# How to Profile Memory Usage in Python

By Luke Lee on December 5, 2019

One of the ways Python makes development fast (not to mention easier than languages like C and C++ ) is memory management. In Python, it's simple because the language handles memory management for you. However, this doesn't mean memory should be forgotten. Good developers will want to track the memory usage of their application and look to lower memory usage. Take a look at the common tools for doing this.

## How Do You Profile Size of Individual Objects?

The lowest layer of memory profiling involves looking at a single object in memory. You can do this by opening up a shell and doing something like the following:

```
>>> import sys

>>> sys.getsizeof({})

136

>>> sys.getsizeof([])

32

>>> sys.getsizeof(set())

112
```

The above snippet illustrates the overhead associated with a list object. A list is 32 bytes (on a 32-bit machine running Python 2.7.3). This style of profiling is useful when determining what type of data type to use.

## How Do You Profile a Single Function or Method?

The easiest way to profile a single method or function is the open source memory-profiler package. It's similar to line_profiler , which I've written about before .

You can use it by putting the **@profile** decorator around any function or method and running **python -m memory_profiler myscript**. You'll see line-by-line memory usage once your script exits.

This is extremely useful if you're wanting to profile a section of memory-intensive code, but it won't help much if you have no idea where the biggest memory usage is. In that case, a higher-level approach of profiling is needed first.

## How Do You Profile an Entire Application?

Rather than having to profile an entire Python application, use guppy .

To use guppy you drop something like the following in your code:

```
from guppy import hpy

h = hpy()

print h.heap()
```

This will print a nice table of usage grouped by object type. Here's an example of an PyQt4 application I've been working on:

```
 Partition of a set of 235760 objects. Total size = 19909080
bytes. Index Count % Size % Cumulative % Kind (class / dict of
class)
0 97264  41 8370996 42 8370996   42 str
1 47430  20 1916788 10 10287784  52 tuple
2 937     0 1106440  6 11394224   57 dict of
PyQt4.QtCore.pyqtWrapperType
3 646     0 1033648  5 12427872   62 dict of module
4 11683  5 841176   4 13269048   67 types.CodeType
5 11684  5 654304   3 13923352   70 function
6 1200   1 583872   3 14507224   73 dict of type
7 782    0 566768   3 15073992   76 dict (no owner)
8 1201   1 536512   3 15610504   78 type
9 1019   0 499124   3 16109628   81 unicode
```

This type of profiling can be difficult if you have a large application using a relatively small number of object types.

## mprof

mprof can show you memory usage over the lifetime of your application. This can be useful if you want to see if your memory is getting cleaned up and released periodically.

Even better, using mprof is easy; just run mprof run script script_args in your shell of choice. mprof will automatically create a graph of your script's memory usage over time, which you can view by running mprof plot. It's important to note here that plotting requires matplotlib. This is helpful when determining Python profile memory usage.

## How Do You Deal with Memory Leaks in Python?

If you are dealing with memory leaks in Python, the easiest way to deal with them is to increase the memory allocation. While this is a quick fix, it comes at the cost of potentially creating an unstable product. A better solution is to profile the memory usage of the application to better gain an understanding of the code and the packages that are being used. Once you are able to profile the memory and understand where leaks are happening, you can better deal with memory dumps.

To try to fix and avoid memory leaks, consider running memory-intensive tasks in separate processes. This will reduce the number of memory leaks and ensures that memory is released only after code has been executed.

Contributor

## Luke Lee

Luke Lee lives in Dresden, Germany and also writes at www.lukelee.me.