In implementing and developing trading strategies, both Python and C/C++ have their own strengths, and the most appropriate one is relative to your objective, performance needs, and development environment.

Python is the prevailing language for quantitative research and algorithmic strategy development in academia and industry. Its simplicity, concise syntax, and extensive ecosystem make it particularly well-suited for rapid prototyping, data analysis, and signal generation. Python has an extensive range of libraries like pandas and NumPy for data manipulation, TA-Lib, finta, and backtrader for technical analysis and backtesting, and scikit-learn, xgboost, and PyTorch for machine learning-based strategies. Most broker and exchange APIs—including Alpaca, Interactive Brokers, Binance, and Finnhub—provide official Python bindings, making trading code integration with real-time market data and order execution straightforward. For the majority of traders operating at the minute or second level, Python is more than adequately performant, and libraries such as asyncio, websockets, and FastAPI facilitate responsive real-time systems. It's also very suitable for paper trading and live execution when sub-millisecond accuracy is not needed.

C/C++, although lacking libraries and compatibility with APIs, it is a more suitable choice for latency-sensitive systems where every microsecond matters. In HFT, scalping, or market-making strategies, speed of execution is the most critical factor, and C++ is the right language for processing millions of messages per second, developing order book engines, and connecting directly to exchange-level protocols such as FIX or ITCH/OUCH. C++ provides direct control over memory, threading, and optimization tweaks, making it perfect for the development of trading infrastructure, low-latency gateways, and custom matching engines. The development process, though, is more time-consuming and error-prone because of the language's complexity and the absence of high-level abstractions for financial data and indicators.

In reality, most companies and individual developers take a hybrid approach. They implement high-performance pieces such as execution engines, feed handlers, and latency-critical analytics in C++, and do research, modeling, and orchestration in Python. pybind11, Cython, and Boost.Python are examples of tools that enable Python to invoke C++ functions, making it easy to mix speed and flexibility. This pattern is particularly prevalent in hedge funds and prop trading firms, where quant researchers implement in Python and execution systems are implemented in C++.

In short, Python is suitable for the majority of strategy development phases—particularly backtesting, research, machine learning, and API-based trading—due to its ease of use and ecosystem. C++ must be utilized when performance can't be sacrificed, such as in ultra-low-latency or infrastructure-level parts. Combined, they comprise a robust fusion that trades off ease of development against execution speed and are thus the two most significant languages in contemporary algorithmic trading.