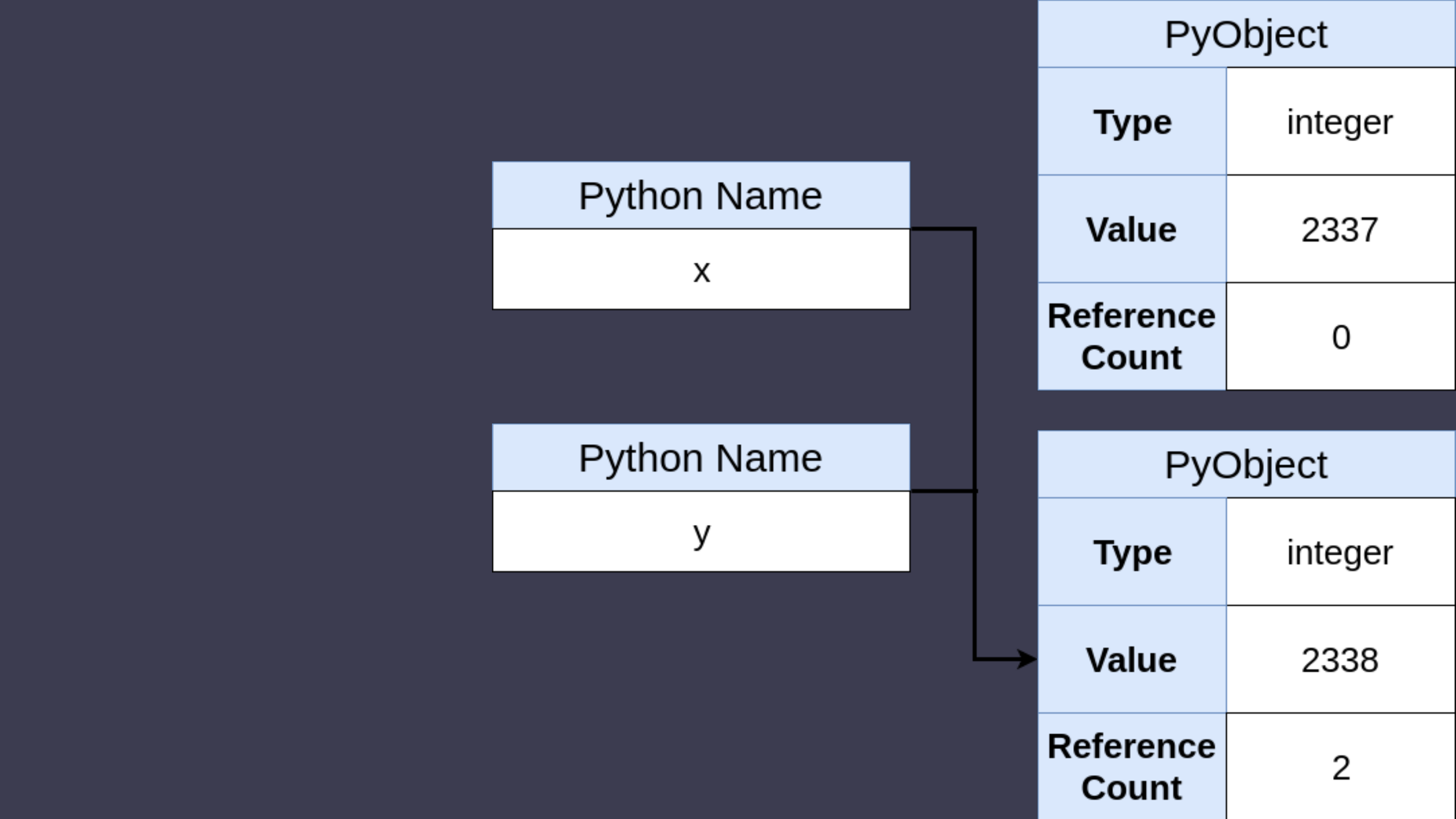
Бейлак Алиев
Expert Software Engineer

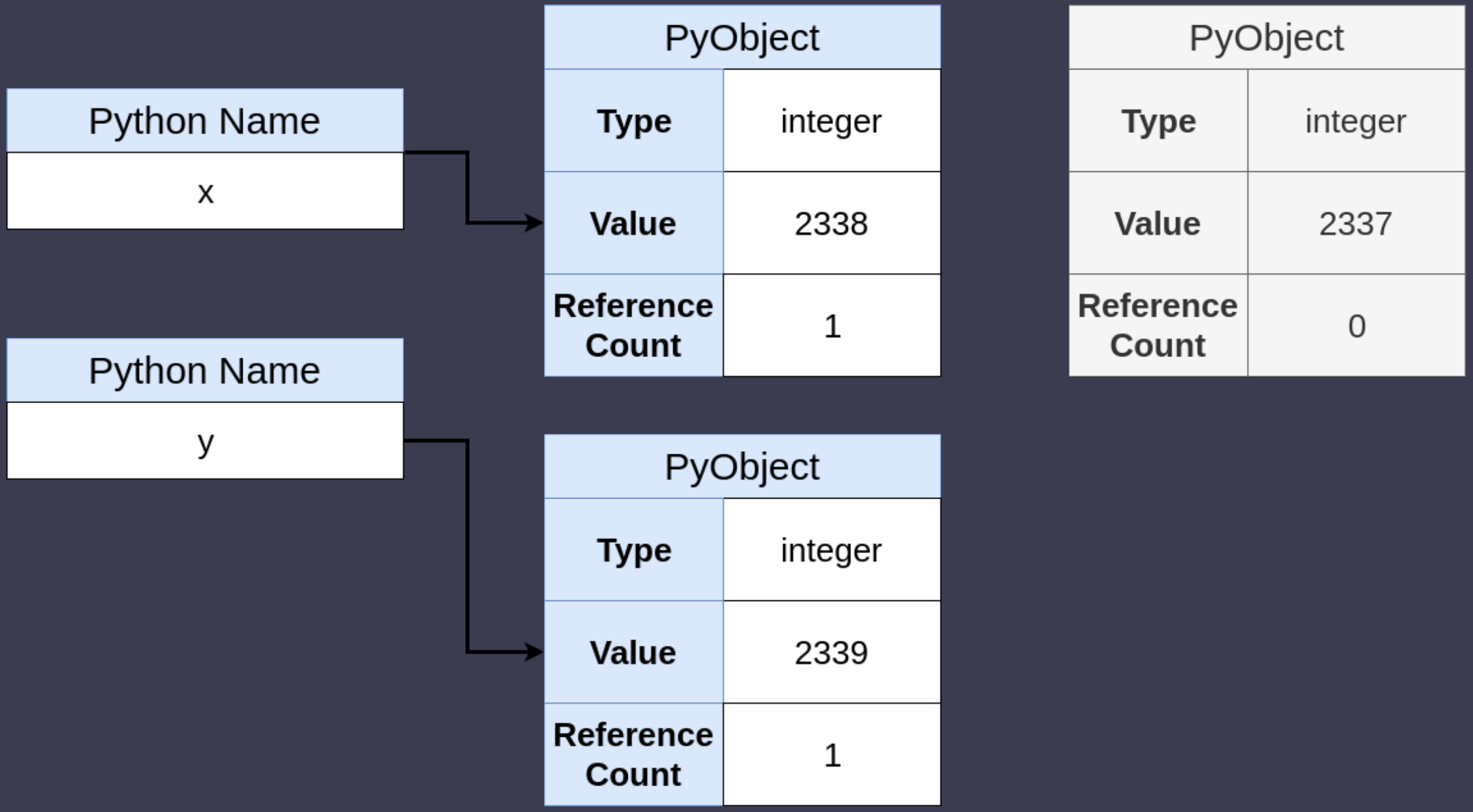
Memory Management

Reference Counting

Garbage Collector

Счетчик ссылок нужен,
потому что железо
неэффективно,
мы все еще не изобрели
бесконечный объем памяти


Python Name
x
Python Name
y
PyObject
Type
integer
Value
2337
Reference Count
0
PyObject
Type
integer
Value
2338
Reference Count
2


Python Name
x
PyObject
Type
integer
Value
2338
Reference Count
1
PyObject
Type
integer
Value
2337
Reference Count
0
Python Name
y
PyObject
Type
integer
Value
2339
Reference Count
1

RefCount macros

Py_INCREF

Py_DECREF

Py_INCREF

```
625 static inline Py_ALWAYS_INLINE void Py_INCREF(PyObject *op)
  1 {
  2 #if defined(Py_LIMITED_API) && (Py_LIMITED_API+0 >= 0x030c0000 || defined(Py_REF_DEBUG))
  3     // Stable ABI implements Py_INCREF() as a function call on limited C API
  4     // version 3.12 and newer, and on Python built in debug mode. _Py_IncRef()
  5     // was added to Python 3.10.0a7, use Py_IncRef() on older Python versions.
  6     // Py_IncRef() accepts NULL whereas _Py_IncRef() doesn't.
  7 #  if Py_LIMITED_API+0 >= 0x030a00A7
  8     _Py_IncRef(op);
  9 #  else
 10     Py_IncRef(op);
 11 #  endif
 12 #else
 13     // Non-limited C API and limited C API for Python 3.9 and older access
 14     // directly PyObject.ob_refcnt.
 15 #if SIZEOF_VOID_P > 4
 16     // Portable saturated add, branching on the carry flag and set low bits
 17     PY_UINT32_T cur_refcnt = op->ob_refcnt_split[PY_BIG_ENDIAN];
 18     PY_UINT32_T new_refcnt = cur_refcnt + 1;
 19     if (new_refcnt == 0) {
 20         return;
 21     }
 22     op->ob_refcnt_split[PY_BIG_ENDIAN] = new_refcnt;
 23 #else
 24     // Explicitly check immortality against the immortal value
 25     if (_Py_IsImmortal(op)) {
 26         return;
 27     }
 28     op->ob_refcnt++;
 29 #endif
 30     _Py_INCREF_STAT_INC();
 31 #ifdef Py_REF_DEBUG
 32     _Py_INCREF_IncRefTotal();
 33 #endif
 34 #endif
 35 }
```

Py_DECREF

```
#define Py_DECREF(op) Py_DECREF(_PyObject_CAST(op))

#elif defined(Py_REF_DEBUG)
static inline void Py_DECREF(const char *filename, int lineno, PyObject *op)
{
    if (op->ob_refcnt <= 0) {
        _Py_NegativeRefcount(filename, lineno, op);
    }
    if (_Py_IsImmortal(op)) {
        return;
    }
    _Py_DECREF_STAT_INC();
    _Py_DECREF_DecRefTotal();
    if (--op->ob_refcnt == 0) {
        _Py_Dealloc(op);
    }
}
#define Py_DECREF(op) Py_DECREF(__FILE__, __LINE__, _PyObject_CAST(op))

#else
static inline Py_ALWAYS_INLINE void Py_DECREF(PyObject *op)
{
    // Non-limited C API and limited C API for Python 3.9 and older access
    // directly PyObject.ob_refcnt.
    if (_Py_IsImmortal(op)) {
        return;
    }
    _Py_DECREF_STAT_INC();
    if (--op->ob_refcnt == 0) {
        _Py_Dealloc(op);
    }
}
#define Py_DECREF(op) Py_DECREF(_PyObject_CAST(op))
#endif
```

Подготовка

```python
import ctypes

ref = ctypes.c_long.from_address


class A:
    def __init__(self) -> None:
        self.my_obj_attr: str = "This is kakoyto text"

```

```python
def exp_0() -> None:

    a = A()
    a_id = id(a)

    ref_a = ref(a_id).value
    print(f"1 - obj a. Obj id {a_id = }")
    print(f"2 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    b = a

    ref_a = ref(a_id).value
    print(f"3 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    del b
    ref_a = ref(a_id).value
    print(f"4 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    ref_a = ref(a_id).value
    print(f"5 - a. Ref count {ref_a = }")
    raw_id_value = ctypes.cast(a_id, ctypes.py_object).value
    print(f"Obj a. Value from id {raw_id_value = }")
    print(raw_id_value.my_obj_attr)
    print("*" * 10)
```

```
(memory-test-py3.12) ➜  memory_test python main.py
1 - obj a. Obj id a_id = 4302858208
2 - obj a. Ref count ref_a = 1
**********
3 - obj a. Ref count ref_a = 2
**********
4 - obj a. Ref count ref_a = 1
**********
5 - a. Ref count ref_a = 1
Obj a. Value from id raw_id_value = <__main__.A object at 0x1007867e0>
This is kakoyto text
**********
```

```python
def exp_0() -> None:

    a = A()
    a_id = id(a)

    ref_a = ref(a_id).value
    print(f"1 - obj a. Obj id {a_id = }")
    print(f"2 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    b = a

    ref_a = ref(a_id).value
    print(f"3 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    del b
    ref_a = ref(a_id).value
    print(f"4 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    ref_a = ref(a_id).value
    print(f"5 - a. Ref count {ref_a = }")
    raw_id_value = ctypes.cast(a_id, ctypes.py_object).value
    print(f"Obj a. Value from id {raw_id_value = }")
    print(raw_id_value.my_obj_attr)
    print("*" * 10)

    ref_a = ref(a_id).value
    print(f"6 - a. Ref count {ref_a = }")
    print("*" * 10)

    del raw_id_value
    del a
    ref_a = ref(a_id).value
    print(f"7 - a. Ref count {ref_a = }")
    print("*" * 10)
```



```
(memory-test-py3.12) ➜  memory_test python main.py
1 - obj a. Obj id a_id = 4302858208
2 - obj a. Ref count ref_a = 1
**********
3 - obj a. Ref count ref_a = 2
**********
4 - obj a. Ref count ref_a = 1
**********
5 - a. Ref count ref_a = 1
Obj a. Value from id raw_id_value = <__main__.A object at 0x1007867e0>
This is kakoyto text
**********
6 - a. Ref count ref_a = 2
**********
7 - a. Ref count ref_a = 0
**********
```

```python
def exp_0() -> None:
    gc.disable()

    a = A()
    a_id = id(a)

    ref_a = ref(a_id).value
    print(f"1 - obj a. Obj id {a_id = }")
    print(f"2 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    b = a

    ref_a = ref(a_id).value
    print(f"3 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    del b
    ref_a = ref(a_id).value
    print(f"4 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    ref_a = ref(a_id).value
    print(f"5 - a. Ref count {ref_a = }")
    raw_id_value = ctypes.cast(a_id, ctypes.py_object).value
    print(f"Obj a. Value from id {raw_id_value = }")
    print(raw_id_value.my_obj_attr)
    print("*" * 10)

    ref_a = ref(a_id).value
    print(f"6 - a. Ref count {ref_a = }")
    print("*" * 10)

    del raw_id_value
    del a
    ref_a = ref(a_id).value
    print(f"7 - a. Ref count {ref_a = }")
    print("*" * 10)
```

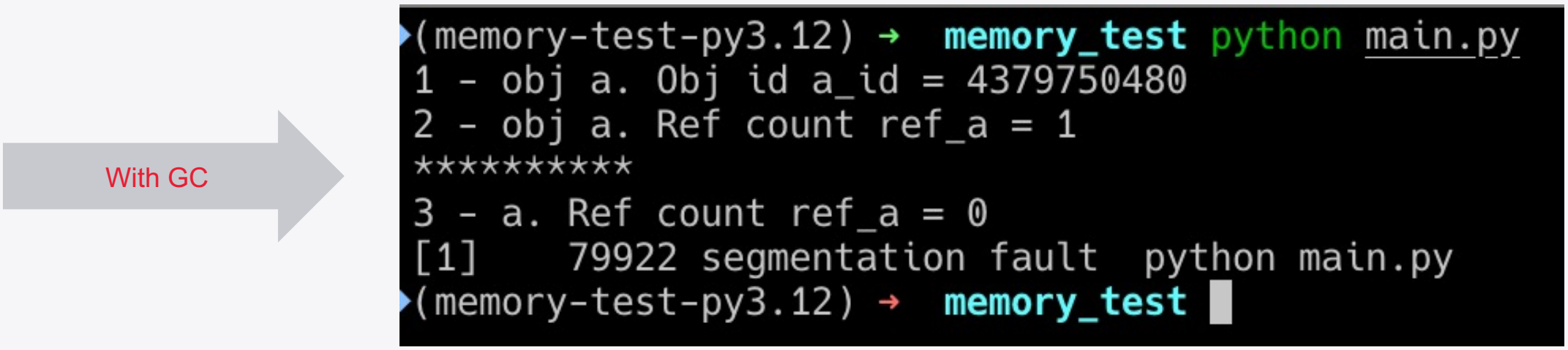
```
(memory-test-py3.12) ➜  memory_test python main.py
1 - obj a. Obj id a_id = 4302858208
2 - obj a. Ref count ref_a = 1
**********
3 - obj a. Ref count ref_a = 2
**********
4 - obj a. Ref count ref_a = 1
**********
5 - a. Ref count ref_a = 1
Obj a. Value from id raw_id_value = <__main__.A object at 0x1007867e0>
This is kakoyto text
**********
6 - a. Ref count ref_a = 2
**********
7 - a. Ref count ref_a = 0
**********
```

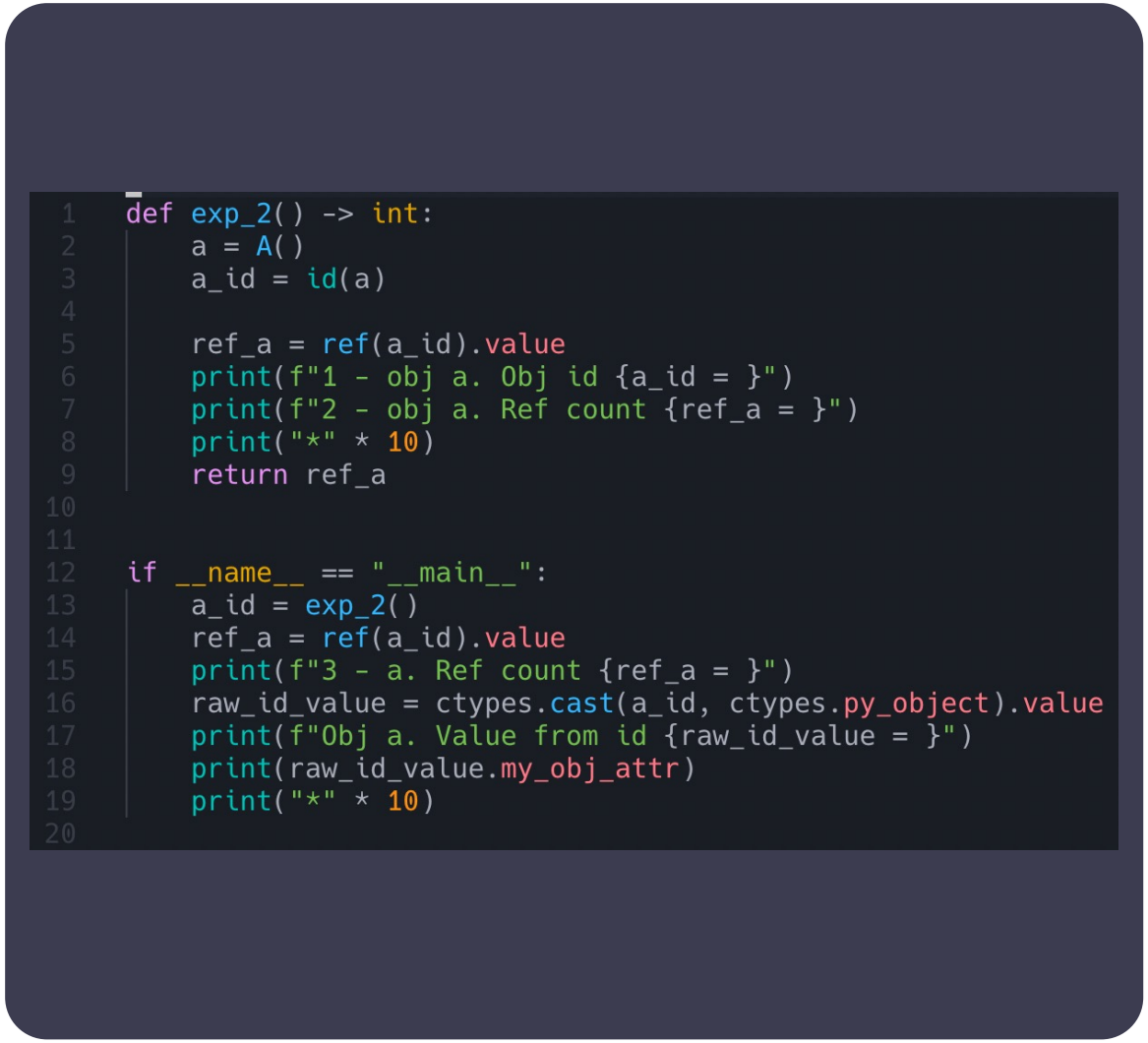
```
(memory-test-py3.12) ➜  memory_test python main.py
1 - obj a. Obj id a_id = 4311607264
2 - obj a. Ref count ref_a = 1
**********
3 - obj a. Ref count ref_a = 2
**********
4 - obj a. Ref count ref_a = 1
**********
5 - a. Ref count ref_a = 1
Obj a. Value from id raw_id_value = <__main__.A object at 0x100fde7e0>
This is kakoyto text
**********
6 - a. Ref count ref_a = 2
**********
7 - a. Ref count ref_a = 0
**********
```

```python
def exp_1() -> None:
    a = A()
    a_id = id(a)

    ref_a = ref(a_id).value
    print(f"1 - obj a. Obj id {a_id = }")
    print(f"2 - obj a. Ref count {ref_a = }")
    print("*" * 10)

    del a
    ref_a = ref(a_id).value
    print(f"3 - a. Ref count {ref_a = }")
    raw_id_value = ctypes.cast(a_id, ctypes.py_object).value
    print(f"Obj a. Value from id {raw_id_value = }")
    print(raw_id_value.my_obj_attr)
    print("*" * 10)


if __name__ == "__main__":
    a_id = exp_1()
```

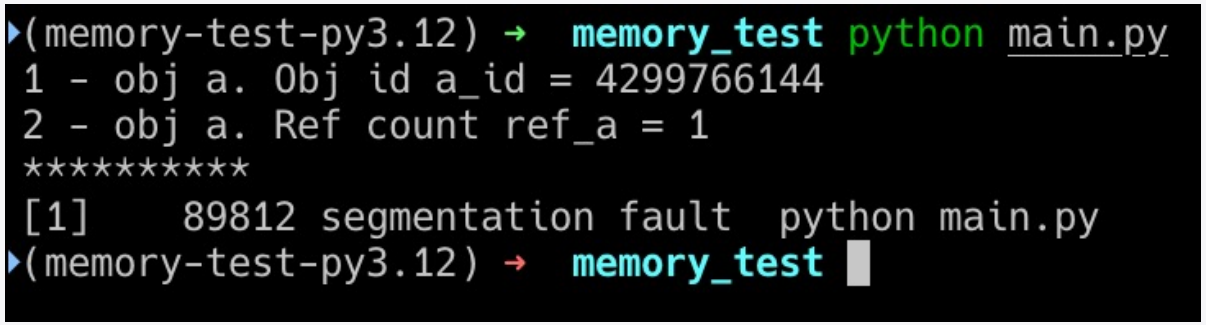
```
(memory-test-py3.12) ➜  memory_test python main.py
1 - obj a. Obj id a_id = 4379750480
2 - obj a. Ref count ref_a = 1
**********
3 - a. Ref count ref_a = 0
[1]    79922 segmentation fault  python main.py
(memory-test-py3.12) ➜  memory_test
```


(memory-test-py3.12) → memory_test python main.py
1 - obj a. Obj id a_id = 4300501216
2 - obj a. Ref count ref_a = 1
****
3 - a. Ref count ref_a = 0
[1] 80685 segmentation fault python main.py
(memory-test-py3.12) → memory_test
No GC


With GC
(memory-test-py3.12) → memory_test python main.py
1 - obj a. Obj id a_id = 4379750480
2 - obj a. Ref count ref_a = 1
****
3 - a. Ref count ref_a = 0
[1] 79922 segmentation fault python main.py
(memory-test-py3.12) → memory_test



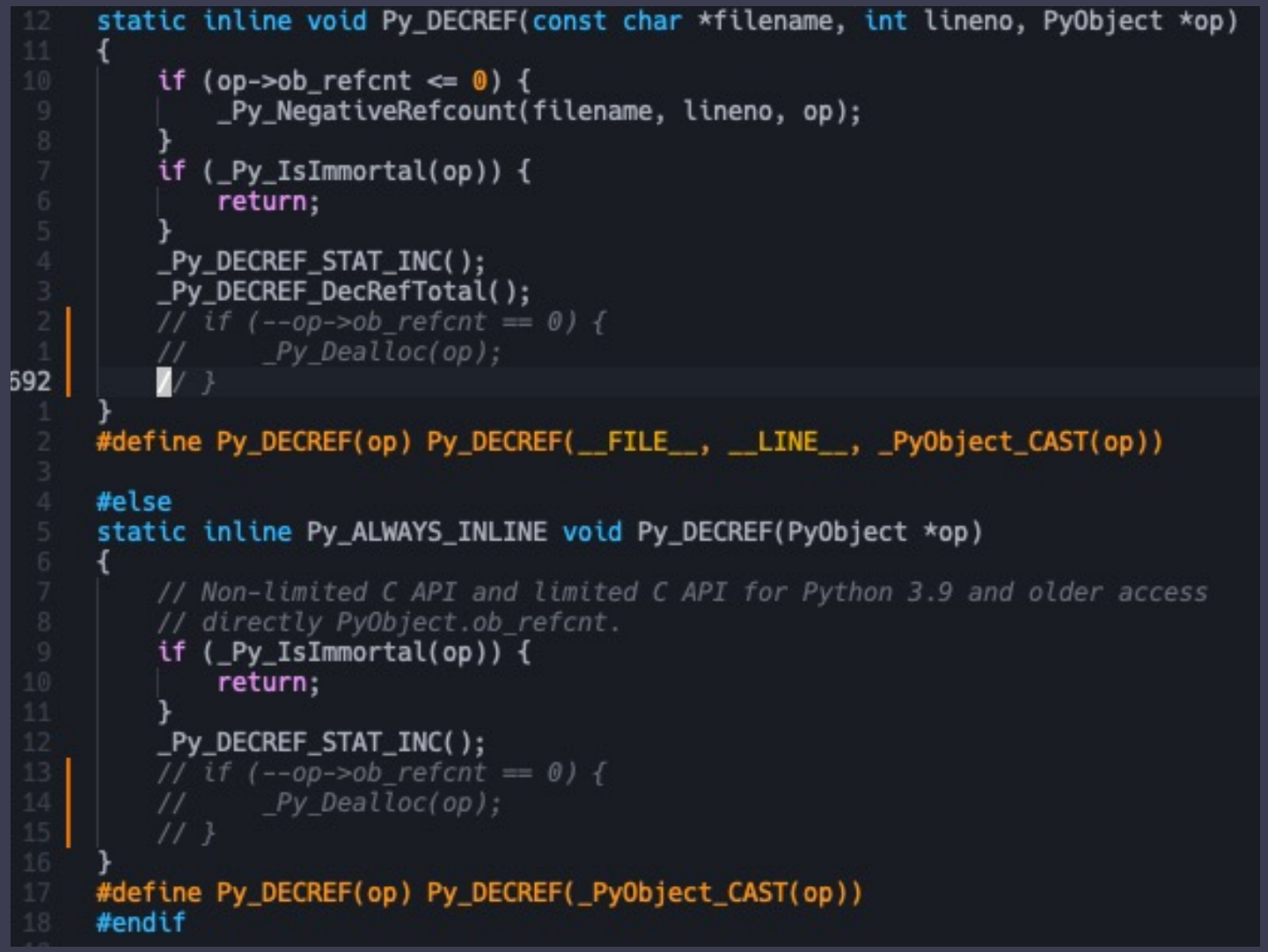
```python
def exp_2() -> int:
    a = A()
    a_id = id(a)

    ref_a = ref(a_id).value
    print(f"1 - obj a. Obj id {a_id = }")
    print(f"2 - obj a. Ref count {ref_a = }")
    print("*" * 10)
    return ref_a


if __name__ == "__main__":
    a_id = exp_2()
    ref_a = ref(a_id).value
    print(f"3 - a. Ref count {ref_a = }")
    raw_id_value = ctypes.cast(a_id, ctypes.py_object).value
    print(f"Obj a. Value from id {raw_id_value = }")
    print(raw_id_value.my_obj_attr)
    print("*" * 10)

```



```
(memory-test-py3.12) ➜  memory_test python main.py
1 - obj a. Obj id a_id = 4299766144
2 - obj a. Ref count ref_a = 1
**********
[1]    89812 segmentation fault  python main.py
(memory-test-py3.12) ➜  memory_test
```

No RefCount

No RefCount



```
12  static inline void Py_DECREF(const char *filename, int lineno, PyObject *op)
11  {
10      if (op->ob_refcnt <= 0) {
9           _Py_NegativeRefcount(filename, lineno, op);
8       }
7       if (_Py_IsImmortal(op)) {
6           return;
5       }
4       _Py_DECREF_STAT_INC();
3       _Py_DECREF_DecRefTotal();
2       // if (--op->ob_refcnt == 0) {
1       //     _Py_Dealloc(op);
592     // }
1   }
2   #define Py_DECREF(op) Py_DECREF(__FILE__, __LINE__, _PyObject_CAST(op))
3
4   #else
5   static inline Py_ALWAYS_INLINE void Py_DECREF(PyObject *op)
6   {
7       // Non-limited C API and limited C API for Python 3.9 and older access
8       // directly PyObject.ob_refcnt.
9       if (_Py_IsImmortal(op)) {
10          return;
11      }
12      _Py_DECREF_STAT_INC();
13      // if (--op->ob_refcnt == 0) {
14      //     _Py_Dealloc(op);
15      // }
16  }
17  #define Py_DECREF(op) Py_DECREF(_PyObject_CAST(op))
18  #endif
```

No RefCount

```python
def _exp_3() -> int:
    a = A()
    a_id = id(a)

    ref_a = ref(a_id).value
    print(f"1 - obj a. Ref count {ref_a = }")
    b = a
    ref_a = ref(a_id).value
    print(f"2 - obj a. Ref count {ref_a = }")
    del b
    ref_a = ref(a_id).value
    print(f"3 - obj a. Ref count {ref_a = }")
    print("*" * 10)
    return a_id


def exp_3() -> None:
    a_id = _exp_3()
    ref_a = ref(a_id).value
    print(f"4 - obj a. Ref count {ref_a = }")
    raw_id_value = ctypes.cast(a_id, ctypes.py_object).value
    print(f"Obj a. Value from id {raw_id_value = }")
    print(raw_id_value.my_obj_attr)
    print("*" * 10)
```

```
1 - obj a. Ref count ref_a = 4
2 - obj a. Ref count ref_a = 5
3 - obj a. Ref count ref_a = 5
**********
4 - obj a. Ref count ref_a = 5
Obj a. Value from id raw_id_value = <__main__.A object at 0x100af95b0>
['This is kakoyto text']
**********
```

```
→  cpython git:(v3.12.1) ✗ ./python.exe /Users/Shared/demo/memory_test/main.py
1 - obj a. Ref count ref_a = 1
2 - obj a. Ref count ref_a = 2
3 - obj a. Ref count ref_a = 1
**********
4 - obj a. Ref count ref_a = 0
Obj a. Value from id raw_id_value = <__main__.A object at 0x10587cb60>
['This is kakoyto text']
**********
→  cpython git:(v3.12.1) ✗
```

No _Py_Dealloc

```
→  cpython git:(v3.12.1) ✗ ./python.exe /Users/Shared/demo/memory_test/main.py
1 - obj a. Ref count ref_a = 1
2 - obj a. Ref count ref_a = 2
3 - obj a. Ref count ref_a = 1
**********
4 - obj a. Ref count ref_a = 0
Obj a. Value from id raw_id_value = <__main__.A object at 0x1030ed7c0>
['This is kakoyto text']
**********
```

No GC

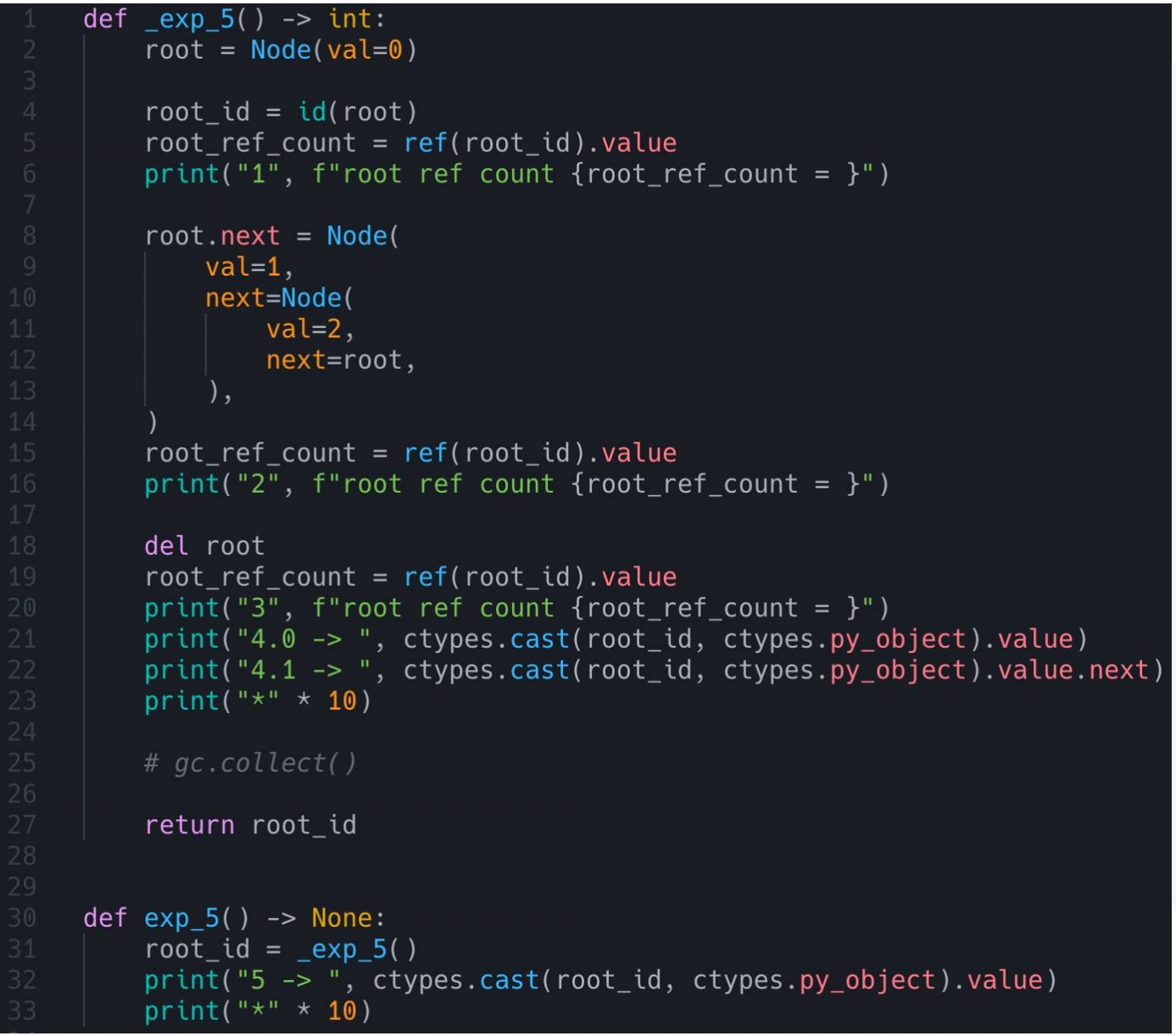
Python 3.12

```
(memory-test-py3.12) →  memory_test python main.py
1 - obj a. Ref count ref_a = 1
2 - obj a. Ref count ref_a = 2
3 - obj a. Ref count ref_a = 1
**********
4 - obj a. Ref count ref_a = 0
[1]    55170 segmentation fault  python main.py
(memory-test-py3.12) →  memory_test
```

GC и Linked List

```python
class Node:
    def __init__(self, val: int, next: Self | None = None) -> None:
        self.val: int = val
        self.next: Self | None = next

    def __str__(self) -> str:
        return f"Node val is {self.val}"

```

GC и Linked List



```python
def _exp_5() -> int:
    root = Node(val=0)

    root_id = id(root)
    root_ref_count = ref(root_id).value
    print("1", f"root ref count {root_ref_count = }")

    root.next = Node(
        val=1,
        next=Node(
            val=2,
            next=root,
        ),
    )
    root_ref_count = ref(root_id).value
    print("2", f"root ref count {root_ref_count = }")

    del root
    root_ref_count = ref(root_id).value
    print("3", f"root ref count {root_ref_count = }")
    print("4.0 -> ", ctypes.cast(root_id, ctypes.py_object).value)
    print("4.1 -> ", ctypes.cast(root_id, ctypes.py_object).value.next)
    print("*" * 10)

    # gc.collect()

    return root_id


def exp_5() -> None:
    root_id = _exp_5()
    print("5 -> ", ctypes.cast(root_id, ctypes.py_object).value)
    print("*" * 10)
```

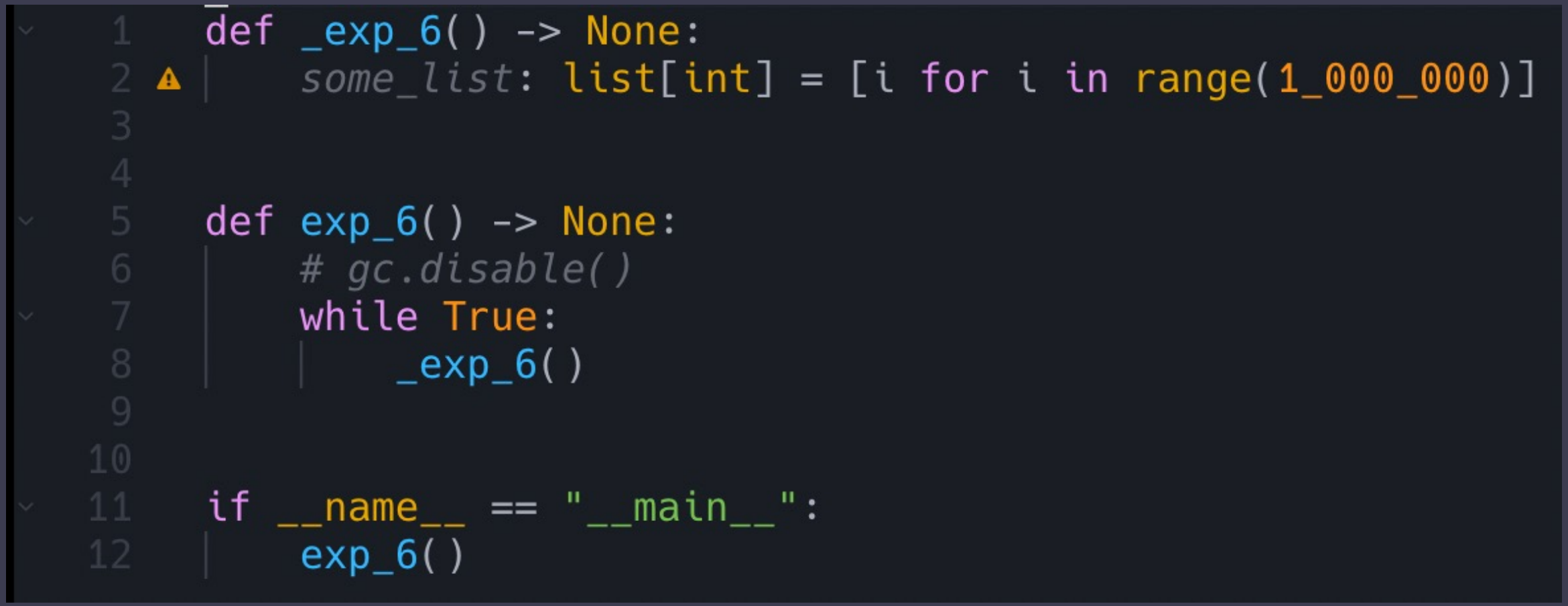
```
(memory-test-py3.12) ➜  memory_test python main.py
1 root ref count root_ref_count = 1
2 root ref count root_ref_count = 2
3 root ref count root_ref_count = 1
4.0 ->  Node val is 0
4.1 ->  Node val is 1
**********
5 ->  Node val is 0
**********
```

GC и Linked List

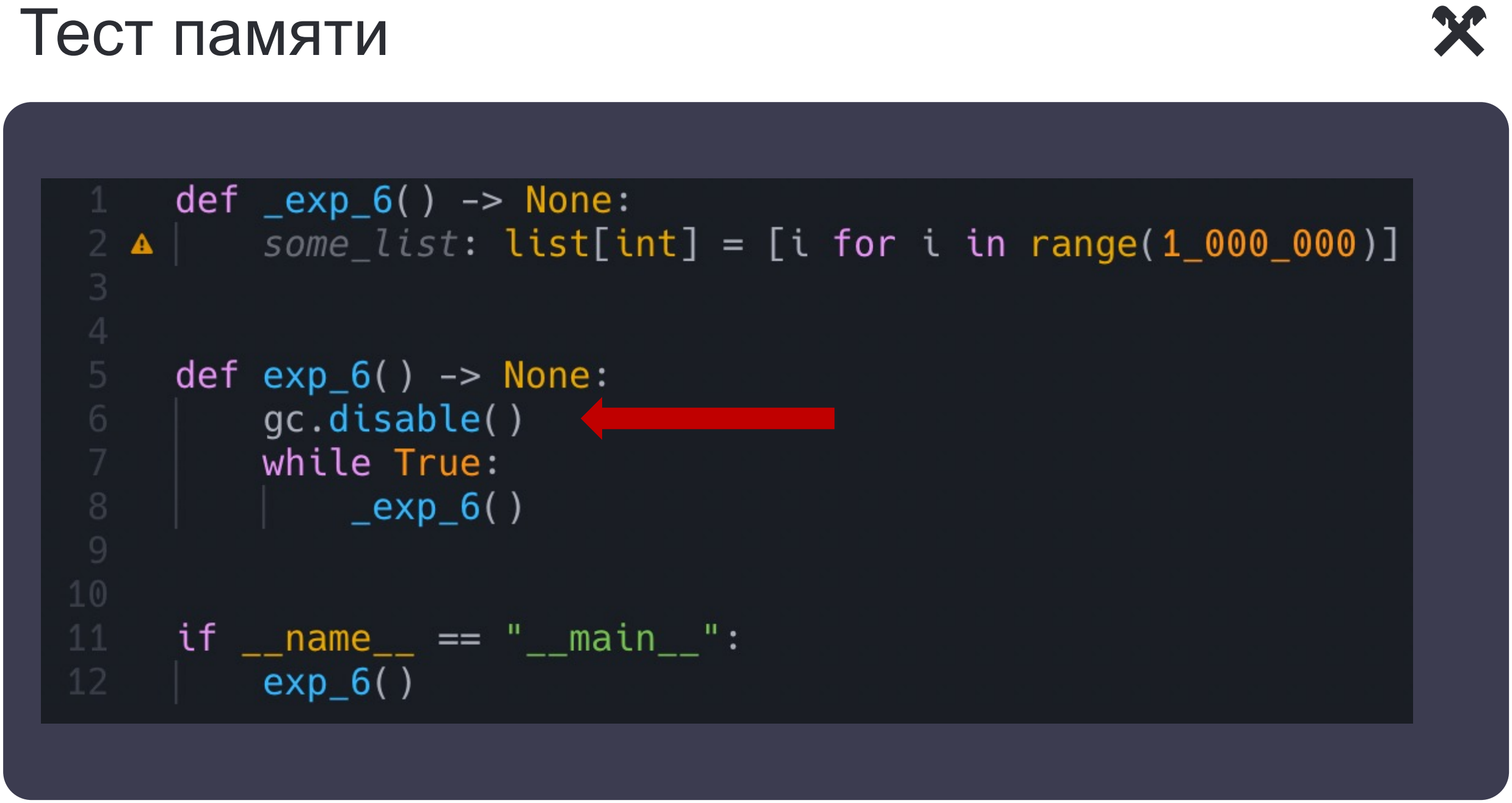

```
    gc.collect()

    return root_id


def exp_5() -> None:
    root_id = _exp_5()
    print("5 -> ", ctypes.cast(root_id, ctypes.py_object).value)
    print("*" * 10)

```

```
(memory-test-py3.12) ➜  memory_test python main.py
1 root ref count root_ref_count = 1
2 root ref count root_ref_count = 2
3 root ref count root_ref_count = 1
4.0 ->  Node val is 0
4.1 ->  Node val is 1
**********
[1]    32715 segmentation fault  python main.py
(memory-test-py3.12) ➜  memory_test
```

Тест памяти


def _exp_6() -> None:
some_list: list[int] = [i for i in range(1_000_000)]
def exp_6() -> None:
# gc.disable()
while True:
_exp_6()
if __name__ == "__main__":
exp_6()

Тест памяти


def _exp_6() -> None:
some_list: list[int] = [i for i in range(1_000_000)]
def exp_6() -> None:
gc.disable()
while True:
_exp_6()
if __name__ == "__main__":
exp_6()

DGTL

@BEILAK

RefCount++

Бейлак Алиев
Expert Software Engineer