

# Python profiling 101

# python -m cProfile mandelbrot.py

4707852 function calls (4707840 primitive calls) in 2.099 seconds

Ordered by: internal time

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	1.622	1.622	2.072	2.072	mandelbrot.py:27(calculate_z_serial)
4638829	0.364	0.000	0.364	0.000	{abs}
22501	0.086	0.000	0.086	0.000	{range}
1	0.006	0.006	2.098	2.098	mandelbrot.py:41(compute)
22501	0.004	0.000	0.004	0.000	mandelbrot.py:15(<genexpr>)
1	0.004	0.004	0.018	0.018	mandelbrot.py:9(show)
1	0.001	0.001	0.005	0.005	Image.py:27(<module>)
1	0.001	0.001	0.001	0.001	{posix.system}
1	0.001	0.001	2.099	2.099	mandelbrot.py:2(<module>)
22806	0.001	0.000	0.001	0.000	{method 'append' of 'list' objects}
1	0.001	0.001	0.001	0.001	__init__.py:4(<module>)
1	0.001	0.001	0.002	0.002	FixTk.py:1(<module>)
1	0.001	0.001	0.001	0.001	{method 'save_ppm' of 'ImagingCore' objects}
1	0.000	0.000	0.001	0.001	hashlib.py:55(<module>)
1	0.000	0.000	0.001	0.001	collections.py:1(<module>)
1	0.000	0.000	0.005	0.005	Image.py:1(<module>)
1	0.000	0.000	0.002	0.002	tempfile.py:18(<module>)
1	0.000	0.000	0.001	0.001	io.py:34(<module>)
1	0.000	0.000	0.001	0.001	random.py:40(<module>)
1	0.000	0.000	0.003	0.003	Image.py:493(_dump)
2	0.000	0.000	0.000	0.000	{built-in method now}

```
python -m cProfile -o mandelbrot.py.pstats mandelbrot.py
runsnake mandelbrot.py.pstats
```

Run Snake Run: mandelbrot.py.pstats

Name	Calls	RCalls	Local	/Call	Cum
<module>	0	2	0.00000	0.00000	2.0021
<module>	1	1	0.00103	0.00103	2.0021
compute	1	1	0.00603	0.00603	2.0011
calculate_...	1	1	1.54349	1.54349	1.9759
<abs>	4638829	4638829	0.35057	0.00000	0.3505
<range>	22501	22501	0.08187	0.00000	0.0818
show	1	1	0.00366	0.00366	0.0178
<module>	1	1	0.00041	0.00041	0.0054
<module>	1	1	0.00140	0.00140	0.0049
show	1	1	0.00000	0.00000	0.0046
_show	1	1	0.00000	0.00000	0.0046
_showxv	1	1	0.00008	0.00008	0.0046
show	1	1	0.00000	0.00000	0.0044
show	1	1	0.00001	0.00001	0.0044
show_ima...	1	1	0.00003	0.00003	0.0043
<genexpr>	22501	22501	0.00401	0.00000	0.0040
save_image	1	1	0.00000	0.00000	0.0027
_dump	1	1	0.00017	0.00017	0.0027
<_import...	1	1	0.00009	0.00009	0.0025
<module>	1	1	0.00086	0.00086	0.0024
<module>	1	1	0.00034	0.00034	0.0017
<module>	1	1	0.00109	0.00109	0.0015
show_file	1	1	0.00002	0.00002	0.0015
<posix.sy...	1	1	0.00153	0.00153	0.0015
<method ...	22806	22806	0.00103	0.00000	0.0010
<module>	1	1	0.00027	0.00027	0.0009
<module>	1	1	0.00054	0.00054	0.0005
<module>	1	1	0.00040	0.00040	0.0005
<module>	1	1	0.00025	0.00025	0.0005

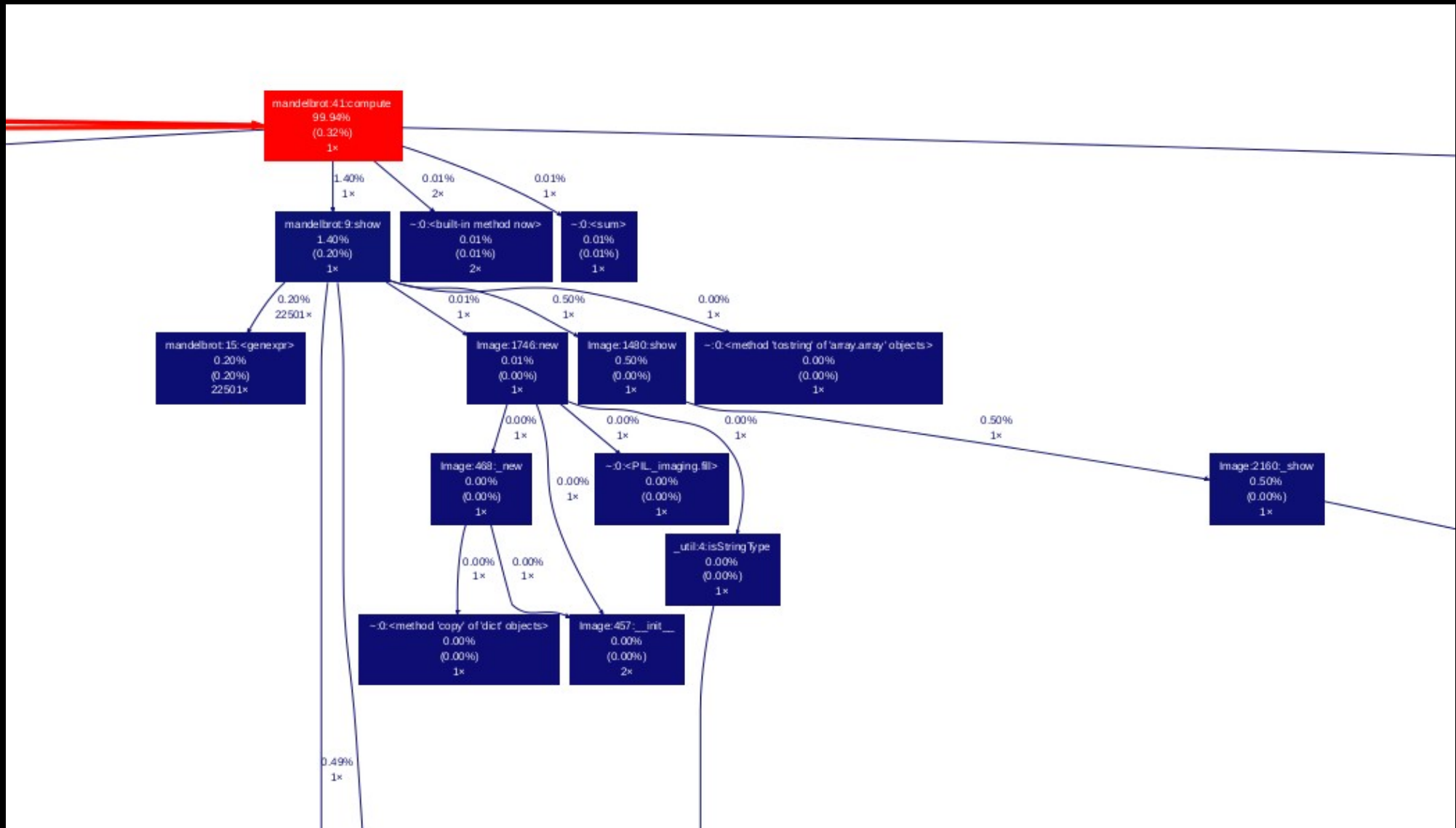
calculate\_z\_serial@mandelbrot.py:27 [1.976s]

<abs>@~:0 [0.351s]

<range>@~:0 [0.082s]

Name	Calls	RCalls	Local	/Call	Cum	/Call	File	Line	Directory
<abs>	4638829	4638829	0.35057	0.00000	0.35057	0.00000	~	0	
<range>	22501	22501	0.08187	0.00000	0.08187	0.00000	~	0	
<len>	41	41	0.00001	0.00000	0.00001	0.00000	~	0	

```
python -m cProfile -o ./mandelbrot.py.pstats mandelbrot.py
gprof2dot -f pstats ./mandelbrot.py.pstats -n 0 -e 0 | dot -Tpdf
-o mb.profile.pdf
```



# kernprof.py --line-by-line mandelbrot\_profile.py python -m line\_profiler mandelbrot\_profile.py.lprof

File: mandelbrot\_profile.py  
Function: calculate\_z\_serial at line 27  
Total time: 11.441 s

Line #	Hits	Time	Per Hit	% Time	Line Contents
27					@profile
28					def calculate_z_serial(q, maxiter, z):
29	1	278	278.0	0.0	output = [0] * len(q)
30	22501	17342	0.8	0.2	for i in range(len(q)):
31	22500	17563	0.8	0.2	if i % 1000 == 0:
32					# print out some progress info since it is so slow...
33	23	691	30.0	0.0	print "%.2f%% complete" % (1.0/len(q) * i * 100)
34	4643349	3213216	0.7	28.1	for iteration in range(maxiter):
35	4638829	4163215	0.9	36.4	z[i] = z[i]*z[i] + q[i]
36	4638829	3940721	0.8	34.4	if abs(z[i]) > 2.0:
37	17980	13230	0.7	0.1	output[i] = iteration
38	17980	74703	4.2	0.7	break
39	1	4	4.0	0.0	return output

```
python -c 'from dis import dis; from mandelbrot import
calculate_z_serial as c; dis(c)'
```

```
28      0 LOAD_CONST          1 (0)
      3 BUILD_LIST          1
      6 LOAD_GLOBAL        0 (len)
      9 LOAD_FAST          0 (q)
     12 CALL_FUNCTION      1
     15 BINARY_MULTIPLY
     16 STORE_FAST         3 (output)

29     19 SETUP_LOOP        161 (to 183)
     22 LOAD_GLOBAL        1 (range)
     25 LOAD_GLOBAL        0 (len)
     28 LOAD_FAST          0 (q)
     31 CALL_FUNCTION      1
     34 CALL_FUNCTION      1
     37 GET_ITER
    >> 38 FOR_ITER           141 (to 182)
     41 STORE_FAST         4 (i)

30     44 LOAD_FAST          4 (i)
     47 LOAD_CONST         2 (1000)
     50 BINARY_MODULO
     51 LOAD_CONST         1 (0)
     54 COMPARE_OP         2 (==)
     57 POP_JUMP_IF_FALSE 90

32     60 LOAD_CONST         3 ('%0.2f%% complete')
     63 LOAD_CONST         4 (1.0)
     66 LOAD_GLOBAL        0 (len)
     69 LOAD_FAST          0 (q)
     72 CALL_FUNCTION      1
     75 BINARY_DIVIDE
     76 LOAD_FAST          4 (i)
     79 BINARY_MULTIPLY
```

<http://github.com/pszostek/pyprof101>