<https://docs.python.org/3/tutorial/stdlib.html>

10.10. Performance Measurement

Some Python users develop a deep interest in knowing the relative performance of different approaches to the same problem. Python provides a measurement tool that answers those questions immediately.

For example, it may be tempting to use the tuple packing and unpacking feature instead of the traditional approach to swapping arguments. The [timeit](https://docs.python.org/3/library/timeit.html" \l "module-timeit" \o "timeit: Measure the execution time of small code snippets.) module quickly demonstrates a modest performance advantage:

>>>

**>>> from** **timeit** **import** Timer

**>>>** Timer('t=a; a=b; b=t', 'a=1; b=2').timeit()

0.57535828626024577

**>>>** Timer('a,b = b,a', 'a=1; b=2').timeit()

0.54962537085770791

In contrast to [timeit](https://docs.python.org/3/library/timeit.html" \l "module-timeit" \o "timeit: Measure the execution time of small code snippets.)’s fine level of granularity, the [profile](https://docs.python.org/3/library/profile.html#module-profile) and [pstats](https://docs.python.org/3/library/profile.html" \l "module-pstats" \o "pstats: Statistics object for use with the profiler.) modules provide tools for identifying time critical sections in larger blocks of code.