www.phoronix-test-suite.com

AMD Ryzen 9 5900X python Benchmarks - Python 3 vs. Pyston 2.1

AMD Ryzen 9 5900X Python benchmarks by Michael Larabel.

Automated Executive Summary

Pyston 2.1.0 had the most wins, coming in first place for 85% of the tests.

Based on the geometric mean of all complete results, the fastest (Pyston 2.1.0) was 1.511x the speed of the slowest (Python 3.8.6). Python 3.9.1 was 0.663x the speed of Pyston 2.1.0 and Python 3.8.6 was 0.998x the speed of Python 3.9.1.

Test Systems:

Python 3.8.6

Python 3.9.1

Pyston 2.1.0

Processor: AMD Ryzen 9 5900X 12-Core @ 3.70GHz (12 Cores / 24 Threads), Motherboard: ASUS ROG CROSSHAIR VIII HERO (3202 BIOS), Chipset: AMD Starship/Matisse, Memory: 16GB, Disk: 1000GB Samsung SSD 980 PRO 1TB, Graphics: Sapphire AMD Radeon RX 5600 OEM/5600 XT / 5700/5700 6GB (1780/875MHz), Audio: AMD Navi 10 HDMI Audio, Monitor: ASUS VP28U, Network: Realtek RTL8125 2.5GbE + Intel I211

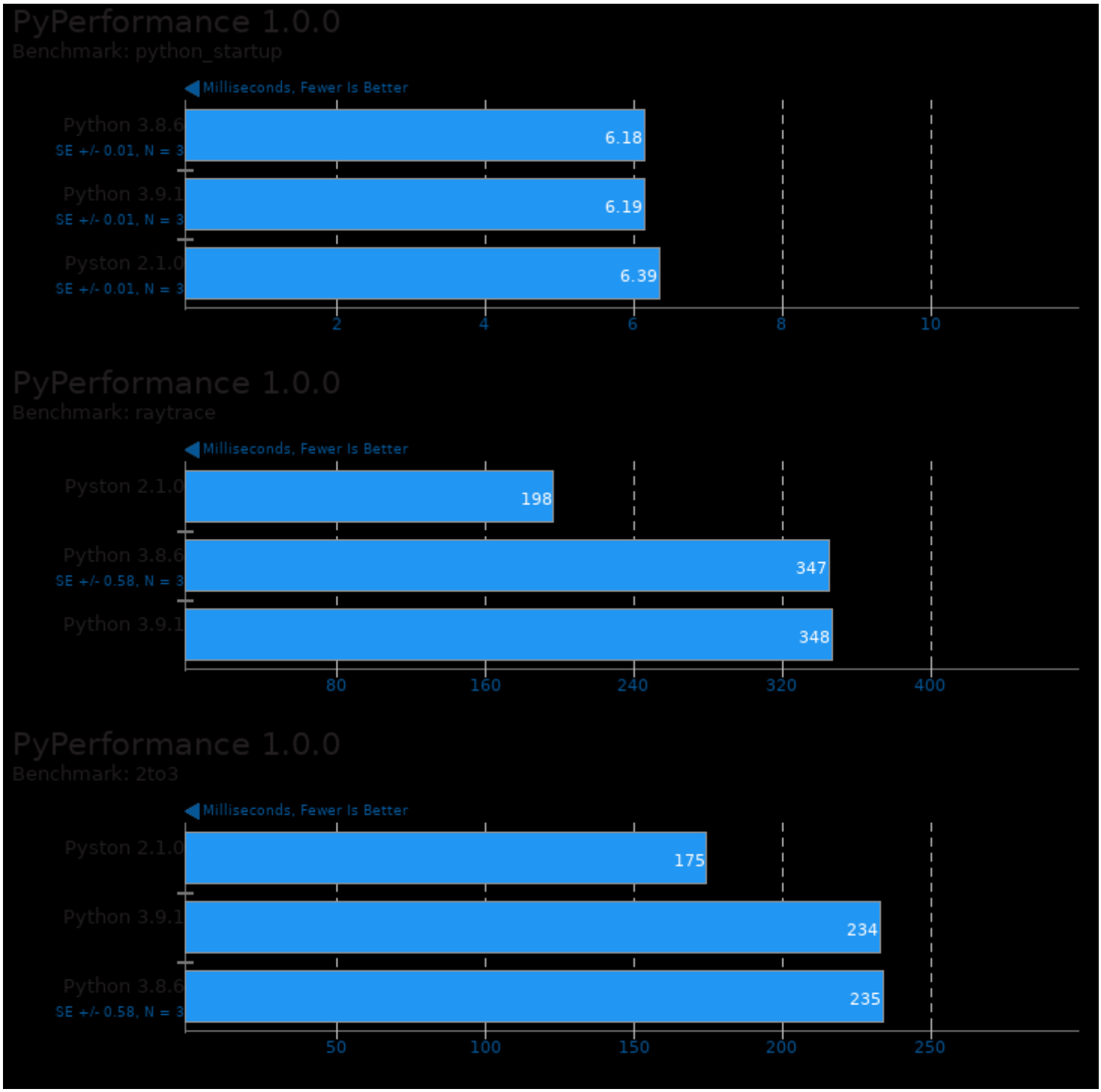
OS: Ubuntu 20.10, Kernel: 5.8.0-38-generic (x86_64), Desktop: GNOME Shell 3.38.1, Display Server: X Server 1.20.9, Display Driver: modesetting 1.20.9, OpenGL: 4.6 Mesa 20.2.1 (LLVM 11.0.0), Vulkan: 1.2.131, Compiler: GCC 10.2.0, File-System: ext4, Screen Resolution: 3840x2160

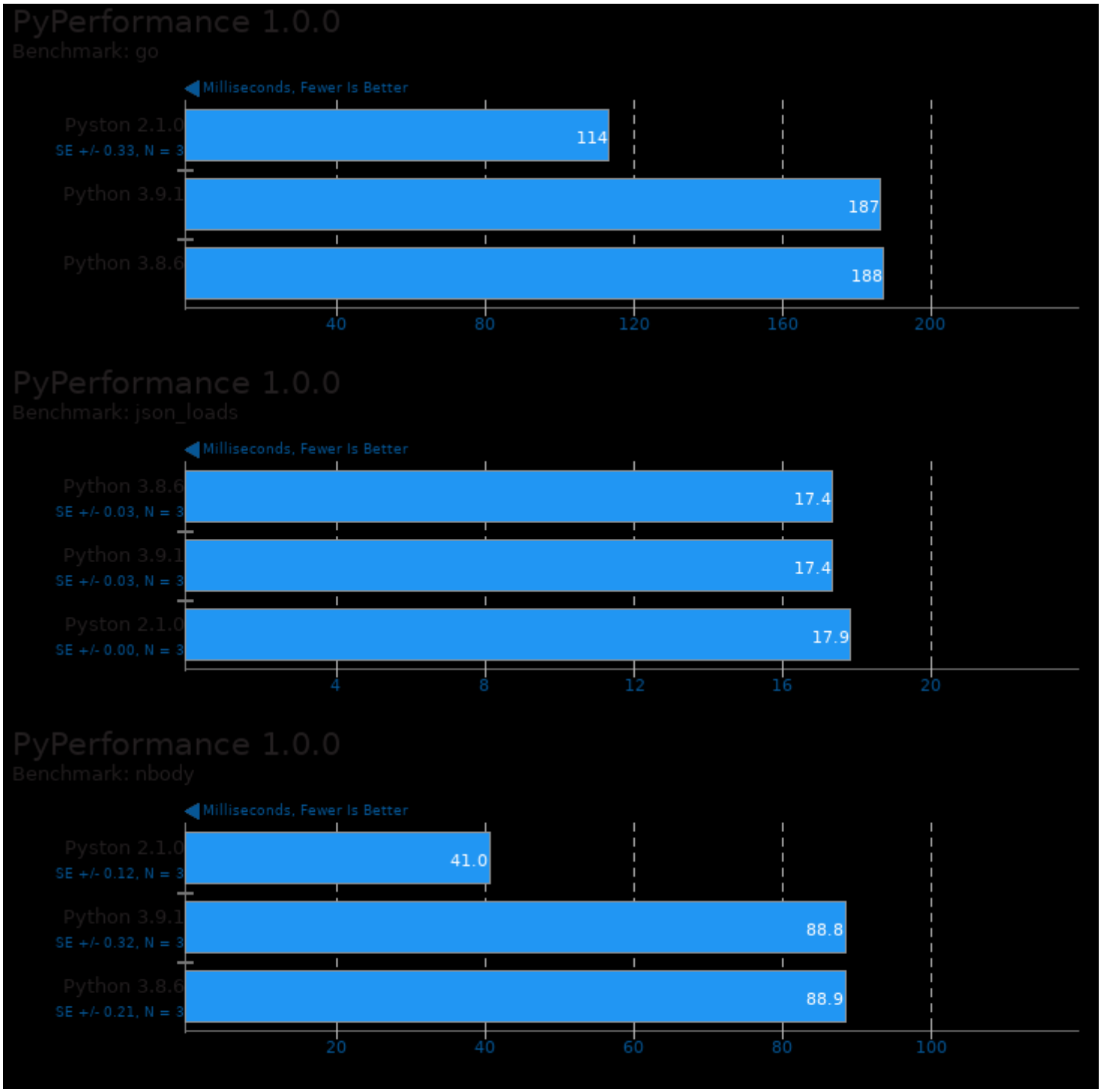
Kernel Notes: Transparent Huge Pages: madvise
Processor Notes: Scaling Governor: acpi-cpufreq ondemand (Boost: Enabled) - CPU Microcode: 0xa201009
Python Notes: Python 3.8.2 (heads/rel2.1:da378ef Jan 12 2021 15:46:12)[Pyston 2.1.0 GCC 9.3.0]
Security Notes: itlb_multihit: Not affected + l1tf: Not affected + mds: Not affected + meltdown: Not affected + spec_store_bypass: Mitigation of SSB disabled via prctl and seccomp + spectre_v1: Mitigation of usercopy/swapgs barriers and __user pointer sanitization + spectre_v2: Mitigation of Full AMD retpoline IBPB: conditional IBRS_FW STIBP: always-on RSB filling + srbds: Not affected + tsx_async_abort: Not affected

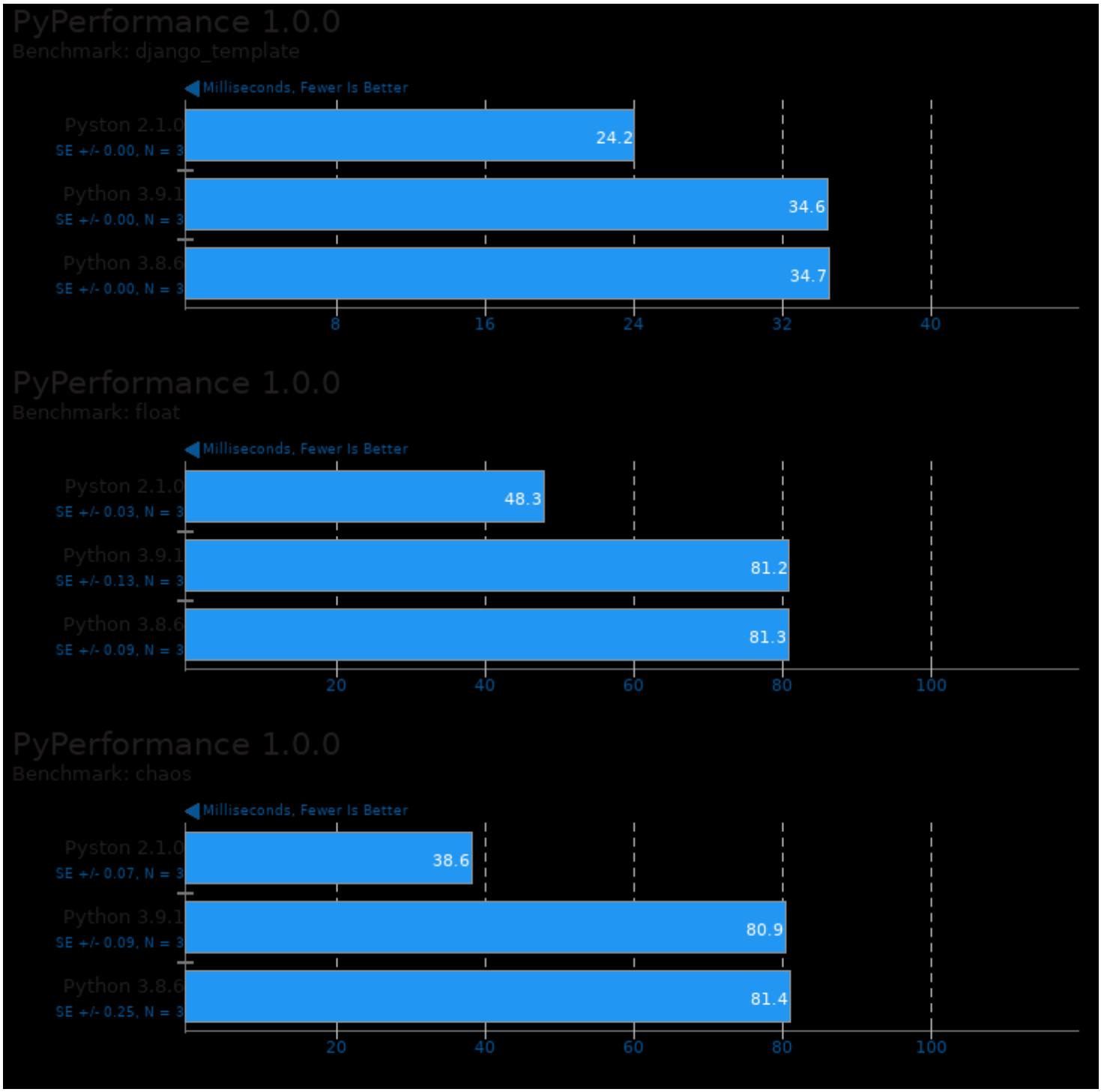
	Python 3.8.6	Python 3.9.1	Pyston 2.1.0
PyPerformance - python_startup	**6.18**	6.19	**6.39**
Normalized	100%	99.84%	96.71%
Standard Deviation	0.2%	0.4%	0.3%
PyPerformance - raytrace (Milliseconds)	347	**348**	**198**
Normalized	57.06%	56.9%	100%
Standard Deviation	0.3%		
PyPerformance - 2to3 (Milliseconds)	**235**	234	**175**
Normalized	74.47%	74.79%	100%
Standard Deviation	0.4%		
PyPerformance - go (Milliseconds)	**188**	187	**114**
Normalized	60.64%	60.96%	100%
Standard Deviation			0.5%
PyPerformance - json_loads (Milliseconds)	**17.4**	**17.4**	**17.9**
Normalized	100%	100%	97.21%
Standard Deviation	0.3%	0.3%	0%
PyPerformance - nbody (Milliseconds)	**88.9**	88.8	**41.0**
Normalized	46.12%	46.17%	100%
Standard Deviation	0.4%	0.6%	0.5%
PyPerformance - django_template (Milliseconds)	**34.7**	34.6	**24.2**
Normalized	69.74%	69.94%	100%
Standard Deviation	0%	0%	0%
PyPerformance - float (Milliseconds)	**81.3**	81.2	**48.3**
Normalized	59.41%	59.48%	100%
Standard Deviation	0.2%	0.3%	0.1%
PyPerformance - chaos (Milliseconds)	**81.4**	80.9	**38.6**
Normalized	47.42%	47.71%	100%
Standard Deviation	0.5%	0.2%	0.3%
PyPerformance - pathlib (Milliseconds)	**12.7**	**12.7**	**11.0**
Normalized	86.61%	86.61%	100%
Standard Deviation	0%	0%	0.5%
PyPerformance - regex_compile	**123**	**123**	**66.2**
Normalized	53.82%	53.82%	100%

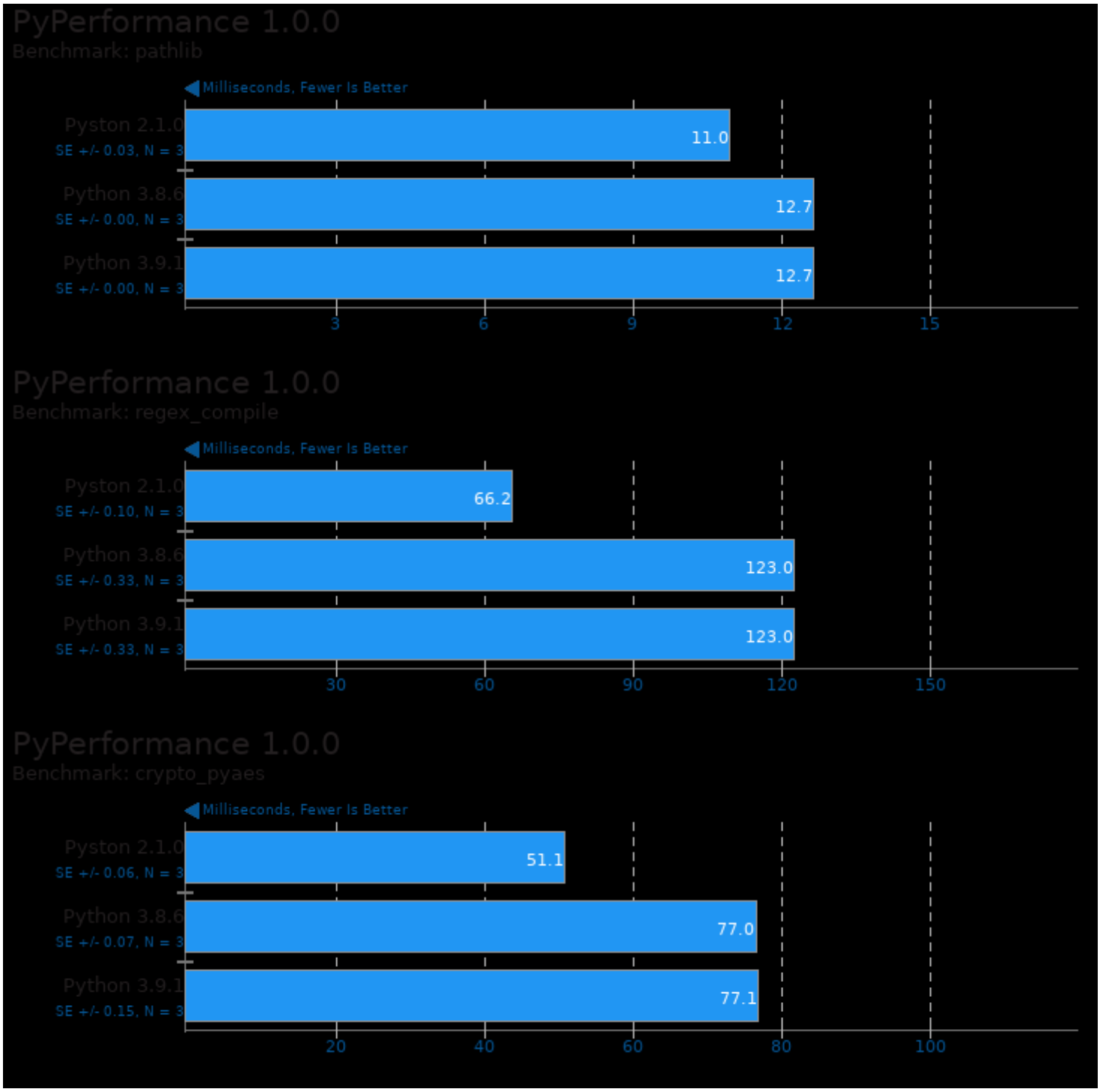
AMD Ryzen 9 5900X python Benchmarks - Python 3 vs. Pyston 2.1

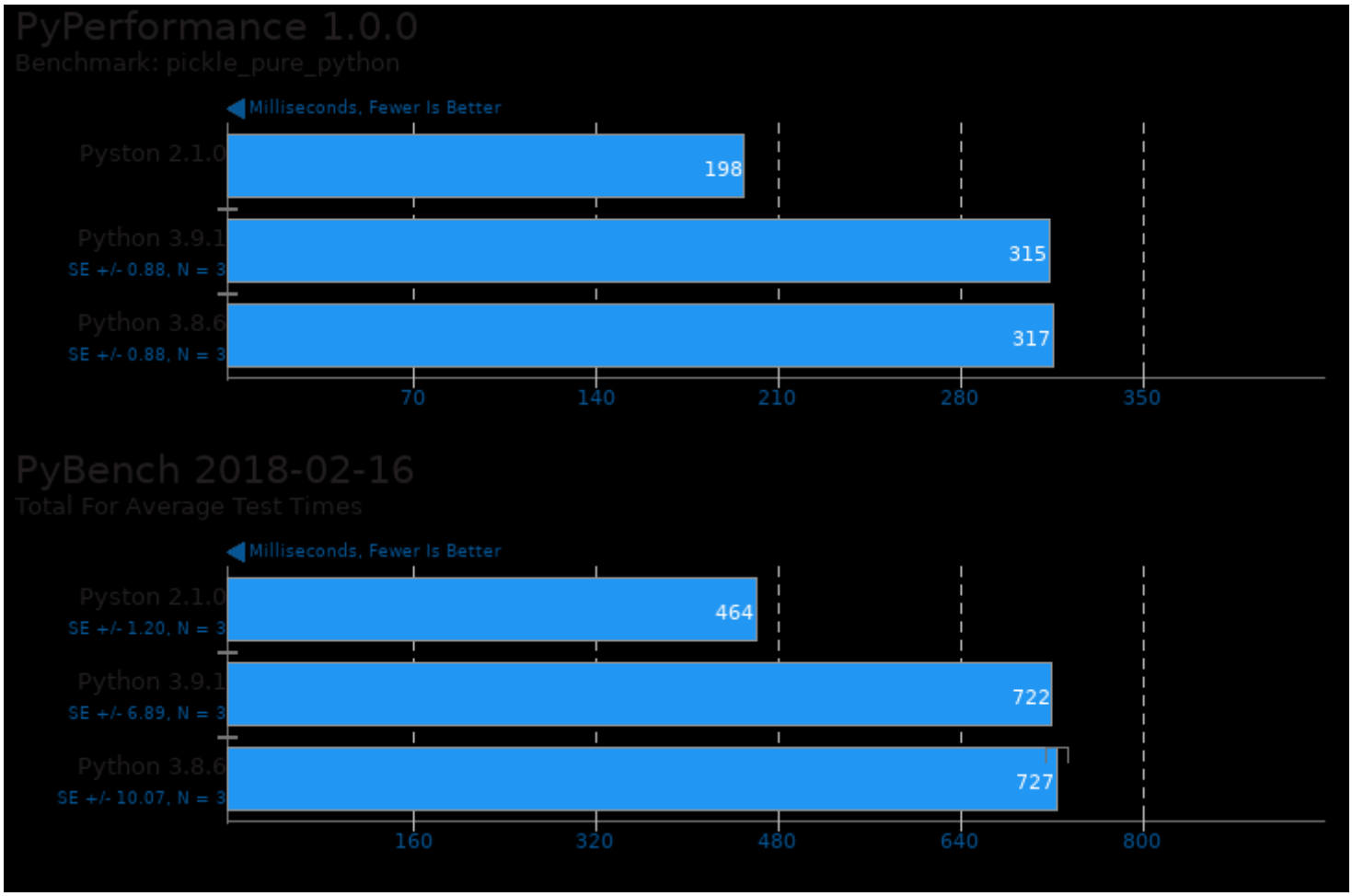
Standard Deviation	0.5%	0.5%	0.3%
PyPerformance - crypto_pyaes	77.0	**77.1**	**51.1**
Normalized	66.36%	66.28%	100%
Standard Deviation	0.1%	0.3%	0.2%
PyPerformance - pickle_pure_python (Milliseconds)	**317**	315	**198**
Normalized	62.46%	62.86%	100%
Standard Deviation	0.5%	0.5%	
PyBench - T.F.A.T.T (Milliseconds)	**727**	722	**464**
Normalized	63.82%	64.27%	100%
Standard Deviation	2.4%	1.7%	0.4%


PyPerformance 1.0.0
Benchmark: python_startup
Milliseconds, Fewer Is Better
Python 3.8.6
SE +/- 0.01, N = 3
6.18
Python 3.9.1
SE +/- 0.01, N = 3
6.19
Pyston 2.1.0
SE +/- 0.01, N = 3
6.39
2
4
6
8
10
PyPerformance 1.0.0
Benchmark: raytrace
Milliseconds, Fewer Is Better
Pyston 2.1.0
198
Python 3.8.6
SE +/- 0.58, N = 3
347
Python 3.9.1
348
80
160
240
320
400
PyPerformance 1.0.0
Benchmark: 2to3
Milliseconds, Fewer Is Better
Pyston 2.1.0
175
Python 3.9.1
234
Python 3.8.6
SE +/- 0.58, N = 3
235
50
100
150
200
250

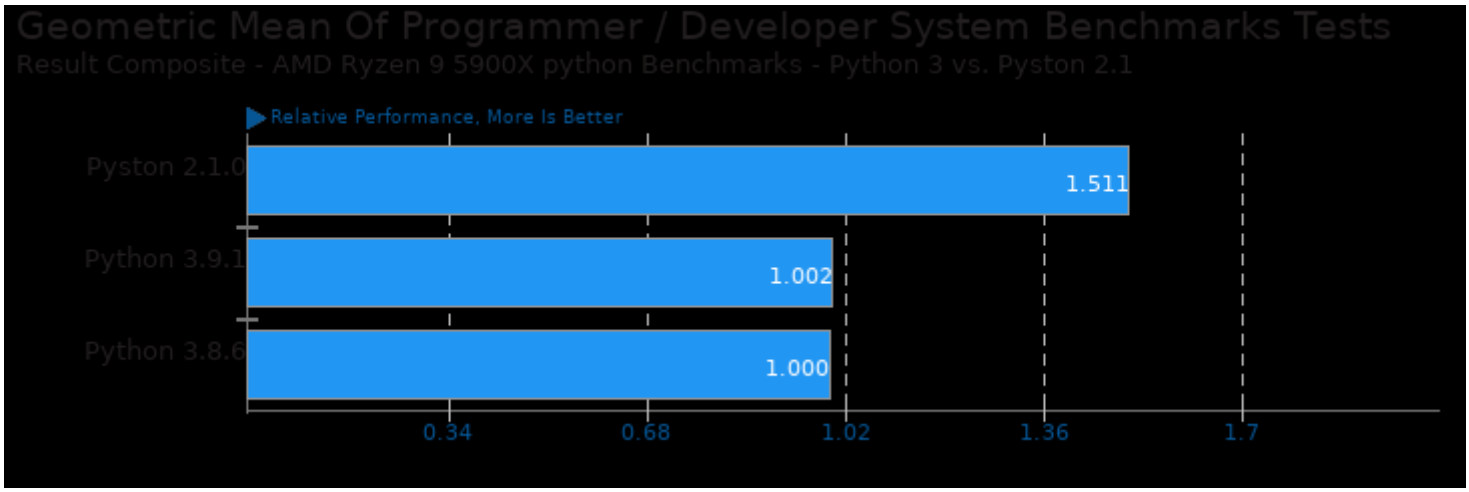

PyPerformance 1.0.0
Benchmark: go
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.33, N = 3
114
Python 3.9.1
187
Python 3.8.6
188
40
80
120
160
200
PyPerformance 1.0.0
Benchmark: json_loads
Milliseconds, Fewer Is Better
Python 3.8.6
SE +/- 0.03, N = 3
17.4
Python 3.9.1
SE +/- 0.03, N = 3
17.4
Pyston 2.1.0
SE +/- 0.00, N = 3
17.9
4
8
12
16
20
PyPerformance 1.0.0
Benchmark: nbody
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.12, N = 3
41.0
Python 3.9.1
SE +/- 0.32, N = 3
88.8
Python 3.8.6
SE +/- 0.21, N = 3
88.9
20
40
60
80
100


PyPerformance 1.0.0
Benchmark: django_template
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.00, N = 3
24.2
Python 3.9.1
SE +/- 0.00, N = 3
34.6
Python 3.8.6
SE +/- 0.00, N = 3
34.7
8
16
24
32
40
PyPerformance 1.0.0
Benchmark: float
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.03, N = 3
48.3
Python 3.9.1
SE +/- 0.13, N = 3
81.2
Python 3.8.6
SE +/- 0.09, N = 3
81.3
20
40
60
80
100
PyPerformance 1.0.0
Benchmark: chaos
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.07, N = 3
38.6
Python 3.9.1
SE +/- 0.09, N = 3
80.9
Python 3.8.6
SE +/- 0.25, N = 3
81.4
20
40
60
80
100

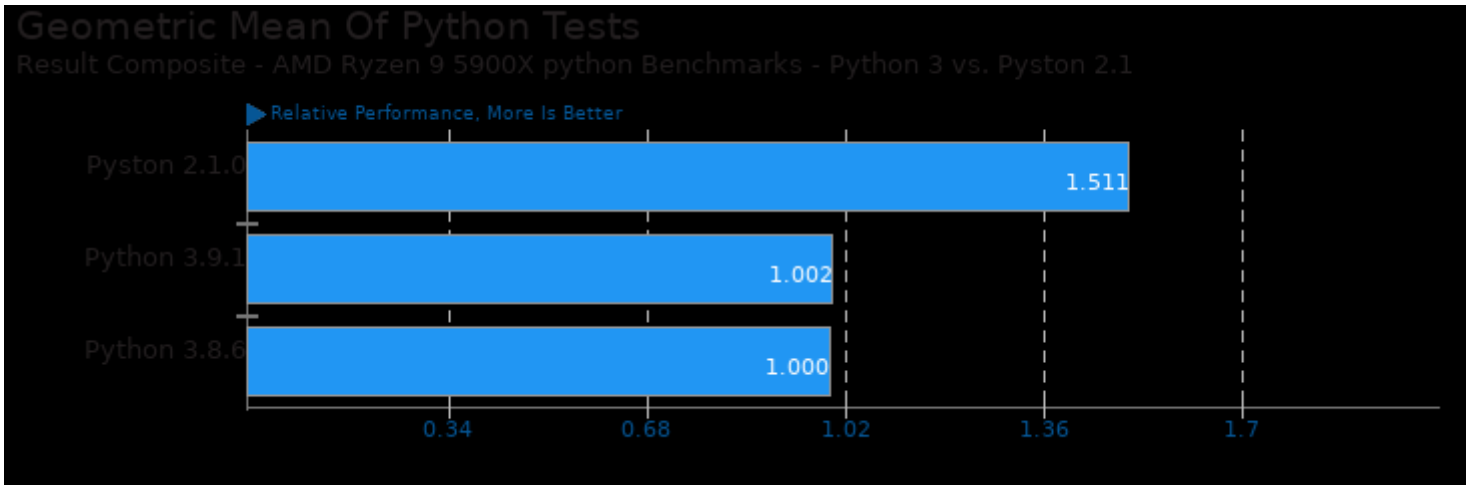

PyPerformance 1.0.0
Benchmark: pathlib
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.03, N = 3
11.0
Python 3.8.6
SE +/- 0.00, N = 3
12.7
Python 3.9.1
SE +/- 0.00, N = 3
12.7
3
6
9
12
15
PyPerformance 1.0.0
Benchmark: regex_compile
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.10, N = 3
66.2
Python 3.8.6
SE +/- 0.33, N = 3
123.0
Python 3.9.1
SE +/- 0.33, N = 3
123.0
30
60
90
120
150
PyPerformance 1.0.0
Benchmark: crypto_pyaes
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 0.06, N = 3
51.1
Python 3.8.6
SE +/- 0.07, N = 3
77.0
Python 3.9.1
SE +/- 0.15, N = 3
77.1
20
40
60
80
100


PyPerformance 1.0.0
Benchmark: pickle_pure_python
Milliseconds, Fewer Is Better
Pyston 2.1.0
198
Python 3.9.1
SE +/- 0.88, N = 3
315
Python 3.8.6
SE +/- 0.88, N = 3
317
70
140
210
280
350
PyBench 2018-02-16
Total For Average Test Times
Milliseconds, Fewer Is Better
Pyston 2.1.0
SE +/- 1.20, N = 3
464
Python 3.9.1
SE +/- 6.89, N = 3
722
Python 3.8.6
SE +/- 10.07, N = 3
727
160
320
480
640
800

These geometric means are based upon test groupings / test suites for this result file.



Geometric mean based upon tests: pts/pyperformance and pts/pybench



Geometric mean based upon tests: pts/pybench and pts/pyperformance

This file was automatically generated via the Phoronix Test Suite benchmarking software on Friday, 29 March 2024 09:12.