Are tuples more efficient than lists in Python?

Tuples offer some performance advantages explained by Python core developer Raymond Hettinger in a StackOverflow answer to the question: "Are tuples more efficient than lists in Python?". To summarize, Hettinger wrote:

- To evaluate a tuple literal, the Python compiler generates bytecode for a tuple constant in one operation; but for a list literal, the generated bytecode pushes each element as a separate constant to the data stack, and then builds the list.
- Given a tuple `t`, `tuple(t)` simply returns a reference to the same `t`. There's no need to copy. In contrast, given a list `l`, the `list(l)` constructor must create a new copy of `l`.
- Because of its fixed length, a `tuple` instance is allocated the exact memory space it needs. Instances of `list`, on the other hand, are allocated with room to spare, to amortize the cost of future appends.
- The references to the items in a tuple are stored in an array in the tuple struct, while a list holds a pointer to an array of references stored elsewhere. The indirection is necessary because when a list grows beyond the space currently allocated, Python needs to reallocate the array of references to make room. The extra indirection makes CPU caches less effective.