Marc Kendal

Builder & Technologist — Python · Performant Code · Real-Time Systems 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 — figuring out what the thing should be before building it. Owned the Python image-processing layer inside Fortress Technology's flagship ICON X-ray inspection machine, deployed at scale across Fortress's installed base on Raspberry Pi edge hardware. Since 2025, solo-building an algorithmic trading platform to optimise personal savings returns — live across four broker APIs. Works spec-first with AI coding agents — plans before prompts, reviews every change before commit. **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, built to optimise personal savings returns — approximately 200 modules covering execution across four broker APIs, risk management, and strategy iteration. Live against real capital; ML-ops layer filtering signals before they become orders 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.
- Iterated the strategy framework through five generations against real trade logs using registry-pattern signal composition and per-strategy balance tracking across candle, momentum, mean-reversion, and spread families.
- Built a separate FastAPI monitor sub-app: live PnL / Sharpe / drawdown dashboards, WebSocket updates, Pushover critical-alert integration, trade-puller, analytics worker.
- Architected and directed two Claude Code plugins — `python-clean-architecture` (published on the Claude Code marketplace, dual-distributed to Codex CLI; twelve commands + skill pack + FastAPI example) and `quant-finance` (private, used inside the trading research; seventeen skills + dedicated subagent operationalising Marcos López de Prado's three books on quantitative ML). Claude-assisted authorship; architecture, curation, and validation owned directly.

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

Owned the Python computer-vision and data-ingestion layers across Fortress's industrial-inspection product line, with the flagship ICON X-ray machine deployed across their 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 inside the ICON X-ray on Raspberry Pi edge hardware, alongside legacy C / OpenGL / Perl code already on the machine: 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, keeping both sides from drifting); numba-JIT-compiled hot paths (`@jit(nopython=True, parallel=True, fastmath=True)` with `prange`) processing the image stream with adaptive-geometry rendering (sqrt-scaled brush radius under performance pressure) and recursive hole-filling with configurable stop thresholds; atomic `hardlink_to` → `.replace()` 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`) finds real bottlenecks empirically rather than guessing. *(product demo)*
- Led the on-device contaminant-detection modelling work (bone and metal), including a coordinate-auto-generation function that made object-detection training viable on production-shaped data and directly influenced the 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 X-ray machines over SSH and gave operators a Svelte 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`) — multi-camera live views served via `sirv` inside kiosk-mode Chromium on Pi, backed by a full offline-deployable installation pipeline (conda environment, screen-session orchestration, boot-time startup, field-installable without internet access). Iterated through seven streaming architectures before landing on the kiosk design. *(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, including user-editable calibration for dispensing precision.
- Owned the multilingual HMI translations pipeline (`Translations_v3`) — bidirectional JSON ↔ Excel tool synchronising 40+ languages across every Fortress machine, with locked-cell UX for non-technical translators, language auto-detection, and Wordhoard synonym integration with rate-limit resilience.
- Mentored junior engineers on the React + Python auth stack through pair programming and code review; 150+ bugs closed together with measurable skill growth on their side.

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

- Built live-data interception code for ba.com in Python (aspect-oriented programming / AOP) — captured real production data streams so tests ran against realistic live inputs instead of synthetic stubs.
- Built Selenium / Cucumber / Concordion BDD suites for desktop and mobile ba.com releases.
- Co-authored a "date-rolling" Python service that kept stubbed test data valid as calendar dates advanced — adopted across all teams.

Selected Projects

- `python-clean-architecture` — Published Claude Code marketplace plugin, dual-distributed to Codex CLI. Twelve commands + skill pack + full FastAPI hotel-api example.
- **Automated trading platform** — ~200-module Python / asyncio system; four-broker execution; live. Described above.

- `quant-finance` — Private Claude Code plugin operationalising López de Prado's trilogy into seventeen skills + a dedicated quant-expert subagent. Used inside the trading research.
- `localytics` — Published self-hosted alternative to GitHub Insights with **function-level cyclomatic complexity** (via Radon), weekly / monthly / yearly activity heatmaps, and per-file churn. Two FastAPI apps — a local scanner + a Render-hosted dashboard (FastAPI + Redis) — that only exchange aggregated numbers; source code never leaves the machine. Built to track the trading platform's development without pushing private source to hosted platforms; MIT-licensed. Live demo.
- `docugit` — Published CLI that extracts detailed git code-changes over configurable time-ranges for R&D tax-credit report generation via LLMs.
- **Shiurim transcription pipeline** — Whisper + SpeechBrain speaker ID + Claude Code CLI 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 **Day-to-day:** FastAPI · asyncio · numba · PyTorch · OpenCV · Docker · Git · Claude Code / Anthropic SDK **Systems:** SysV shared memory · Redis / RQ · SQLite · WebSockets · systemd · mamba / conda · Raspberry Pi · paramiko / SCP · memray **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.

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

PGDip Philosophy · The Heiden Institute
