AI109 Midterm Review Guide

How to use this guide

- For each topic, make sure you can explain the listed ideas in plain language.
- Use the readiness checks to confirm you can apply each concept.
- Focus on conceptual clarity: definitions, comparisons, tradeoffs, and limitations.

Foundations

What you should know

- High-level differences between **symbolic** and **data-driven approaches**.
- How technical facts and social impacts should inform each other.
- Why AI literacy is now a baseline skill for everyone.
- One concrete way AI can improve productivity and one concrete way it can reduce productivity (for example, while building a website).

Readiness checks

1. You can contrast symbolic and data-driven AI with a clear strength and limitation for each.
2. You can describe one productivity gain and one productivity risk from using AI tools in coursework.

Programming Foundations

What you should know

- **Programming** as translating ideas into precise, executable steps.
- Why computers follow rules exactly, without interpreting intent.
- How simple control ideas (choice - **if statements** - and repetition - **loops**) create complex behavior.
- Why **abstraction** helps humans manage complexity in larger programs.
- How to trace small JavaScript snippets and list exactly what gets printed.

Readiness checks

1. You can explain why programming requires precision that ordinary language often does not.
2. You can explain why **debugging** is part of the core reasoning process, not just typo-fixing.
3. You can trace a simple `if/else` and a simple `for` loop and predict printed output.

AI-Assisted Programming and SDLC

What you should know

- The purpose of each **SDLC** phase: **requirements**, **design**, **implementation**, **testing**, **deployment**, **maintenance**.
- Why unclear requirements produce downstream failures.
- Why testing is not optional, even when software appears to work.
- **Graphs** as a general model of relationships and paths.
- Why **markdown** planning artifacts can improve collaboration with AI coding tools.

- Be able to name at least 3 SDLC phases and briefly state what each one does.

Readiness checks

1. You can explain how SDLC structure reduces project risk.
2. You can give an example of representing a real problem as a graph.
3. You can name three SDLC phases and describe the purpose of each.

Knowledge Representation I

What you should know

- **Data** versus **information** versus **knowledge**.
- Why **inference** is central to **knowledge representation**.
- **Syntax** versus **semantics** and why both are required.
- **Modus Ponens** as a basic **inference rule**.
- Limits of **propositional logic** for rich real-world domains.
- Ontological commitments and their impact on representational power.
- Tradeoff: fidelity to reality versus computational feasibility.

Readiness checks

1. You can explain why formal languages are useful for machine reasoning.
2. You can describe both a strength and a weakness of rigid logic-based systems.
3. You can explain how modeling choices constrain what can be inferred.
4. You can correctly classify example statements as data, information, or knowledge, with justification.

Neural Representations I

What you should know

- **Combinatorial explosion** and why symbolic search scales poorly.
- Why AI researchers explored **neuron**-inspired alternatives.
- **Threshold** behavior in simple neuron models.
- Why **perceptrons** were historically significant.
- Binary perceptron outputs are discrete class labels (typically 0 or 1).
- Tradeoffs between interpretability and scalability.

Readiness checks

1. You can explain combinatorial explosion in plain language.
2. You can describe one gain and one loss when moving from symbolic rules to learned weights.
3. You can explain why simplified neuron models are still useful.
4. You can state the perceptron computation sequence and its possible outputs.

Neural Representations II

What you should know

- **Classification** as a core AI task.

- **Binary** versus **multiclass** classification.
- Why perceptrons naturally fit binary outputs.
- Why embeddings are useful across text, images, and audio.
- Distinguish classification from non-classification tasks.

Readiness checks

1. You can explain the embedding idea at a conceptual level.
2. You can explain why feature design includes both technical and ethical choices.
3. You can distinguish classification tasks from other tasks with examples.

Decision-Making

What you should know

- **Preference relations** and why completeness/transitivity are assumed.
- **Utility functions** as numeric representations of preference.
- Why AI simplifies decision-making by modeling goals as preferences.
- What is gained, and what is lost, when human goods are represented as utility.
- Philosophical limits of treating rational choice as maximizing a score.
- Why utility maximization can conflict with fairness, rights, or dignity in real cases.

Readiness checks

1. You can explain the difference between a preference ordering and a utility function.
2. You can identify a human value that is hard to represent as a single utility score.
3. You can explain both a strength and a philosophical weakness of utility-based decision models.
4. You can give a concrete case where utility maximization may produce an unethical result.

Final Review Checklist

- Can you define each major concept in plain language?
- Can you compare approaches (symbolic vs learned, rules vs utility) with tradeoffs?
- Can you identify failure modes and limitations for each method?
- Can you connect technical concepts to ethical and social implications?
- Can you justify claims with course concepts rather than intuition alone?