A typical dev would bemoan the constantly changing python, etc libraries as costs in maintaining OOP functions over time. But is this conclusion appropriate, even in classical models? Functional prog suffers from path-dependence. The python changing libraries (OSS) are often a vibrant result of shared collaboration, and necessary or optimizing code, voted-in choices. Functional prog. can be lazier and prevent new documentation from keeping a diversified codebase. Functional prog is limited in scope to what your limit of (ossified) knowledge is at the time, which can prevent new methods from being introduced simply from ignorance of having to re-review documentation. In OOP, therefore updating alpha in maintaining new code structure, in line with tested best python libraries practices, allows the best chances of automatically fixing many prior hidden bugs (you haven't uncovered) and edge cases from updating libraries, while being (mostly) only the most necessary changes at the library level. Often voting decisions are well-reasoned wishes of the aggregate audience so there is a value in their updating documentation, not captured in unchanging functionalized code.

In ML, DL, AI, GA,… this is particularly evident. Popular OSS for supporting e.g. cvxopt, will update with possibly new functionality as a new combined source of alpha (changes over time as new comb alpha sources). We can capture this optionality by simply keeping in-line with the latest standards in tech stacks; that automatically forces you to stay abreast on the latest bleeding-edge techniques.