# Roadmap for Agentic AI

## 1. Python & Core Programming

|  |  |  |
| --- | --- | --- |
| **Skill** | **Why It’s Needed** | **Focus / Hands-On** |
| Advanced Python (OOP, decorators, context managers) | Core language for AI agents; modular & maintainable code | Build utility libraries, reusable modules |
| Async Programming (asyncio) | Handle multiple tasks or API calls concurrently | Implement async data fetchers, concurrent agent calls |
| Logging & Exception Handling | Debugging and monitoring production agents | Add structured logging to pipelines |
| Type Hints & Static Analysis | Reduce runtime errors and improve maintainability | Use mypy and pylint |
| WebSockets (FastAPI / Socket.IO) | Real-time agent communication (streaming LLM output) | Build streaming chat demo with LLM tokens |
| File I/O & Config Management | Agents need to read/write config, models, documents | Use .yaml or .env for configs |
| Git/GitHub | Version control, collaboration | Branching, PR workflow, commit discipline |

## 2. Math, Stats & Machine Learning

|  |  |  |
| --- | --- | --- |
| **Skill** | **Why It’s Needed** | **Focus / Hands-On** |
| Linear Algebra & Calculus | Understanding embeddings, gradients, attention mechanisms | Implement simple linear algebra operations in Python |
| Probability & Statistics | Essential for model evaluation, uncertainty, RAG retrieval scoring | Build notebooks for probability experiments & hypothesis tests |
| ML Fundamentals (Supervised/Unsupervised) | Core for embeddings, classifiers, similarity searches | Train small sklearn pipelines, evaluate metrics |
| Feature Engineering & Evaluation | High-quality inputs = better agent output | Build feature pipelines & test embeddings |

## 3. Deep Learning & LLMs

|  |  |  |
| --- | --- | --- |
| **Skill** | **Why It’s Needed** | **Focus / Hands-On** |
| Neural Networks (CNN, RNN, Transformers) | Underlying technology for embeddings & LLMs | Implement small Transformer models in PyTorch |
| Embeddings | Convert text/data into vector space for search | Use sentence-transformers, visualize vectors |
| Transformers & Attention | Understand how LLMs generate context-aware outputs | Inspect attention matrices, fine-tune small models |
| Prompt Engineering (Zero/Few-shot, CoT, ReAct) | Agents act and reason based on prompts | Experiment with prompts for reasoning tasks |
| RAG (Retrieval-Augmented Generation) | Ground agent responses in knowledge, reduce hallucination | Build document Q&A bot using FAISS/Pinecone |

## 4. Agentic AI Systems

|  |  |  |
| --- | --- | --- |
| **Skill** | **Why It’s Needed** | **Focus / Hands-On** |
| Multi-Agent Systems (Planner, Worker, Committee) | Complex tasks decomposed across agents | Implement planner-worker agent to solve multi-step problems |
| Memory Management (Short-term, Long-term) | Maintain context for multi-turn interactions | Store conversation in vector DB and recall relevant info |
| LangChain / LlamaIndex | Frameworks to orchestrate agents | Build chain of tools → memory → LLMs |
| Agent Orchestration & Tool Integration | Agents need to call APIs, DBs, tools | Integrate web search, calculator, or knowledge DB into agent |

## 5. Backend & Production Engineering

|  |  |  |
| --- | --- | --- |
| **Skill** | **Why It’s Needed** | **Focus / Hands-On** |
| REST APIs (FastAPI/Flask) | Agents exposed as services | Create endpoints to query agents |
| WebSockets | Real-time streaming outputs | Stream LLM tokens to frontend for chat |
| Docker & Docker Compose | Containerized agents for reproducible environments | Containerize agent pipelines |
| CI/CD (GitHub Actions/Jenkins) | Automated builds, tests, deployment | Build pipeline for ML + agents |
| Caching & Performance Optimization | Reduce LLM API cost, improve response time | Use Redis/memory caching for frequent queries |
| Logging, Monitoring & Observability | Track agent performance and failures | Integrate OpenTelemetry + Grafana dashboards |

## 6. Cloud & Deployment

* AWS Core Services: EC2, S3, IAM, Lambda, API Gateway
* Serverless: Deploy lightweight agent functions
* Bedrock (if available) for managed LLMs
* Infrastructure as Code (Terraform basics)
* Scaling & Load Balancing

## 7. Security & Reliability

* Prompt injection & guardrails
* Input validation & sanitization
* Secrets management (AWS Secrets Manager / Vault)
* Testing: unit, integration, load testing

## 8. Delivery & Portfolio

* Documentation (README, runbooks)
* Architecture diagrams (Draw.io / Mermaid)
* Full-stack agentic AI platform (RAG + Multi-agent + Frontend + Deployment)

**Tip:** Tick off each skill as you study and implement. This roadmap ensures you’re fully confident and industry-ready for building agentic AI platforms.

###### Author - Reetesh V (Ai Engineer)

[Portfolio](https://reetesh-v.netlify.app)