

IVS – profiling – xpotucm00, xhegrfi00, xtumafi00
Output after calling stddev.py and inputing 1mil. values.

```
15.002828053395959
12000019 function calls in 3.886 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1      1.058    1.058    3.858    3.858 stddev.py:6(main)
2000000 0.926    0.000    1.333    0.000 mathlib.py:8(add)
8000006 0.821    0.000    0.821    0.000 {built-in method builtins.isinstance}
1000000 0.488    0.000    0.695    0.000 mathlib.py:24(sub)
1000000 0.487    0.000    0.694    0.000 mathlib.py:40(mul)
1      0.054    0.054    0.054    0.054 {method 'split' of 'str' objects}
1      0.028    0.028    3.886    3.886 <string>:1(<module>)
1      0.015    0.015    0.025    0.025 {method 'read' of '_io.TextIOWrapper' objects}
1      0.010    0.010    0.010    0.010 {built-in method codecs.utf_8_decode}
1      0.000    0.000    3.886    3.886 {built-in method builtins.exec}
1      0.000    0.000    0.010    0.010 <frozen codecs>:319(decode)
1      0.000    0.000    0.000    0.000 {built-in method builtins.print}
1      0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
1      0.000    0.000    0.000    0.000 mathlib.py:113(nthrt)
2      0.000    0.000    0.000    0.000 mathlib.py:56(div)
1      0.000    0.000    0.000    0.000 {built-in method builtins.len}
```

The program was running for 3.886 seconds. Most of the time was spent in our MathLib. All 3 MathLib functions are running circa the same amount of time compared to the number of calls. The only thing that we think can be optimized is reducing the number of calls of `.isinstance()`. In the case of the calculator this can probably be done in the parser stage. We can also use some other language than python and thus speed it up or look for other way to circumvent the need for `.isinstance()` without compromising the functioning and type safety of the MathLib. Also in the case of the stddev.py file itself we can substitute the MathLib calls for in-program math (like instead of calling `MathLib.add(x, y)` just write `x + y`. But again this doesn't optimize the MathLib itself :).