Marc Kendal

Python Engineer — Algorithmic Trading · Real-Time Systems · Performant Code London, UK
marckendal@gmail.com · +44 7939 130 443 marckendal.com · github.com/MKToronto

---

Summary

Python engineer drawn to the work where the design doesn't exist yet. Since May 2025, the full-time work has been a solo-built algorithmic trading platform (~200 modules, Python / asyncio, live across Interactive Brokers, IG, OANDA, and Alpha Vantage), with a private Claude Code plugin operationalising Marcos López de Prado's three books on quantitative ML alongside it. Day-to-day workflow is spec-first with AI coding agents; reviews every change before commit. Before that, owned the Python computer-vision layer inside Fortress Technology's flagship ICON X-ray inspection machine, deployed at scale across Fortress's installed base on Raspberry Pi edge hardware alongside legacy C / OpenGL / Perl. **Every piece of code shipped at Fortress went out with zero post-ship bugs and no customer complaints.**

Experience

Founding Engineer — Automated Trading Platform (independent R&D) · May 2025 – Present

Sole architect and engineer of a production-grade algorithmic trading platform in Python / asyncio — approximately 200 modules covering execution across four broker APIs, risk management, and strategy iteration. Live against real capital; an ML-ops layer filtering signals before they become orders is under active development. *(Following paternity leave, Sep 2024 – April 2025.)*

- Architected a single exchange-abstraction protocol behind which Interactive Brokers, IG, OANDA, Alpha Vantage, and a mock venue all implement; identical strategy code runs against live, paper, or mocked venues through a single switch.
- Owned execution end-to-end: order lifecycle management, position reconciliation against broker reality, corrective adjustment when local records diverge from live fills.
- Owned risk management: timezone-aware overnight guard auto-closing positions before broker swap fees, shadow / paper-trade mode for safe hypothesis exercise, ML-gated signal filtering, adaptive rate limiting against broker API pressure.
- Iterated the strategy framework through five generations against real trade logs using a registry-pattern signal composition and per-strategy balance tracking across candle, momentum, mean-reversion, and spread families.
- Built a separate FastAPI monitor sub-app with live PnL / Sharpe / drawdown dashboards, WebSocket updates, Pushover critical-alert integration, trade-puller, and analytics worker.
- Architected and directed a **private** Claude Code plugin (`quant-finance`) — seventeen skills and a dedicated subagent operationalising Marcos López de Prado's three books (*Advances in Financial ML*, *ML for Asset Managers*, *Causal Factor Investing*) — used inside the trading research. Covers triple-barrier labelling, fractional differentiation, covariance denoising with Marcenko-Pastur bounds, causal-factor validation with Monte Carlo experiments, the seven sins of backtesting, meta-labelling, sample weighting, microstructural features. Claude-assisted authorship; architecture, curation, and validation owned directly.
- Architected and shipped a **published** Claude Code plugin (`python-clean-architecture`) on the Claude Code marketplace, dual-distributed for Codex CLI — twelve commands plus skill pack plus FastAPI example. Installable by any engineer working with Claude Code.

Research Engineer — Fortress Technology (Toronto) · Oct 2019 – Aug 2024

Owned the Python computer-vision and data-ingestion layers inside Fortress's industrial-inspection product line, with the flagship ICON X-ray inspection machine deployed across Fortress's installed base (many hundreds of production facilities). **Zero post-ship bugs and no customer complaints across shipped work.**

- Owned the real-time image-processing pipeline on Raspberry Pi edge hardware, alongside legacy C / OpenGL / Perl: C detector code writing into SysV shared memory; Python reading via `sysv_ipc` with byte offsets computed programmatically from a numpy dtype mirror of the C struct (`extract_constants.py` parses the C header and caches to JSON — keeps the C ↔ Python contract from drifting silently); numba-JIT hot paths (`@jit(nopython=True, parallel=True, fastmath=True)` with `prange`) including adaptive-geometry rendering with **sqrt-scaled brush radius** under performance pressure and recursive hole-filling with configurable stop thresholds; atomic `hardlink_to` → `.replace()` file publishing so the UI never sees a half-written image; FastAPI / WebSocket layer coordinating nine distinct asyncio locks over disjoint state regions, with a ThreadPoolExecutor / asyncio bridge bringing SHM polling into the async server. Telemetry-driven performance engineering — Pearson-correlation analysis (`stats_on_production.py`) to find real bottlenecks rather than guess them. *(product demo)*
- Led on-device contaminant-detection modelling work (bone and metal classification), including a coordinate-auto-generation function that made object-detection training viable on production-shaped data and directly influenced hardware design of the next generation of machines.
- Made the strategic call to invest in labelled-data acquisition at R&D-leadership level, then designed and built the FastAPI + Svelte + Redis-queue data-ingestion tool (`image_dev_tool_fastapi`) that pulled time-windowed image batches from fielded machines over SSH and gave operators a UI to categorise contaminants — the dataset foundation every downstream anomaly-detection model relied on.
- Built the SvelteKit operator UI running on every shipped ICON X-ray (`xray_streaming`) — kiosk-mode Chromium served via `sirv` on Pi, with a full offline-deployable installation pipeline. *(Fortress ICON product page)*
- Founding engineer on the Knapp pill-dispensing product — Flask control system shipped through external-customer handoff on a first Fortress project with zero post-ship bugs.
- Mentored junior engineers on the React + Python auth stack through pair programming and code review; 150+ bugs closed together with measurable skill growth.

Software Engineer — British Airways (London, UK) · 2014 – 2016

First engineering role. Test-automation for ba.com: Python aspect-oriented-programming code that intercepted live data streams to give tests realistic inputs drawn from real site traffic; Selenium / Cucumber / Concordion BDD suites across desktop and mobile; co-authored a "date-rolling" service that kept stubbed test data valid as calendar dates advanced — adopted across all teams.

Selected Projects

- **`quant-finance`** — Private Claude Code plugin operationalising Marcos López de Prado's trilogy into seventeen skills + five commands + a dedicated quant-expert subagent. Used inside the live trading research. Skills cover triple-barrier labelling, fractional differentiation, covariance denoising, causal-factor validation, the backtest seven-sins checker, and related quant-ML methodology.
- **Automated trading platform** — ~200-module Python / asyncio system; four-broker execution with unified Protocol abstraction; custom risk-management layer; strategy framework through five generations; live. Described above.

- `python-clean-architecture` — Published Claude Code marketplace plugin, dual-distributed to Codex CLI. Twelve commands + skill pack + FastAPI hotel-api example.
- `localytics` — Published self-hosted alternative to GitHub Insights with **function-level cyclomatic complexity** (Radon), weekly / monthly / yearly activity heatmaps, and per-file churn. Two FastAPI apps — local scanner + Render-hosted dashboard (FastAPI + Redis) — exchanging only aggregated numbers; source stays on the machine. Built to track the trading platform's development; MIT-licensed. Live demo.
- `docugit` — Published CLI extracting detailed git code-changes over configurable time-ranges for R&D tax-credit report generation via LLMs.
- **Shiurim transcription pipeline** — Whisper + SpeechBrain speaker identification + Claude Code CLI content classification over a large Dropbox audio corpus; ChromaDB-backed natural-language query plugin (`triebitz-halacha`) over decades of Rabbi Triebitz's lectures.

Tech

Primary: Python · C (ctypes + shared memory) · JavaScript / Svelte / SvelteKit **Scientific Python:** NumPy · SciPy · Pandas · numba · scikit-learn · FastAI · PyTorch **Systems / Infra:** FastAPI · asyncio · Docker · SQLite · Redis · WebSockets · systemd · mamba / conda · Raspberry Pi · paramiko / SCP · memray profiling · SysV shared memory **Familiar (concept-equivalent tooling experience, applied at smaller scale):** Airflow (use RQ / asyncio for equivalent workflow orchestration), Kafka (use Redis / websockets for equivalent streams), Kubernetes (use systemd / Docker-compose for equivalent orchestration) **Quant methodology:** Lopez de Prado-style labelling, fractional differentiation, causal-factor inference, backtesting validation (seven sins), sample weighting, microstructural features **Working practice:** Agile · CI/CD · code review · unit testing · refactoring · spec-driven, agent-orchestrated development · review every change before commit · Claude Code + Codex CLI day-to-day

---

Education

MSc Data Science · City, University of London · 2016 – 2019 Dissertation (Computer Vision): *Improving the Optical Character Recognition of Middle Eastern Languages* — PyTorch CNNs, convolution analysis and heat-maps, classifier interpretation.

MSc Philosophy of the Social Sciences · London School of Economics Certificate of attendance. Left to pursue formal philosophy studies in Jerusalem (see below).

PGDip Philosophy · The Heiden Institute

BSc (Hons) Economics, 2:1 · University of Nottingham · 2005 – 2008
