IVS - Profiling

Testovací tým

15. dubna 2019

Výsledek profilovací knihovny:

15265 function calls in 0.048 seconds

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.048	0.048	:0(exec)
6025	0.009	0.000	0.009	0.000	:0(isinstance)
1	0.000	0.000	0.000	0.000	:0(len)
1	0.000	0.000	0.000	0.000	:0(print)
1	0.000	0.000	0.000	0.000	:0(setprofile)
200	0.000	0.000	0.000	0.000	:0(split)
3	0.000	0.000	0.000	0.000	:0(utf_8_decode)
1	0.000	0.000	0.048	0.048	<string>:1(<module>)
3	0.000	0.000	0.000	0.000	codecs.py:318(decode)
1	0.006	0.006	0.048	0.048	deviation.py:16(run)
1	0.000	0.000	0.048	0.048	profile:0(deviation.run())
0	0.000	0.000			profile:0(profiler)
2	0.000	0.000	0.000	0.000	rgg_mathlib.py:12(sub)
3	0.000	0.000	0.000	0.000	rgg_mathlib.py:18(mul)
2	0.000	0.000	0.000	0.000	rgg_mathlib.py:24(div)
6018	0.018	0.000	0.026	0.000	rgg_mathlib.py:3(__valid_operand)
1001	0.005	0.000	0.014	0.000	rgg_mathlib.py:30(pow)
1	0.000	0.000	0.000	0.000	rgg_mathlib.py:51(root)
2000	0.010	0.000	0.027	0.000	rgg_mathlib.py:6(add)

Výsledek odchylky:

1.0000000864464458

Závěr:

Podle všeho trávíme nejvíce času ověřováním platnosti argumentů (0.018s). To protože ověřování platného vstupu je zapotřebí v každém volání knihovní funkce. Kdyby se nám podařilo zefektivnit toto ověřování, mělo by to veliký dopad na rychlost naší knihovny.

Zároveň ovšem ověřování platnosti čísel netrvá tak dlouho, jako samy výpočetní funkce. Vidíme, že ověření čísla bylo zavoláno 3x častěji než sčítání, ale netrvalo dohromady ani dvakrát tak dlouho. Proto by nebylo od věci zaměřit se i na výpočetní funkce, kde už jsme limitováni možnostmi jazyka (Python) a jediné řešení by tedy bylo přejít třeba na Cython.