

# Track memory leaks in Python



Pycon 2014, Montréal



Victor Stinner  
victor.stinner@gmail.com

# Victor Stinner

---



- Python core developer since 2010
- [github.com/haypo/](https://github.com/haypo/)
- [bitbucket.org/haypo/](https://bitbucket.org/haypo/)
- Working for **eNovance**

# Reference cycle

---



```
a.b = b
```

```
b.a = a
```

```
# a → b → a
```

```
a = None
```

```
b = None
```

```
# a and b are not deleted
```



# Reference cycle

---



```
a.b = b
b.a = weakref.ref(a)
# b.a() is a
a = None      # delete a
# b.a() is None
```

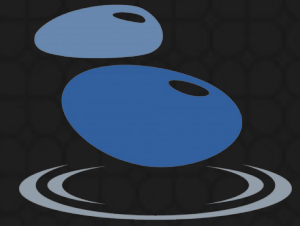
# View the references

---

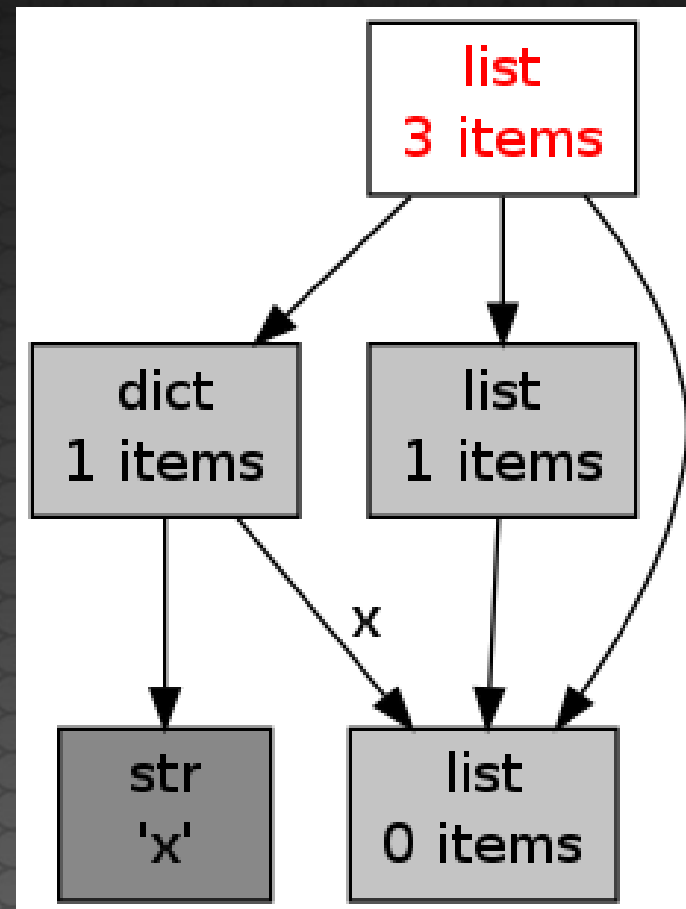


```
>>> import gc
>>> data = {'abc': 123}
>>> gc.get_referents(data)
['abc', 123]
```

# View the references



objgraph project



<http://mg.pov.lt/objgraph/>

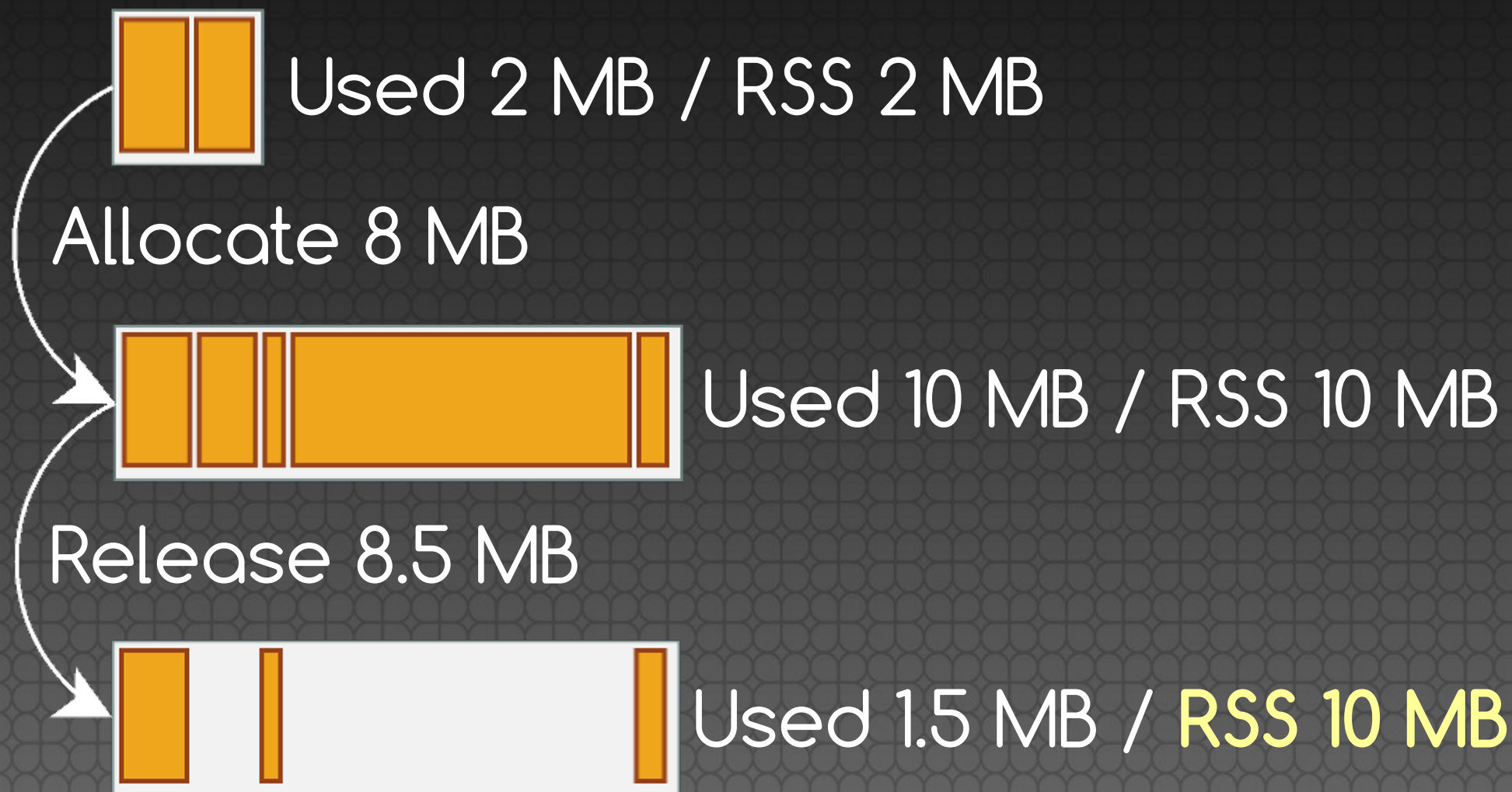
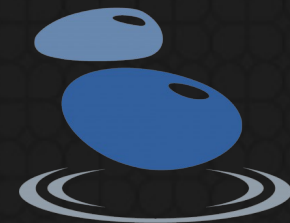


# RSS memory



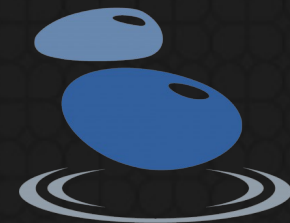
- Representative for the system
- Coarse measurement
- Heap fragmentation
- Difficult to exploit

# Heap fragmentation





# memory\_profiler



Mem usage	Increment	Line Contents
-----------	-----------	---------------

=====

		@profile
5.97 MB	0.00 MB	def my_func():
13.61 MB	<b>7.64 MB</b>	a = [1] * (10 ** 6)
166.20 MB	<b>152.59 MB</b>	b = [2] * (10 ** 8)
13.61 MB	<b>-152.59 MB</b>	del b
13.61 MB	0.00 MB	return a

[http://pypi.python.org/pypi/memory\\_profiler](http://pypi.python.org/pypi/memory_profiler)

# Manual computation

---



```
>>> data = {None: b'x' * 10000}
>>> sys.getsizeof(data)
296
>>> sum(sys.getsizeof(ref)
...      for ref in gc.get_referents(data))
10049
```

# Heapy, Pympler, Melia



- List all Python objects:  
`gc.get_objects()`
- Compute the objects size
- Group objects by type



# Heapy, Pympler, Melia



Total 17916 objects, 96 types,  
Total size = 1.5MiB

Count	Size	Kind
701	<b>546,460</b>	dict
<b>7,138</b>	414,639	str
208	94,016	type
1,371	93,228	code
...		

# Heapy, Pympler, Melia



- Don't trace all the memory (ex: zlib)
- Don't provide the origin of objects
- Difficult to exploit

# PEP 445: API malloc()



- PyMem\_GetAllocator()
- PyMem\_SetAllocator()
- Replace memory allocators
- Set up a hook on allocators
- Implemented in Python 3.4



# PEP 454: tracemalloc

---



```
traces = {}
```

```
def trace_malloc(size):  
    ptr = malloc(size)  
    if ptr:  
        tb = traceback.extract_stack()  
        traces[ptr] = (size, tb)  
    return ptr
```

# PEP 454: tracemalloc

---



```
def trace_free(ptr):  
    if ptr in traces:  
        del traces[ptr]  
    free(ptr)
```

# Tracemalloc features

---



- No overhead when disabled
- Get the traceback where an object was allocated
- Compute statistics per filename, line number or traceback
- Compute differences between two snapshots



# Tracemalloc

Previous Next

Snapshot: tracemalloc-351-0002.pickle (12.0 MiB, 91096 traces, 2014-03-12 18:42:01)

compared to: tracemalloc-351-0021.pickle (67.0 MiB, 491921 traces, 2014-03-12 18:52:21)

Load

Group by: Line number ☐ Cumulative sizes Filters: (none)

Line	Size	Size Diff ▲	Count	Count Diff	Item Size	%Total
<frozen importlib._bootstrap>:656	19.0 MiB	+14.0 MiB	166052	+116214	121 B	28.3 %
<frozen importlib._bootstrap>:321	10.0 MiB	+10144 KiB	104375	+100355	105 B	15.5 %
.../default/Lib/linecache.py:127	3223 KiB	+2535 KiB	28313	+22414	116 B	4.6 %
.../Lib/unittest/case.py:574 lines = fp.readlines()	154 KiB		4731	+4134	509 B	3.4 %
.../Lib/test/test_enumerate.py:150	1698 KiB	+1698 KiB	29533	+29533	58 B	2.5 %
.../Lib/test/test_datetime.py:32	1248 KiB	+1248 KiB	27	+27	46.0 KiB	1.8 %

Lines: 35414 - Total: 67.0 MiB (+55.0 MiB)

/home/haypo/prog/python/default/Lib/linecache.py:127

```

122:         pass
123:     else:
124:         return []
125:     try:
126:         with tokenize.open(fullname) as fp:
127:             lines = fp.readlines()
128:     except OSError:
129:         return []
130:     if lines and not lines[-1].endswith('\n'):
131:         lines[-1] += '\n'
132:     size, mtime = stat.st_size, stat.st_mtime
133:     cache[filename] = size, mtime, lines, fullname
134:     return lines

```

# tracemallocqt

Line	Size	Size Diff ▲
<frozen importlib._bootstrap>:656	19.0 MiB	+14.0 MiB
<frozen importlib._bootstrap>:321	10.0 MiB	+10144 KiB
.../default/Lib/linecache.py:127	3223 KiB	+2535 KiB
.../Lib/unittest/case.py:574	lines = fp.readlines()	1054 KiB
.../Lib/test/test_enumerate.py:150	1698 KiB	+1698 KiB
.../Lib/test/test_datetime.py:32	1248 KiB	+1248 KiB

Lines: 35414 - Total: 67.0 MiB (+55.0 MiB)

# tracemallocqt

```
122:         pass
123:     else:
124:         return []
125:     try:
126:         with tokenize.open(fullname) as fp:
127:             lines = fp.readlines()
128:     except OSError:
129:         return []
130:     if lines and not lines[-1].endswith('\n'):
131:         lines[-1] += '\n'
132:     size, mtime = stat.st_size, stat.st_mtime
133:     cache[filename] = size, mtime, lines, fullname
134:     return lines
```



# tracemallocqt

	Size ▲	Count	Item Size	%
/Lib/linecache.py:15 <= ...	360 KiB	3169	116 B	22.3

Traceback (most recent first):  
/home/haypo/prog/python/default/Lib/linecache.py:127  
    **lines = fp.readlines()**  
/home/haypo/prog/python/default/Lib/linecache.py:41  
    **return updatecache(filename, module\_globals)**  
/home/haypo/prog/python/default/Lib/linecache.py:15  
    **lines = getlines(filename, module\_globals)**  
/home/haypo/prog/python/default/Lib/traceback.py:65  
    **line = linecache.getline(filename, lineno, f.f\_globals)**  
/home/haypo/prog/python/default/Lib/traceback.py:18  
    **for filename, lineno, name, line in extracted\_list:**  
/home/haypo/prog/python/default/Lib/traceback.py:153  
        **yield from \_format\_list\_iter(\_extract\_tb\_iter(tb, limit=**  
/home/haypo/prog/python/default/Lib/traceback.py:181  
    **return list(\_format\_exception\_iter(etype, value, tb, lir**  
/home/haypo/prog/python/default/Lib/unittest/result.py:181

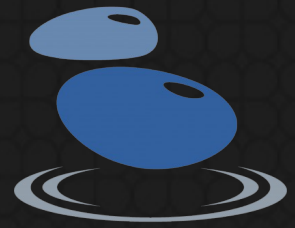
# tracemallocqt

	Size ▲	Count	Item Size	%
/Lib/linecache.py:15 <= ...	360 KiB	3169	116 B	22.3

Traceback (most recent first):

```
/home/haypo/prog/python/default/Lib/linecache.py:127
  lines = fp.readlines()
/home/haypo/prog/python/default/Lib/linecache.py:41
  return updatecache(filename, module_globals)
/home/haypo/prog/python/default/Lib/linecache.py:15
  lines = getlines(filename, module_globals)
/home/haypo/prog/python/default/Lib/traceback.py:65
  line = linecache.getline(filename, lineno, f.f_globals)
/home/haypo/prog/python/default/Lib/traceback.py:18
  for filename, lineno, name, line in extracted_list:
/home/haypo/prog/python/default/Lib/traceback.py:153
  yield from _format_list_iter(_extract_tb_iter(tb, limit=
/home/haypo/prog/python/default/Lib/traceback.py:181
  return list(_format_exception_iter(etype, value, tb, lir
/home/haypo/prog/python/default/Lib/unittest/result.py:181
```

# tracemalloc backport



- Available at PyPI
- Require to patch and recompile Python
- ... maybe also recompile Python extensions written in C
- Patches for Python 2.7 and 3.3
- Ubuntu packages



# Questions ?

<http://pytracemalloc.readthedocs.org/>



Contact :

[victor.stinner@gmail.com](mailto:victor.stinner@gmail.com)

# PEP 445 (API malloc)



- Ticket opened in 2008
- Patch proposed in march 2013
- Patch committed in june 2013
- Commit reverted => PEP 445
- Better API thanks to the PEP
- BDFL delegate: Antoine Pitrou

# PEP 454 (tracemalloc)



- Store the traceback, not just 1 frame
- Code rewritten from scratch
- Much better API
- Exchanges with Kristján Valur Jónsson
- BDFL delegate: Charles-François Natali



Thanks David Malcom  
for the LibreOffice model

<http://dmalcolm.livejournal.com/>