

Profiling Memory Use: %memit and %mprun

Another aspect of profiling is the amount of memory an operation uses. This can be evaluated with another IPython extension, the `memory_profiler`. As with the `line_profiler`, we start by pip-installing the extension:

```
$ pip install memory_profiler
```

Then we can use IPython to load the extension:

```
In[12]: %load_ext memory_profiler
```

The memory profiler extension contains two useful magic functions: the `%memit` magic (which offers a memory-measuring equivalent of `%timeit`) and the `%mprun` function (which offers a memory-measuring equivalent of `%lprun`). The `%memit` function can be used rather simply:

```
In[13]: %memit sum_of_lists(1000000)

peak memory: 100.08 MiB, increment: 61.36 MiB
```

We see that this function uses about 100 MB of memory.

For a line-by-line description of memory use, we can use the `%mprun` magic. Unfortunately, this magic works only for functions defined in separate modules rather than the notebook itself, so we'll start by using the `%%file` magic to create a simple module called `mprun_demo.py`, which contains our `sum_of_lists` function, with one addition that will make our memory profiling results more clear:

```
In[14]: %%file mprun_demo.py
def sum_of_lists(N):
    total = 0
    for i in range(5):
        L = [j ^ (j >> i) for j in range(N)]
        total += sum(L)
        del L # remove reference to L
    return total
```

Overwriting `mprun_demo.py`

We can now import the new version of this function and run the memory line profiler:

```
In[15]: from mprun_demo import sum_of_lists
        %mprun -f sum_of_lists sum_of_lists(1000000)
```

The result, printed to the pager, gives us a summary of the memory use of the function, and looks something like this:

Filename: ./mprun_demo.py

Line #	Mem usage	Increment	Line Contents
4	71.9 MiB	0.0 MiB	L = [j ^ (j >> i) for j in range(N)]

Filename: ./mprun_demo.py

Line #	Mem usage	Increment	Line Contents
1	39.0 MiB	0.0 MiB	def sum_of_lists(N):
2	39.0 MiB	0.0 MiB	total = 0
3	46.5 MiB	7.5 MiB	for i in range(5):
4	71.9 MiB	25.4 MiB	L = [j ^ (j >> i) for j in range(N)]
5	71.9 MiB	0.0 MiB	total += sum(L)
6	46.5 MiB	-25.4 MiB	del L # remove reference to L
7	39.1 MiB	-7.4 MiB	return total

Here the Increment column tells us how much each line affects the total memory budget: observe that when we create and delete the list L, we are adding about 25 MB of memory usage. This is on top of the background memory usage from the Python interpreter itself.

For more information on `%memit` and `%mprun`, as well as their available options, use the IPython help functionality (i.e., type `%memit?` at the IPython prompt).

More IPython Resources

In this chapter, we've just scratched the surface of using IPython to enable data science tasks. Much more information is available both in print and on the Web, and here we'll list some other resources that you may find helpful.

Web Resources

The IPython website

The IPython website links to documentation, examples, tutorials, and a variety of other resources.

The nbviewer website

This site shows static renderings of any IPython notebook available on the Internet. The front page features some example notebooks that you can browse to see what other folks are using IPython for!