AI 0→1 Learning Path

> Purpose: a practical, timeline-free path. For each step,
>
> **Learn the items**
>
> **Tick the checklist**

Step 1 — Python Core

Learn: syntax, control flow, functions & modules, error handling; data structures (list/tuple/set/dict), list/dict comprehensions; basic CLI scripting.

Checklist

- [ ] Write 10 small exercises covering loops, functions, slicing, and exceptions.
- [ ] Implement a CSV→JSON CLI with argparse and basic logging.
- [ ] Explain mutability, shallow vs deep copy, and when to use each container.
- [ ] Set up a virtual environment and install a third-party package.

Step 2 — Intermediate Python

Learn: file I/O for CSV/JSON/XML; OOP (classes, inheritance, dataclasses), typing; iterators/generators/decorators; context managers; packaging and environment management (uv/poetry or venv+pip).

Checklist

- [ ] Build a small I/O utility library (read/write CSV, JSON, XML) with docstrings.
- [ ] Add unit tests with pytest (cover ≥80%).
- [ ] Add pre-commit with ruff/black/isort and ensure it runs locally.
- [ ] Publish the library in a private GitHub repo with a clear README.

Step 3 — Scientific Computing & Data Stack

Learn: NumPy (arrays, broadcasting, vectorization), Pandas (indexing, groupby, merge, missing values), Matplotlib (core plots; Seaborn optional), Data Quality checks.

Checklist

- [ ] Produce an EDA notebook on a public dataset with at least 3 plots and key findings.
- [ ] Deliver a Data Quality Report (source, schema, missingness, outliers, assumptions).
- [ ] Demonstrate vectorized NumPy operations vs Python loops with timing.

Step 4 — Math Foundations (with Python)

Learn: Linear Algebra (norms, eigen/singular values intuition), Calculus/Optimization (derivatives, gradients, LR schedules), Probability & Statistics (distributions, sampling, estimation, confidence intervals, bias–variance trade-off).

Checklist

- [ ] Implement gradient descent from scratch for linear regression; compare to sklearn.
- [ ] Plot learning curves and diagnose under/overfitting.
- [ ] Explain in your own words: variance vs bias, regularization, and early stopping.

Step 5 — Classical Machine Learning (scikit-learn)

Learn: Supervised (linear/logistic, trees/forests, gradient boosting), unsupervised (k-means, PCA); feature engineering (scaling/encoding/selection); model evaluation (train/val/test, CV, metrics, leakage); Pipeline & ColumnTransformer.

Checklist

- [ ] Build an end-to-end sklearn Pipeline (preprocess→model→evaluate) with CV.

☐ Compare at least 3 models and report metrics (ROC/PR/F1) and feature importance.

☐ Provide a model card and data card; ensure a fresh clone can reproduce your score.

Step 6 — Deep Learning Basics

Learn: choose PyTorch or TensorFlow; tensors & autograd; optimizers (SGD/Adam); regularization (dropout/weight decay); scheduling; early stopping; CNNs for images; saving/loading checkpoints.

Checklist

☐ Train a small CNN on CIFAR-10 (or similar) and achieve a baseline accuracy.

☐ Save best checkpoints; plot loss/accuracy curves; create a confusion matrix.

☐ Explain vanishing/exploding gradients and remedies (init, normalization, skip connections).

Step 7 — Advanced Topics (modular, pick and combine)

7A. LLM & RAG (recommended)

Learn: chunking strategies, embeddings, vector DBs (FAISS/pgvector), retrieval→reranking→answering pipelines, evaluation (exact match, semantic similarity, citation hit rate, hallucination rate), lightweight fine-tuning (LoRA/QLoRA), inference optimization (quantization, batching, caching).

Checklist

☐ Build a PDF Q&A assistant (docs→chunks→embeddings→retrieval→answer) with an API.

☐ Create an evaluation set; report precision@k, citation hit rate, hallucination rate.

☐ Run an A/B of prompt templates or rerankers; document latency/cost trade-offs.

7B. NLP / CV / RL (choose one to deepen)

Learn:

- NLP: tokenization, classic features vs transformers, sequence classification.
- CV: OpenCV basics, modern CNN/ViT intuition, augmentations.
- RL: agents, environments, policies, reward design.

Checklist

☐ Deliver one small demo (e.g., text classifier, image classifier, or basic RL agent).

☐ Include error analysis and at least one robustness test.

7C. MLOps (strongly encouraged)

Learn: Docker, FastAPI inference service, experiment tracking (MLflow or W&B), data/model versioning, basic monitoring (latency, throughput, failure rate, drift alerts), CI with GitHub Actions.

Checklist

☐ Containerize training and inference; expose a /predict endpoint.

☐ Track experiments (params, metrics, artifacts) and keep runs reproducible.

☐ Add CI to lint/test; provide a Makefile for make setup/train/eval/serve.

Step 8 — Engineering Best Practices

Learn: Git branching; code style (PEP8 with ruff/black/isort), typing (mypy/pyright), testing pyramid (unit→integration), documentation (docstrings/README), observability (structured logs/metrics), basic performance testing.

Checklist

☐ New teammate can clone and run make setup && make train && make serve without edits.

☐ Linting and tests pass locally and in CI; type checks clean.

☐ Architecture and data-flow diagram included in README.

Self-Check Rubrics (tick when you can convincingly demo)

- Reproducibility: fresh environment can re-train, re-evaluate, and start the API.
- Metrics literacy: can choose metrics, explain AUC vs PR, avoid leakage.
- Training diagnostics: can address overfitting with concrete interventions.
- RAG quality: can measure and reduce hallucinations; provide failure cases.
- Ops readiness: images build, P95 latency within target, basic alerts wired.

Project Skeleton

```
project-name/
  ├── data/ (raw, interim, processed)
  ├── notebooks/
  ├── src/ (features, models, training, inference)
  ├── tests/
  ├── configs/ (yaml)
  ├── deployment/ (Dockerfile, compose.yaml)
  ├── Makefile
  ├── pyproject.toml (or uv/poetry)
  └── README.md (model/data cards)
```

Resources

Official Documentation (primary)

- scikit-learn User Guide — https://scikit-learn.org/stable/user_guide.html
- PyTorch Tutorials — https://pytorch.org/tutorials/
- TensorFlow Learn/Guide — https://www.tensorflow.org/learn
- Hugging Face LLM Course — https://huggingface.co/learn/llm-course

- MLflow Documentation — https://mlflow.org/docs/
- Weights & Biases Docs — https://docs.wandb.ai/
- FAISS Docs — https://faiss.ai/
- Kaggle Learn — https://www.kaggle.com/learn

Blogs (read as needed)

- Machine Learning Mastery — https://machinelearningmastery.com/
- fast.ai Blog — https://www.fast.ai/posts/
- Google AI Blog — https://ai.googleblog.com/
- OpenAI Blog — https://openai.com/research
- Towards Data Science — https://towardsdatascience.com/
- KDnuggets — https://www.kdnuggets.com/
- Analytics Vidhya — https://www.analyticsvidhya.com/blog/
- Real Python — https://realpython.com/
- DataCamp Tutorials — https://www.datacamp.com/tutorial

YouTube Channels (pick a few, not all)

- freeCodeCamp.org — https://www.youtube.com/@freecodecamp
- Corey Schafer — https://www.youtube.com/@coreyms
- Tech With Tim — https://www.youtube.com/@TechWithTim
- StatQuest with Josh Starmer — https://www.youtube.com/@statquest
- Sentdex — https://www.youtube.com/@sentdex
- Data School — https://www.youtube.com/@dataschool
- DeepLearningAI (Andrew Ng) — https://www.youtube.com/@Deeplearningai
- Two Minute Papers — https://www.youtube.com/@TwoMinutePapers
- Krish Naik — https://www.youtube.com/@krishnaik06
- Codebasics — https://www.youtube.com/@codebasics

- CS50 AI (Harvard) — https://www.youtube.com/playlist?list=PLhQjrBD2T382Nz7z1AEXmioc27axa19Kv

---