🤖 AI-Driven Learning

This overview merges a detailed course introduction that stresses **project-based learning** with a brief workshop guide on leveraging **large-language models** to auto-create Python implementations of **system-dynamics** models.

📚 Content Sources

- Section 1: Highlights the **project-based learning** approach and **instructor profile** of an 18-hour course from a 104-minute audio recording. (source)
- Section 2: Shows a rapid workshop on using **large-language models** to produce **system-dynamics** Python code from a 5-minute audio recording. (source)

Section 1: Course Philosophy & System-Dynamics Foundations 🎯

This section covers comprehensive course introduction, ML critique, and system-dynamics modeling from a 104-minute audio recording (source)

1.1 Course Blueprint & Instructor Profile

Element	Detail
Total duration	18 h (first 104 min recorded)
Format	1 day concepts/demos → 4 days (≈14 h) building a **system-dynamics model** in small groups
Certification	2-hour Microsoft formation + 2-hour Microsoft certification on Friday afternoon
Instructor	**Mehdi Munsi**, PhD Reinforcement Learning, 5 y consulting for military & industrial clients

Philosophy	Project-based, hands-on; bridge research → industrial AI adoption

> *Learning goals*: (1) See why benchmark-centric ML fails in production, (2) Design **causal feedback models**, (3) Build a **Gen-AI tool** that accelerates model creation/editing.

1.2 Student Lens on AI 🔍

When you hear "AI" what surfaces?

- **Assistant**, not magic: pragmatic, tool-oriented view.
- **Deployment pain-points**: UI design is hard; free-text prompts are flawed.

Already-built projects

1. **Food-waste chatbot**: fridge photo → expiration checker → recipe & shop API; deployed on Render + Streamlit.
2. **Chess analyzer**: YOLO-v8 board detection + LLM explanations; custom-tuned 3 B model for speed.

Market anxiety

- Accenture/Capgemini layoffs highlight need to **stay updated**; classical coding may shrink; **AI fluency** becomes core.
- Companies already **pay for every employee** to use Cursor-like assistants.

1.3 Benchmark-Centric ML: Four Traps 🚫

1. **99 % benchmark ≠ production ready**
2. **98 → 98.5 % chase** burns weeks for negligible value
3. **Model ≈ 5 % of system**; missing **monitoring, UI, recovery**
4. **Beautiful model + awful UX = 0 adoption**

> *Industrial AI needs systems-thinking, not leaderboard climbing.*

1.4 System-Dynamics Primer 🌐

> **Definition**: A mathematical framework encoding **causal, feedback-driven relationships** via stocks, flows & ODEs.

Core Vocabulary

Term	Meaning	Emoji
Stock	Accumulating quantity (trees, debt, reputation)	📦
Flow	Rate changing a stock (birth, issuance, decay)	🚰
R-loop	Reinforcing (amplifies change)	🔴
B-loop	Balancing (dampens change)	🟢

Canonical Example 1 – Canadian Spruce-Budworm 🌲🐛

- **Stocks**: Trees, Budworms, Predators
- **Dynamics**: Spruce dominance → budworm feast → outbreak; insecticide kills predators → **threshold system**—tiny shock can tip catastrophe.
- **Lesson**: Short-term spray ignores **hidden feedbacks**; system model exposes trade-offs.

Canonical Example 2 – Police Incentives & Trans Sex-Work 🚔

- **Stocks**: #SexWorkers, Debt, Tickets, PolicePoints, Reputation
- **Red R-loop**: More sex-work → more stops → fines → debt → more sex-work
- **Blue R-loop**: Tickets → promotion pressure → more tickets
- **Green B-loops**: Workers learn rights; police adapts tactics (delays)
- **Insight**: Policy incentives can create **runaway dynamics** if loops misunderstood.

1.5 Formal Modelling Ingredients 🧮

Equation form

Ordinary Differential Equations (ODEs) → simulate in Python (scipy.integrate.odeint)

Example – Reputation Capital Index (RCI)

> $dRCI/dt = -\alpha \cdot RCI + \gamma \cdot Visibility \cdot PartnerScale$

- **α**: decay constant | **γ**: partner boost | both estimated via **domain knowledge or Bayesian priors**

Bayesian mindset
Treat every constant as **probability distribution**; propagate uncertainty; update with data (Twitter sentiment, sales, etc.)

1.6 Project Assignment 🛠️

Deliverable
A **Gen-AI-assisted tool** that removes friction in creating/editing system-dynamics models (minimum: textual description → editable JSON of stocks/flows → simulation plot).

Business Cases

Team	Question	Key loops
AéroDIN	Invest in AI-controlled lethal weapons?	PublicBacklash ↔ Regulation ↔ Secrecy
Euromotion Automotive	How do tariffs & lockdowns affect supply?	Inventory ↔ TariffDelay ↔ PoliticalRisk

Required model elements

1. **Stocks** (e.g., PublicBacklash, RegulatoryPressure)
2. **Flows** (BacklashGrowthRate, PolicyChangeRate)
3. **R & B loops** with documented delays
4. **Constants** with priors & data source

1.7 Tool Design Checklist ✅

Requirement	Rationale
Low-bandwidth input	Single click/pdf paste beats giant prompts
Human-editable JSON	Rapid iteration & error correction
Provenance tags	Auditability for execs
Quick simulation button	Prove model runs, not just diagram
Bayesian priors suggestion	Handle scant data rigorously

Extensible schema	Fields: stock, flow, equation, source, type(R/B), delay

Suggested dev loop

1. Build toy model manually (RCI example)
2. Prompt LLM to generate Python integrator
3. Extract JSON structure
4. Wrap in Streamlit/Gradio UI
5. Validate against micro-dataset & document limitations

1.8 Broader Reflections 🌐

AI labour market
Pyramid → **Diamond**: fewer senior roles, many mid-level; **AI fluency** becomes differentiator.

Board-level value map

- Internal efficiency
- External margin & speed
- Brand association
- **Measurable KPIs** via InternalEfficiencyMultiplier

Gen-AI strength/weakness
✅ Fast execution, code & data suggestions
❌ Poor 2nd-order thinking; needs human **validation & refinement**

Section 2: LLM-Driven Rapid Prototyping Lab ⚡

This section covers hands-on LLM workflow for instant system-dynamics code generation from a 5-minute audio recording (source)

2.1 Session Objective

Produce **easily-modifiable Python code** that implements any system-dynamics model by chaining two LLM calls—no manual equation writing.

2.2 Two-Step LLM Workflow

1. **Conceptual Session**
    - Ask **any open question** with potential feedback loops
    - *Example prompt*: "What would be the mechanism to avoid a war in Kazakhstan?"
    - Capture the **textual causal narrative**
2. **Code-Generation Session (separate)**
    - Feed narrative into new LLM window
    - *Example follow-up*: "Write the system-dynamics model in Python that displays that."
    - Receive runnable Python (uses numpy, scipy.integrate, or custom interpreter)

> *Key rule*: **Never mix prompts in one chat**—keeps context clean & reproducible

---

2.3 Recommended Stack

Layer	Tool	Why
Core logic	**Python**	Universally known, integrates with science libs
Plotting	**Matplotlib**	Quick time-series or phase-space visuals
Optional UI	**JavaScript / React**	If interactive web dashboard desired

Speaker mantra: "Use Python everywhere" to minimise friction.

2.4 Live Exercise Instructions

1. **All participants** open **two LLM tabs** (ChatGPT, Claude, local API—any)
2. In tab-1 ask **any system-dynamics question** (economics, ecology, conflict, etc.)
3. Copy answer → tab-2 → prompt "Write Python system-dynamics code for this."
4. **≈ 2 minutes** allotted; instructor circulates to confirm success

---

2.5 Reference Material

- A **pre-existing Python script** already encodes canonical equations (shown earlier)
- **Eurodin** demo model serves as quality benchmark for generated code

---

2.6 Logistics & Announcements (French segment)

Item	Detail
Carrière Week	Next week, 5 days, multiple daytime events
Research / CDI	Check **ESN** (Écoles Supérieures du Numérique) portal for opportunities
Communication	Email + WhatsApp; **answer promptly**—response count drives event scale
Attendance	**Physical preferred**; fallback → **Teams** (camera + mic **mandatory**, or be muted)
Survey	Short questionnaire incoming to coordinate slots

2.7 Quick Take-away

By **piping narrative → code across two LLM sessions**, participants can **stand-up an executable system-dynamics prototype in minutes**, merging **immediate technical practice** with upcoming **career-week engagement**.

🎯 Putting it All Together

🌐 Shared Vision

> *System dynamics* is presented as the **missing bridge** between isolated machine-learning benchmarks and real-world decision making.
> Both recordings stress that **human-centred AI** must be embedded in a *causal, feedback-driven* representation of the problem domain before any model is deployed.

- **Why benchmarks fall short** – high accuracy on static datasets does not guarantee production reliability.

- **What replaces them** – stocks, flows, and reinforcing/balancing loops that expose hidden risks (e.g., public backlash, supply-chain disruptions).
- **The end goal** – a **modular, editable Python implementation** that can be visualised, simulated, and iterated quickly.

🤖 LLM-Driven Prototyping Pipeline

Step	104-min audio focus	5-min audio focus
1️⃣ Concept capture	Students discuss AI perceptions, then identify a concrete business question (e.g., "Should we invest in AI-controlled weapons?").	Prompt any LLM with an open-ended system-dynamics question ("How avoid war in Kazakhstan?").
2️⃣ Textual description	Instructor emphasises **system-thinking**: write out variables, loops, and causal hypotheses.	Take the LLM's natural-language answer as the *qualitative model*.
3️⃣ Code generation	Use generative-AI-assisted tool to turn the description into **Python ODE code** (stocks/flows → scipy.integrate.odeint).	Pipe the description into a **second LLM session** requesting "Python code that implements the system-dynamics model."
4️⃣ Edit & validate	Teams manually adjust parameters, add Bayesian priors, and run simulations against real data (e.g., sentiment scores).	Participants edit the generated script, run a quick plot with *Matplotlib*, and note any mismatches.
5️⃣ Deploy & iterate	Wrap the model in a **Streamlit** UI, expose as a web service, and iterate based on stakeholder feedback.	Optionally connect the script to a simple front-end (JavaScript/React) for interactive demo.

🛠️ Recommended Tool Stack

- **Python** – core language for ODE definition, simulation, and data handling.
- **Matplotlib** – quick visualisation of time-series trajectories.
- **Streamlit / Gradio** – minimal UI for parameter tweaking and result display.
- **JavaScript / React** – optional layer for richer interactive dashboards.

- **LLM APIs (OpenAI, Azure, etc.)** – power the two-step prompt workflow.

📚 Pedagogical Flow (From Theory to Practice)

1. **Build a toy model** (e.g., Reputation-Capital Index) **by hand** to internalise stocks, flows, and loops.
2. **Prompt an LLM** for a narrative description of a new problem domain.
3. **Generate Python code** automatically, then **refactor** the JSON-like data structure (stocks, flows, equations).
4. **Run a simulation**, visualise outcomes, and **compare** against any available data.
5. **Iterate**: edit JSON, re-prompt LLM for adjustments, re-run.
6. **Document provenance** – each element records its source (article, LLM output, data set) for auditability.

📈 Business Cases & Modelling Elements

Case	Core Stocks	Core Flows	Typical Reinforcing Loop	Typical Balancing Loop
AéroDIN (defense)	WeaponProductionCapacity, PublicBacklash, RegulatoryPressure	R&DInvestmentRate, BacklashGrowthRate, PolicyChangeRate	More weapons → higher backlash → stricter regulation → secrecy → more covert weapons	Public education → reduced backlash
Euromotion Automotive	InventoryStock, TariffImpact, ProductionCapacity	SupplyDelayRate, DemandFluctuation, InvestmentInResilience	Tariffs ↑ → supply delays ↑ → inventory depletion → price hikes → demand drop → further delays	Investment in alternative suppliers → stabilises inventory

Each model follows the **JSON schema** suggested in the 104-min session (fields: stock, flow, equation, source, type, delay), ensuring **interoperability** between teams.

🧩 Skills & Market Insights

- **AI fluency** now eclipses pure coding; ability to **prompt, validate, and steer LLMs** is a core competency.

- **System-dynamics literacy** differentiates senior contributors who can foresee second-order effects from those who only optimise isolated metrics.
- **Job market shift** – from a pyramid of junior-heavy roles to a **diamond** shape where mid-level expertise in AI-augmented systems design is scarce and high-value.
- **Board expectations** – measurable internal efficiency gains, external revenue uplift, and brand reputation improvements, all traceable to the **system-dynamics-informed AI roadmap**.

📌 Practical Take-aways

- Adopt the **two-session LLM workflow** for any new domain: first capture the causal story, then let the model write the ODE code.
- Keep the **codebase modular** (separate JSON definition, simulation engine, UI layer) to enable rapid experimentation.
- Use **Bayesian priors** for uncertain parameters; update them with observed data to maintain a transparent uncertainty budget.
- Record **source provenance** for every stock/flow to satisfy both internal audit and external stakeholder scrutiny.
- Leverage the **same Python-centric stack** across prototypes, demos, and production deployments to minimise context switching.