APPLIED AI ENGINEERING ACADEMY

Reinforcement Learning

From Markov Decision Processes and Control to PPO, Continuous Control and RLVR

3 DAYS | 21 CONTACT HOURS | HANDS-ON INTENSIVE

A mathematically grounded and implementation-oriented course that connects classical sequential decision-making, control, tabular learning, policy gradients and modern reinforcement learning practice.

AUDIENCE	DELIVERY	CAPSTONE
ML engineers, researchers, simulation engineers, robotics developers and advanced students.	Algorithms from scratch where pedagogically valuable, frameworks where implementation complexity would hide the ideas.	A small Gymnasium environment with baseline control, trained policy, ablation and multi-seed evaluation.

Feasibility verdict: Feasible only as an intensive foundation course. Tabular RL and REINFORCE are implemented; PPO and continuous control use libraries; RLVR is a verifier and training-loop practicum, not a full large-model run.

Version 1.0 | Framework review date: 1 August 2026

Course overview

The course gives participants a coherent path from Markov decision processes and Bellman reasoning to policy gradients, PPO, reward design and continuous control. It also explains where current LLM post-training and reinforcement learning with verifiable rewards fit into the wider RL landscape.

Learning outcomes

- Formulate a problem as an MDP with explicit observations, states, actions, transitions, rewards, termination and truncation.
- Explain returns, value functions, action values, Bellman equations, policies, advantages and the exploration-exploitation trade-off.
- Implement value iteration, SARSA and Q-learning and compare on-policy with off-policy behaviour.
- Derive and implement a minimal REINFORCE agent and explain variance reduction, actor-critic methods and generalized advantage estimation.
- Train and evaluate PPO and a continuous-control agent using maintained libraries.
- Design reward functions, identify reward hacking and run reward or hyperparameter ablations across multiple seeds.
- Explain the relationship between RL, feedback control, LQR/MPC, offline RL, safe RL and modern LLM post-training.
- Design a verifiable reward function and interpret a small RLVR training loop without requiring a large distributed GPU job.

Realistic depth within three days

IMPLEMENT	PRACTISE	SURVEY / DEMO
• Value iteration, SARSA and Q-learning • Minimal REINFORCE policy gradient • Custom environment or reward modification	• PPO training and diagnostics • Continuous control with PPO or SAC • Multi-seed evaluation and reward ablation	• Deep control-theory derivations • Offline, model-based and multi-agent RL • Distributed RLHF/RLVR at frontier-model scale

Conditions that make the schedule feasible

- Require Python, basic probability, vectors, gradients and neural-network familiarity; provide an optional pre-course mathematics primer.
- Use small environments such as FrozenLake or CliffWalking, CartPole and Pendulum so training completes during the scheduled lab.
- Implement only the algorithms whose mechanics are central: tabular TD methods and minimal REINFORCE. Use Stable-Baselines3 for PPO and SAC.
- Provide starter environment templates, plotting utilities, seeded configurations and saved checkpoints.
- Treat RLVR hands-on work as verifier design, rollout scoring and advantage inspection; a live small-model update is instructor-led or optional.
- Start the capstone late on Day 2 and complete it on Day 3 rather than introducing a separate final project at the end.

Recommended participant profile

Primary audience. Machine-learning engineers, researchers, simulation and control engineers, robotics developers, quantitative practitioners and technically advanced students.

Prerequisites. Comfort with Python, NumPy, vectors and matrices, derivatives, expectations and basic neural networks. Participants without this background should complete a two-hour preparatory primer.

Day 1 - MDPs, Bellman Reasoning and Tabular Control

Establish the mathematical objects and implement core temporal-difference learning.

Time	Module	Content and activity
09:00-09:30	**Orientation and problem framing**	Sequential decisions, agent-environment interaction, trajectories and examples from robotics, scheduling and resource allocation.
09:30-10:30	**Markov decision processes**	States, observations, actions, transition dynamics, rewards, policies, Markov property, partial observability, termination and truncation.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**Values and Bellman equations**	Discounted return, V and Q functions, Bellman expectation and optimality equations, policy evaluation, value and policy iteration.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:15	**Control-theory bridge**	Dynamical systems, feedback, open- and closed-loop control, stability, PID, LQR and MPC as complementary approaches.
14:15-15:00	**Monte Carlo and TD learning**	Bootstrapping, bias and variance, TD error, eligibility intuition and update targets.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**SARSA and Q-learning**	On-policy versus off-policy control, exploration policies and why learned behaviour differs in risky environments.
16:00-17:30	**Lab 1: tabular RL**	Implement value iteration, SARSA and Q-learning in a small Gymnasium environment and compare policies and learning curves.

DAY OUTPUT

Working tabular implementations and a short analysis of convergence, exploration and on-policy/off-policy behaviour.

INSTRUCTOR NOTE

Control theory is a bridge, not a second full course. Limit it to the modelling assumptions and decision criteria participants need to compare classical control and RL.

Day 2 - Policy Gradients, PPO and Continuous Control

Move from value tables to neural policies and practical training pipelines.

Time	Module	Content and activity
09:00-09:30	**Recap and diagnostics**	Review Bellman and TD concepts and inspect common learning-curve failure patterns.
09:30-10:30	**Function approximation and DQN**	Neural value functions, replay buffers, target networks, instability and the role of DQN as a bridge to deep RL.
10:30-10:45	**Break**	Coffee and questions.
10:45-12:15	**Policy gradients and REINFORCE**	Log-derivative trick, stochastic policies, return-weighted updates, baselines and a minimal implementation.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:15	**Actor-critic and advantages**	Critics, temporal-difference advantages, generalized advantage estimation, entropy and the bias-variance trade-off.
14:15-15:00	**Proximal Policy Optimization**	Clipped objectives, rollout batches, epochs, value loss, entropy bonus, diagnostics and implementation details.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:00	**Reward and experiment design**	Sparse and dense rewards, potential-based shaping, reward hacking, normalization, seeds, wrappers and confidence intervals.
16:00-17:30	**Lab 2: PPO and continuous control**	Train PPO on a discrete task and PPO or SAC on Pendulum; compare stability, sample use, returns and policy entropy.

DAY OUTPUT

A minimal REINFORCE implementation, a framework PPO/SAC run and a reproducible experiment configuration.

INSTRUCTOR NOTE

DQN is explained and demonstrated, not implemented fully from scratch. This preserves time for the policy-gradient path and continuous action spaces requested for the course.

Day 3 - Evaluation, Safe RL and Modern LLM Post-Training

Evaluate policies rigorously and connect classical RL to current verifiable-reward training.

Time	Module	Content and activity
09:00-09:45	**RL evaluation**	Training versus evaluation episodes, multi-seed confidence, generalization, robustness, checkpoint selection and reproducibility.
09:45-10:30	**Continuous-control choices**	PPO versus SAC and TD3, deterministic and stochastic policies, action bounds and sample efficiency.
10:30-10:45	**Break**	Coffee and questions.
10:45-11:30	**Advanced RL landscape**	Offline, imitation, model-based and multi-agent RL: objectives, data assumptions and when each becomes relevant.
11:30-12:15	**Constrained and safe RL**	Constraints, shields, risk-sensitive objectives, unsafe exploration and reward audits.
12:15-13:15	**Lunch**	Lunch break.
13:15-14:30	**RL for language models**	SFT, preference data, reward models, PPO, DPO context, GRPO-style group baselines and agentic rollouts.
14:30-15:00	**RLVR fundamentals**	Verifiable rewards, deterministic checkers, sparse signals, verifier errors, reward hacking and suitable domains.
15:00-15:15	**Break**	Coffee and questions.
15:15-16:15	**Lab 3: verifier practicum**	Implement arithmetic or code reward functions, score grouped rollouts and inspect normalized advantages and training diagnostics.
16:15-17:00	**Capstone completion**	Finish environment, baseline, trained policy, reward ablation and multi-seed evaluation.
17:00-17:30	**Demonstrations and review**	Present policies, evidence, failure modes and decisions about the next algorithm or experiment.

DAY OUTPUT

A completed RL capstone plus a small RLVR verifier and rollout-analysis notebook.

INSTRUCTOR NOTE

A full participant-run RLVR fine-tune is not reliable in this timeframe without dedicated GPUs and substantial setup. Keep the live optimization optional and provide completed traces.

Hands-on structure and capstone

Progressive lab sequence

Lab	Focus	Participant deliverable
1	Tabular control	Value iteration, SARSA, Q-learning, exploration comparison and policy visualization.
2	Deep policy learning	Minimal REINFORCE plus PPO and continuous-control runs with reproducible configurations.
3	RLVR verifier	Custom verifiable reward functions, grouped rollout scoring and advantage diagnostics.
4	Capstone experiment	Custom or modified environment, baseline, trained policy, ablation and multi-seed evaluation.

Capstone specification

Teams create or adapt a small Gymnasium decision environment. A supplied template prevents environment boilerplate from consuming the workshop and ensures that evaluation remains the central deliverable.

- Explicit observation, action, transition, reward, termination and truncation definitions.
- A non-learning baseline such as random, heuristic or simple feedback control.
- At least one trained RL policy using an algorithm appropriate to the action space.
- One reward, exploration or hyperparameter ablation and an explanation of the expected mechanism.
- Evaluation across at least three seeds with mean, spread and failure cases.
- A reward-hacking and safety analysis, including what the policy could exploit.
- Saved configuration, learning curves, final policy and a short experiment report.

Assessment

Component	Weight
Concept checks and derivation exercises	15%
Three guided implementation labs	35%
Capstone environment, policy and evidence	35%
Technical explanation and experimental defence	15%

Framework stack and delivery operations

Primary workshop stack

Layer	Recommended framework	Why it is used in this course
Environment API	**Gymnasium**	Standard single-agent environment interface and small reference tasks.
Foundations	**NumPy + PyTorch**	Used for tabular methods and the minimal REINFORCE implementation.
Reliable baselines	**Stable-Baselines3**	Maintained PPO, DQN, SAC and TD3 implementations for practical training.
Research components	**TorchRL**	PyTorch-native collectors, buffers, objectives and modular research building blocks.
Distributed / multi-agent	**Ray RLlib**	Discussed for scaling, multi-agent systems and offline data workflows.
LLM post-training	**Hugging Face TRL**	Accessible custom reward functions and GRPO/PPO-style trainers for workshop-level examples.
Scale-out RLVR	**verl or OpenRLHF**	Surveyed for distributed, production-oriented LLM reinforcement learning.

Frameworks discussed but not all taught hands-on

- CleanRL is useful for readable single-file reference implementations, but the workshop already includes from-scratch tabular and REINFORCE code.
- PettingZoo can be introduced when the participant group specifically needs multi-agent environments.
- MuJoCo tasks are optional because system installation and compute variability can consume too much workshop time; Pendulum provides a reliable continuous-control baseline.

Operational requirements

PARTICIPANT SETUP

Laptop with Python 3.11+, NumPy, PyTorch and the pinned workshop environment. CPU is sufficient for core labs; optional GPU access supports the instructor RLVR demonstration.

INSTRUCTOR PREPARATION

Test training times on the exact machines, pin seeds and versions, provide starter environments, cached curves, saved checkpoints and a no-GPU fallback for every exercise.

CLASS SIZE

12-18 participants per instructor because debugging environments and learning instability require more individual support than standard software labs.

DATA AND SAFETY

Use simulations and toy verifiers only. No policy should control real equipment, execute untrusted code outside a sandbox or make financial or medical decisions during the course.

Framework versions should be pinned and the complete lab environment should be tested immediately before delivery. The framework recommendations reflect documentation available on 1 August 2026.

Selected current references

Official documentation and standards used to validate the framework recommendations and the current syllabus scope. Accessed 1 August 2026.

1. Gymnasium documentation — *Farama Foundation.* Current standard environment API, creation guides and termination/truncation semantics.
2. Stable-Baselines3 documentation — *DLR-RM.* Reliable PyTorch implementations including PPO, DQN, SAC and TD3.
3. TorchRL documentation — *PyTorch.* Modular environments, collectors, replay buffers, objectives and research components.
4. RLlib documentation — *Ray.* Scalable reinforcement learning, offline data and multi-agent support.
5. TRL documentation — *Hugging Face.* LLM post-training trainers, reward functions and agentic RL examples.
6. verl documentation — *verl project.* Distributed production-oriented reinforcement learning for LLM post-training.
7. OpenRLHF documentation — *OpenRLHF project.* Ray and vLLM-based scalable RLHF and RLVR training.

Syllabus maintenance note

Before delivery, benchmark every training run on the classroom hardware, verify Gymnasium API compatibility, update Stable-Baselines3 and TorchRL examples, and review the fast-moving TRL, verl and OpenRLHF APIs.

> **Important:** The course provides a strong foundation and practical workflow. It cannot provide deep mastery of control theory, every modern RL family and distributed frontier-model post-training within 21 hours.