The course in brief

AI Literacy: Ada, AI and Verification

v0.1.0 – 2026-11-05

The content of *AI Literacy: Ada, AI and Verification* in compact form: what each unit teaches, without its activities, checkpoints and exercises. Keys in square brackets, such as [L5], refer to the course's reference list; "as of November 2026" marks a statement about a product or service on that date.

The method

The course's method has four steps: **specify** what correct means; **compare** against evidence independent of the candidate; **diagnose** disagreement; and **state the limits** of the check.

And a rule for when to pause: pause rather than rely on the work as soon as you can no longer predict its relevant behaviour, explain how input became output at the level your claim needs, name a suitable independent check, or interpret a check that failed.

Before you begin

Unit 0: Introduction and how to use this course

Scope. "AI" here means **generative** systems, which produce text, code, images or sound in answer to a request. Systems that classify, rank, recommend or predict (a spam filter, a credit score) fail differently – often silently and across many cases at once – so they are judged by measuring errors over many cases, not by checking one output. The central question, what a claim is checked against, applies to them too. "Artificial intelligence" names a field: *intelligence*, *understanding* and *consciousness* have no agreed definitions, and the course uses none of them to say what a system is (Unit 17).

Example and arc. In 1843 Ada Lovelace published notes describing how Babbage's never-completed Analytical Engine could compute the Bernoulli numbers (Lovelace 1843, Note G; on the engine, [S1], [S4], [S12]): an ordered procedure, every quantity named, every step written down. The arc is a loop of four steps: (1) an exact ordered procedure, which follows its specification or does not; (2) generated text, fluent whether or not correct; (3) verification against suitable evidence; (4) revision, or a scoped claim. Generating output is now the easy part of most tasks; establishing that it is right is the part that still needs a person.

Declaration versus certificate. A **declaration** asks to be believed: a statement of how work was made cannot be checked from the work, and "a person reviewed it" is about process, not correctness. A **certificate** lets someone repeat a check and get the same result without trusting its maker: a value to recompute, a test to rerun, a quotation with a page number. Checking one is usually far quicker than producing the work. Disclosure is the easy half; the harder half is standing behind the work: understanding it, and handing over evidence others can check.

Not new. Printed tables had errors, quotations are misattributed, reviewers miss bugs; the 1843 Note G table contains errors [P2a], [S7, sec. 5.2]. Generative AI changes mostly **speed and volume**.

The learning contract. "Using an AI system does not remove your responsibility for the output you put your name to." Others may hold duties for design, deployment, policy or harm; the contract does not settle those. Disclosure is provenance, not evidence of correctness. Tool use is often impossible to establish from the work; correctness and answerability can be. The running question: *what would I have to check before I could stand behind this?* **Permission** (what may be sent where, what help is allowed) is a separate constraint, found out by asking.

When to pause. When one can no longer predict relevant behaviour, explain input to output *at the level the claim needs*, name a suitable independent check, or interpret a failed check. Opaque tools (a compiler) can

be checkable from outside. Confidence is not evidence: narrow the task, rebuild the foundation, or seek qualified review. Calculations are run against known values; claims about the world are looked up; a low-risk draft may need less checking, but claims, personal data, confidential material and consequential decisions still need appropriate handling.

What these systems are

Unit 1: What generative AI is for

The question that does not date. Not "what is it good at?" but *where would the evidence come from that the output is right?* A **reference** is whatever the output is compared against. The class it puts a task in identifies the checking job and sets a typical cost, not a fixed price.

Five classes.

- **Transformation** (content supplied, wanted in another form: translated, summarised, reformatted). Reference: the input; checking is low-cost if you know the input. Characteristic failure: **omission**, easy to miss because output reads as complete.
- **Grounded answering** (a source plus a question; some systems fetch it [L3, sec. 2]). Reference: the source, read at the cited point. A source changes *where* the check is aimed, not whether one is needed: does it say what the answer says, and is it the right source? As of November 2026, one provider's documentation says its assistant may search the web unasked [D4, opening note]; each link is a claim that a source exists and supports the answer.
- **Generation to a specification** (what counts as correct is stated). Reference: the specification; moderate cost. Failure: meeting the specification written, not the one meant.
- **Recall** (a question only). Reference: external. Generation consults no record of truth, so recalled claims carry no evidence and no marker of which are right; checking is a separate research task, and the gap between producing and checking is widest. References to non-existent works have been found in published peer-reviewed papers [L10, secs. 2.1, 4.1], and one audit of early 2026 found fabricated references in about one biomedical paper in 277 [L11]. Checking that a cited work exists is the cheapest check.
- **Judgement** (no single determinate answer). Nothing to compare against, but premises are checkable and the output can be assessed against stated criteria. The output is a **proposal**; treating it as a finding is the error. A confident register is a property of the text, not evidence about the question.

The typical cost ranking is of work left afterwards, not usefulness: recall and judgement are usually the expensive classes, which makes them the ones owed the most checking, not the ones to avoid.

Moving class. Supplying the source turns recall into grounded answering ("here is the regulation – what does clause 4 require?"). Writing down what counts as correct turns a vague request into generation to a specification. In any class, ask for checkable evidence ("quote the sentence you rely on"; "give code and tests I can run") – a certificate, itself checked against the source since it can be fabricated.

Worked example. "Does our coursework policy allow AI to draft an essay plan?" Nothing attached: recall. Policy pasted in: grounded answering (an answer stretching a brainstorming clause into drafting is found in a minute). A table of every AI clause with number and wording: transformation (look for a dropped clause). "Is it a good policy?": judgement. Likewise "what is B8?" is recall; "here is the recurrence – compute it" is generation to a specification.

Limits. Ask: if this is wrong, how would I find out, and at what cost? The answer is bad when no reference is obtainable at a payable price (the future, private facts, the edge of knowledge); when one would not recognise a wrong answer (outside one's competence every surface cue is available to a wrong answer); and when the doing is the point. Cheap to check is not checked.

Unit 2: Ada's ordered procedure

The document. In 1843 an English journal published Lovelace's translation of Luigi Menabrea's French-language account of the Analytical Engine, written after Babbage presented it at meetings in Turin in 1840 [P1], [P3, pp. 129-130]. Signed with her initials, it carried seven notes, A to G, longer than the memoir. Note G sets out how the engine could compute the Bernoulli numbers.

Note G is a prose note containing a table: one line per operation, first to last, each giving the operation, the storage columns supplying inputs, and the column receiving the result (Lovelace 1843, Note G). Babbage's

design separated a **store** (numbers in columns) from a **mill** (arithmetic on numbers brought from the store) [S1], [P3, pp. 117-118]. The engine was never completed, so the procedure had to be exact without being run.

Three properties.

- **Determinism.** Same starting values and operations, same result, every time. A disagreement between runs is therefore a fault that can be found, not variation.
- **Explicit state.** The **state** is the values held in the storage columns, each a stated starting value or put there by an earlier line; nothing is merely understood to be available. This fails first in hand-written instructions.
- **Control flow.** Order is part of the meaning; swapping two lines gives a different procedure. Note G's diagram marks operations to be repeated, the count depending on which Bernoulli number is wanted (Lovelace 1843, Note G). The distinction is between **a worked trace of one case** and **a general procedure with a stopping rule**, which computes a family of answers; a recipe ("simmer until it thickens") is usually the latter.

"The first program". Firm: publication in 1843 with the translation, signed with her initials; Note G contains an ordered table of operations for Bernoulli numbers [P2, pp. 722-731]. These do not by themselves establish priority. Contested: Babbage wrote operation sequences in unpublished notebooks before 1843, so "first" depends on whether publication is the criterion [S1], [S4]; the division of work between Lovelace and Babbage is debated [S2], [S3], [S5], [S10, "Changing perceptions of Lovelace"], [S11, sec. 1]; "program" was not used in this sense in 1843, and whether the operation sequences amount to the concept depends on how the concept is defined; applying the word is a defensible choice, not a fact about 1843.

Printed errors. One – operation 4, an inverted division – is established from a facsimile [P2a]; a self-published reconstruction [S7, sec. 5.2] lists further faults in operations 24 and 21. Who introduced them is contested, and what the faulty table computes depends on which errors are repaired and what it is fed.

The originating sentence. "The Analytical Engine has no pretensions whatever to originate any thing" (Lovelace 1843, Note G): in context, a claim about following orders. It assumes one can say precisely what the machine was ordered to do and check whether it did. Note G is fully checkable: any written line can be found wrong, and one was (when is not known). A procedure written out in full is evidence anyone can check without trusting its author; human authorship never guaranteed correctness.

Background only. Lovelace was Byron's daughter; he left England in 1816 and never saw her again [S5], [G1, pp. 605-610, 1227]. The ghost-story challenge that produced *Frankenstein* (1818) and Polidori's *The Vampyre* (1819) is conventionally dated to that summer's Villa Diodati gathering [G2, pp. 125-126] [G3] [G4]. The link is a visual motif, not a historical inference.

Unit 3: What a language model does differently

Unlike an ordered procedure's run, a generated answer is not an execution trace proving each claim.

Mechanism. A large language model built on the **Transformer** architecture, introduced in 2017 [L1, sec. 3]; the argument rests on next-token generation over a context, not the architecture. Text is split into **tokens** (words, word parts, punctuation). Over many layers, **attention** gives each position a weight for every token up to that point, and combines information from those tokens in proportion; the final layer scores each possible next token, scores become probabilities, one token is selected and appended, and the step repeats. Attention is not a lookup, a citation or a truth test.

Layers. Pre-training tunes parameters for next-token prediction on a large corpus. Assistants may be **post-trained** on demonstrations and preference judgements [L2, secs. 3.1, 3.5] and add retrieval [L3, sec. 2], tools, memory, or intermediate reasoning text. None makes a claim correct by definition; visible reasoning is generated text about the work, not a transcript of it. Computational effort is not formal reasoning.

Training cut-off. Nothing written after its training text ends shaped the base model, and without tools may present an outdated state as current. Tools change this only for what they fetch; as of November 2026 one provider's documentation says its assistant may search unasked [D4, opening note]. Ask when training text ends and whether anything was fetched this time: a link in an answer is generated text, a product's search record is evidence, otherwise record "unknown".

Consequences.

- **Sampling. Decoding** picks the next token: always the top one, or one drawn by probability. Sampling, or a changed service, lets the same prompt give different text. Repetition shows stability or change,

not correctness; re-asking is not an independent check. **Sycophancy**: a study published in 2024, of assistants from 2023, found that those trained with human feedback shift towards the user's apparent belief, and a bare "are you sure?" can make them abandon a correct answer [L5, secs. 3.2-3.3]. Ask neutrally, or for evidence against.
- **Correctness is not computed at generation.** Without tools no external record is consulted; retrieval moves where checking is aimed, not whether it is needed.
- **Fluency does not establish correctness.** True and false text can be equally fluent, specific and confident; polish, once a weak signal of care, now comes free. "I checked" is generated text, not a record of a check. **Hallucination**, fluent false output, is the same process as correct output when the plausible and true continuations differ.

Output is not random, and the point is not avoidance: inspection alone cannot tell which case one is in, so confidence must be proportionate to an appropriate check.

The triage. Ask: *how would this be wrong, and what check catches that?*

- **Exact checking** – calculations, conversions, code, data transformations: run against known values or a test suite. Mismatched units mean wrong at once.
- **Source checking** – dates, citations, attributions, requirements, API specifications, summaries of unread documents: look up the official record. Re-running or "are you sure" is not **independent verification**, nor is a system's account of its own checks; tests count only if independent of what they test.
- **Low-risk drafting** – error cheap and visible, text to be reworked.

A judgement fits none of the three: check its premises, and keep its conclusion a proposal.

Sort by how an error would reach you, not by importance; one output can span categories (code and a historical comment fail independently). Re-prompting and self-critique can improve output [T9, sec. 3] but can preserve a shared error [T9, sec. 3.3].

Unit 4: Choosing the tool

Each layer added to a base model moves the check and adds one of its own; choosing a system is choosing a checking job.

Four kinds.

- **Plain generation.** The whole check stays with the user; recall stays recall. Right for transformation, usually fastest and cheapest; not a way of finding things out. Failure: a claim with no evidence and no marker.
- **Retrieval-backed** (generates conditioned on fetched sources [L3, sec. 2]). The most useful single upgrade for recall, moving it to grounded answering. Two checks: does the source say what the answer says (a real citation can be attached to a claim it does not make), and is it the right source? As of November 2026 one provider's documentation says its assistant may search unasked [D4, opening note]; record whether it did. **Research modes** plan, run many searches and return a long linked report (as one provider described in December 2024 [D5, "Deep Research: Your personal AI research assistant is here to help"]): retrieval in a long loop. Forty citations owe forty checks, and none shows what was searched or left out; compare with a source chosen independently and state what remains uncertain.
- **Tool-calling** (runs deterministic code). Within its own terms the computation is right or wrong in the ordinary way – a real gain; the choice to call it, with those inputs, for this question, is generated. Failure: a correct computation of the wrong question (right formula, wrong column), or wrong generated code. Shown code and output are rerunnable evidence; "it was computed" is a claim. For "all tests pass", ask whose tests.
- **Multimodal.** The reference is no longer directly searchable: search may need a transcript or an image tool, and what it finds is checked against the original. A description can confidently name what is not in the frame; charts are read from pixels (gridlines, truncated axes, log scales); a picture or recording may be generated or altered – treat it as a claim about where it came from.

Systems that act differ again: a wrong answer can be discarded, a wrong action cannot.

Decision aid. (1) Name the class. (2) If recall, try to supply the source – this removes more checking than any product choice. (3) Ask what the answer will contain: sources, a computed number, an image reading. (4) Ask whether *you*, with your time and access, can do that check. (5) Take the cheapest kind that leaves a doable check. Step 4 does the work. A benchmark score does not answer it: it means little if test questions

were in the training data [E17, p. 10776], [E18, sec. 2.2], and a clean score concerns the benchmark's tasks, not yours.

Money, time, energy often share drivers: text in and out, model size, number of steps. A longer conversation can cost more because history may be sent; price lists change. No per-query energy or carbon figure is given: a defensible one must name model, hardware, query length, whether the cost of training is shared across queries, and electricity supply, and dates within months. The stable shape: other things equal, more computation uses more resources; money and time also depend on pricing, caching and hardware. An actual figure is a recall task needing a dated source stating its conditions.

Cheap is often right: for transformation or grounded answering when the comparison will be done anyway, a draft to rewrite, or when extra output would go unchecked. The expensive option earns its cost when its layer removes a check one could not otherwise do, and that check is used. Retrieval and tool-calling are differently checkable, not more trustworthy.

Unit 5: Prompts, context and attachments

Two "same" questions can get different answers because a prompt is not what reaches the system.

What goes in. For a text-only model, generation is conditioned on one sequence assembled just before generation; multimodal systems may also receive encoded images, audio or video. Besides the prompt it can hold: unseen standing product instructions; earlier turns, possibly truncated or summarised; attachment contents; retrieved passages; tool results; saved memory and instructions. The prompt can be a minority of the context and the same words in another conversation are a different input.

The window has an edge. On overflow, earlier turns may be dropped or summarised. A long document may be chunked and only some pieces retrieved; one that fits may be used unevenly – on the tasks one study tested (published 2024, models from 2023), models used material in the middle of a long input worst [L8, p. 157]. None of this is usually shown. Characteristic failure: an answer as though the document said less than it does, confident because its context did say less. A **locator** (page, section, clause) lets one check the cited passage exists and supports the answer, not that it reached the model.

Attachments are not references. Attaching moves the task into grounded answering but does not make the answer an extract: it can paraphrase beyond the document or attach a real page number to a claim from elsewhere. Discipline: ask for the passage; read it in the document; ask whether it settles the question (the step skipped). **Prompt injection**: text in an attachment or retrieved page, visible or hidden, can act as instructions. Indirect injection was shown against a real deployed system in 2023 [L6, abstract]; 2025 industry guidance counts web pages and files as routes [L7, "Indirect Prompt Injections"] and lists no fool-proof prevention [L7, "Prevention and Mitigation Strategies"]. Treat it as data.

What changes the answer. Stating what counts as correct; an example of the form wanted (constrains shape, not truth); a new conversation (clears accumulated context). None makes output self-verifying.

Worked example. For a twenty-page handbook, "What does this say about late submission?" may be right, but one cannot tell without reading the handbook. Better: "Using only the attached handbook, list every rule it states about late submission. For each one give the section number and quote the sentence it comes from. If the handbook does not state a rule about something, say so rather than filling the gap." It states what counts as correct, asks for locators, and separates silence from a statement that no rule exists. The answer is *checkable*, not verified, and checking the returned rules does not show completeness: that needs a search for omitted rules, exceptions and precedence clauses. A quoted sentence with its section number is a certificate.

Habits.

- **Record the whole input**, not just the prompt. As of November 2026 one provider's documentation says a shared conversation link is readable by anyone who has it, and copies outlive revoking it [D3, "Frequently asked questions"].
- **A fresh conversation is the reproducible case**; repeating in the same conversation is least informative. Fresh is clean only if no memory, account-wide instructions or project instructions carry over; one provider's documentation (as of November 2026) describes all three, with memory able to be on by default [D1, "Instructions for Claude", "Project instructions"], [D2, "What is Claude's memory?"]. Where the right answer should not depend on wording, an answer that changes with rewording is wrong at least once – a metamorphic relation [E20]; surviving it shows consistency, not correctness.
- **Ask for locators**: "Clause 4.2, second sentence", not "the policy permits this".

Both hold: prompting matters, and wording cannot recover what is not in the context; supply the part that matters rather than hope the whole arrived.

The worked case

Unit 6: Meet the Bernoulli numbers

Judging generated output needs an independent answer to "what should the answer be", so the reference comes first: build enough understanding to judge, then use the tool.

Definition. The Bernoulli numbers are a fixed sequence of exact rational numbers, one per whole-number index from 0, written B0, B1, B2, … Given the index and the convention, the value is completely determined: there is a right answer and disagreement can always be resolved.

Values (first convention): B0 = 1, B1 = -1/2, B2 = 1/6, B3 = 0, B6 = 1/42, B10 = 5/66, B12 = -691/2730, B14 = 7/6, B16 = -3617/510, B18 = 43867/798, B20 = -174611/330, B30 = 8615841276005/14322; B4 is derived by hand. All odd B_n for n > 1 are exactly zero. Values do not shrink tidily, and decimals can look right at every index and be exact at almost none, so only an exact comparison catches them.

Origin. Closed formulas for sums of powers share one general form whose coefficients are this sequence; it was established before Lovelace's Note G, which calls them "the Numbers of Bernoulli" [P2a]. The reference form uses Bernoulli polynomials [M6, sec. 24.4(iii)]; with B1 = -1/2 it sums to one less than the limit:

```
0^2 + 1^2 + ... + (m-1)^2 = (m^3 + 3*B1*m^2 + 3*B2*m) / 3
```

For m = 9 this gives 204 = 0^2 + … + 8^2, not 285 = 1^2 + … + 9^2; summing from 1 needs B1 = +1/2. Checking that a procedure was carried out correctly is not checking that it was the right procedure.

Recurrence. From B0 = 1, for every n >= 1:

```
sum_{j=0}^{n} C(n+1, j) * B_j = 0
```

C(n+1, j) comes from row n+1 of Pascal's triangle; j stops at n. Every term is known except the last, which contains B_n: substitute and solve. n = 1: B0 + 2*B1 = 0, so B1 = -1/2.* n = 2: 1 + 3*(-1/2) + 3*B2 = 0, so 3*B2 = 1/2 and B2 = 1/6 – no calculator or anyone's assurance needed. n = 4 has five terms and yields B4. It is an ordered procedure with no judgement at any step. Read the other way it is a check, and the cheaper direction: substitute claimed values and see whether each row sums to exactly zero.

Trap 1: two conventions. First Bernoulli numbers have B1 = -1/2, second have B1 = +1/2, and every other value agrees. Both are standard; which is wanted depends on the formula being fed, and neither is more correct in the abstract. This course fixes the first. A **convention** is a choice fixed by agreement rather than derived: two people following different conventions can both be right and still disagree. When competent sources disagree, check first whether they answered the same question; a specification that omits a convention has not made it go away, only made the disagreement harder to diagnose. Code: `b1-sign`.

Trap 2: Ada's indexing. Lovelace leaves the zero odd terms unlabelled and numbers the non-zero values from the modern B2 with the odd integers, so her Bk is the modern B(k+1) (Lovelace 1843, Note G): her B1, B3, B5, B7 are the modern B2, B4, B6, B8. Note G's worked example computes Ada's B7, the modern B8 = -1/30. This is a different notation with identical symbols, not an error. A routine can be a faithful reading of Note G and a broken modern component at once: historical fidelity and specification compliance differ. Code: `ada-indexing`.

Exactness. Comparing a claim against a known-good table to find where they first part is a professional technique in its own right. In code, `fractions.Fraction` (exact rational arithmetic, nothing rounded) is required: a float gives 0.16666666666666666 for B2, which does not compare equal to 1/6. An **oracle** is a trusted reference an answer is compared against.

Unit 7: Ask AI to generate the routine

The prompt is "Write a Python function that returns Bernoulli numbers." It is what people type, and it can work well enough. It leaves open the convention, the return type (exact or float), the range, the odd indices, and whether exactness matters. A language model produces a statistically plausible continuation of its input; where the text is silent the code must still contain something, and that reflects patterns in similar training code, not unstated requirements. So an unstated requirement usually appears not as a gap but as a confident default. A system may instead ask, or state its assumption, which is better because a stated assumption can

be checked, but is still a choice made for you. None of this is peculiar to AI – every one of these failures occurs in code people write; what is new is that a generator settles open decisions instantly, silently and at volume.

Five failure modes: - `b1-sign` – `B1 = +1/2`, the second convention's value. If it follows from the missing convention clause, it is not broken, only wrong for this specification; a sign error at B1 prints the same values, so the specification or the code settles which. - `odd-terms` – odd terms above B1 not exactly zero. B3 is zero, not approximately zero. - `ada-indexing` – only non-zero terms numbered, as in Note G; invited by prompts mentioning Lovelace or Note G. - `off-by-one` – every value shifted one index; each value genuine and exact, under the wrong index. A `n + 1` in plain sight can look like nothing, because the error exists only relative to a specification. - `not-fraction` – not a `fractions.Fraction`. In Python, dividing one integer by another with `/` already gives a float. The type is wrong from B0 (1.0 is exact, so the fault there is type, not value); inaccuracy arrives separately, as binary floating point cannot represent most later values. A test over B0 to B12 accepting anything within 1e-9 would pass it; a tolerance is a decision, not a default.

None makes code look suspicious. Reading with the specification can spot some, but cannot establish compliance: testing locates faults, reading says what to test for. A test that finds an error is the successful one [E10]; property-based testing checks a property that must hold for every input on many chosen inputs [E11]. `raises`, `import-error` and `no-function` announce themselves.

The sixth failure mode, which no test catches. A comment such as "Ada Lovelace used this same recurrence in Note G (1843)" comes from the same process as the code and fails independently. Arithmetic can be tested; claims about the world can only be looked up; both can be equally polished. The point is not that the comment is false but that you do not know, and correct adjacent code is no evidence either way. Generated tests reported as passing are likewise the tool's own claims ("all tests pass" is one more claim), and a displayed step-by-step explanation, one 2023 study showed, need not reflect what drove the answer [L9, sec. 3.2].

Worked example. "Goedel's theorem proves that an AI cannot verify its own output" applies a theorem about one kind of system to "an AI" without showing that its conditions or conclusion transfer: the theorem needs a source check, the transfer a separate argument.

Output as evidence. Keep it exactly as generated, with the prompt verbatim, the date and the tools that were on; some may be on by default – one provider's documentation, as of November 2026, says its assistant may search the web unasked and memory can be on by default [D2, "What is Claude's memory?"], [D4, opening note]. A tidied version is a summary of the evidence. Generated code is from a source you cannot vouch for: read it before running it, in a disposable environment.

Unit 8: Verification lab

Five ideas. A **test oracle** gives correct answers trusted independently of the thing tested; a second opinion from the same source is not one. **Differential testing** runs two implementations of one specification on the same inputs and compares. **Property-based checking** asserts a statement over a class of inputs (every odd B_n above B1 is exactly zero). The **first divergence** is the lowest input where two implementations disagree: a start for debugging, not the cause of later disagreements. A **certificate** lets someone check a claim without trusting its producer – results against an independent reference, values substitutable into their identity, a quotation with its locator – and checking it is usually far cheaper than producing. A checker report (command, oracle and its confirmation, PASS over 0 to 30 or the first divergence) is one; "I checked it" is not.

Why the oracle is trusted. Exact `Fraction` arithmetic, so a disagreement is never rounding. It checks its recurrence against 22 published values, including zero odd entries, and computes B0 to B31 through the recurrence and confirms each odd term from B3 to B31 is zero; the source is the NIST Digital Library of Mathematical Functions, section 24.2, Tables 24.2.1 and 24.2.3, checked on 2026-07-24 [M2]. It is checked at B30 (about 602 million: 601,580,873.9) because small indices hide bugs. It states its convention, B1 = -1/2. A second implementation by a different algorithm (Akiyama-Tanigawa) agrees – stronger evidence than self-agreement, though both belong to this course and still need human review.

What a PASS establishes: agreement with the oracle over 0 to 30, in exact fractions, and the zero-odd-terms property. That is evidence, not an explanation; what you hand on is the certificate, not a declaration. The table route, on paper, is a full route too: each route misses things (a table in a chat window may be generated text, not program output), neither is complete, and both are real verification.

Running generated code. The checker executes the file in a separate process with limits: a blast-radius reduction, not a security boundary. Generated code is untrusted – a statement about what you know of it – so read it and its imports before running it. The same holds for code people wrote; closed-source code cannot be read at all, which is one reason open source helps checking.

Property versus table. A table covers the rows written down; a property covers indices never thought about. The float example gives B3 = -1.1102230246251565e-16, which an absolute tolerance (`abs(x) < 1e-9`, `numpy.isclose`, `pytest.approx`) accepts as zero; `math.isclose` with its relative default does not. Tolerances are where inexact arithmetic hides.

Diagnoses. The `[divergence]` line says where; the code says what kind. `one-observation` means `ada-indexing` and `off-by-one` both matched; they agree everywhere except B0. Examples: wrong B1 sign diverges at n=1 (1/2), a specification failure, not an arithmetic one. Ada indexing diverges at n=1 (1/6) and matches both readings; B0 = 1 is all that separates them, and settling it means reading the source – a comparison says where to look, not always what you are looking at. A wrong B4 sign diverges at n=4 (expected -1/30, got 1/30) with no named code: the codes name recurring faults, not every fault.

Checks needing no table. Even values from B2 alternate in sign [M5, eq. 24.2.2]; for even n >= 2 the denominator is the product of primes p with p - 1 dividing n (von Staudt-Clausen [M4, eq. 24.10.1]; for B12, 2*3*5*7*13 = 2730); for large even n, |B_n| is asymptotic to 2 * n! / (2*pi)^n, their ratio tending to 1 [M5, eq. 24.11.1]. Each is necessary, not sufficient. The checker cannot say whether its convention is the one the task needs.

The recurrence as certificate. Substituting all values into sum_{j=0}^{n} C(n+1, j) * B_j = 0 needs no oracle; a list from B0 = 1 passing every row to n = 30 is the sequence to B30. This is **metamorphic testing** [E13, sec. 3.3], [E20, sec. 2.2]. Also B_n^+ = (-1)^n B_n^-. Most relations are necessary, not sufficient; the recurrence is the rare complete one.

External-reference principle. A check has to come from a reference the thing checked does not control; asking the same source again is not a check. A tool's self-written passing tests encode its own reading of the task: a generated test asserting B1 = +1/2 passes the `b1-sign` routine.

Evaluation (no oracle): state a question that could disappoint; choose the measure before seeing output; pilot small and discard its data; freeze and date the design; run without adjusting; report with uncertainty. Failure is expected at step 2: a measure that is accurate but does not measure the brief.

Extending the method

Unit 9: Grounding and citation checking

The same method on a case with no oracle: the artefact is a referenced summary, the reference is what the cited sources actually say. Compare claim against source one at a time, stop at the first unsupported claim, name it, and end with "supported at this strength, by these sources".

A generated citation establishes nothing by itself: it is generated text like the claim. Retrieval (one provider's documentation, as of November 2026, says its assistant may search unasked [D4, opening note]) plausibly helps the first two questions below and leaves the third untouched. But a citation is the cheapest part to falsify: a locator makes the check bounded. The full certificate is the quotation with its locator; a checked quotation certifies that the source says this, not that it is right. Trust in a source is checked by **lateral reading**: fact checkers left an unfamiliar site early to read what others said about who stood behind it, rather than staying within the site to judge it (vertical reading) [E19, abstract].

Images and recordings. In a 2026 study, reviewers with PhD-level training identified research images altered by a generative model no better than chance; the best automated detector tested did better but was far from reliable [V8, "Results"]; an earlier study found free AI-image detectors unreliable on laboratory images [V7, "Discussion"]. A missing provenance label shows nothing about how a file was made; when an image or recording asks you to do something that matters, confirm through a channel you found yourself. Trace an image like a citation: earliest publication, independent records; "untraced" is not "generated".

Three questions, in order. Does the work exist (cheapest, settles least)? Does the locator exist? Does the source support the claim as stated, at this strength (the whole exercise)? Question 1 cannot be assumed: a scan of 4,841 of the 5,290 papers accepted at one 2025 machine-learning conference found 100 fabricated citations in 53 of them, two thirds to works with no real counterpart [L10, secs. 2.1-2.2, 3.1]; an audit of about 2.5 million biomedical papers found at least one fabricated reference in about 1 in 2,828 papers from 2023, 1

in 458 from 2025 and 1 in 277 from the first weeks of 2026 [L11]. These measure what reached print, not how often systems invent references, and both studies used automated tools to find candidates.

Six modes, closed for this exercise but not exhaustive of reality: - `fabricated-locator` – the work exists, the locator does not. - `stale-version` – accurate, superseded, and the claim is about the current state. - `misattributed` – finding absent from the cited work, supported by another identified source. - `over-specified` – same proposition, population and scope, at unsupported precision ("approximately 20 percentage points" -> "19.6"). - `unsupported-inference` – the source supports a weaker proposition; the summary adds causation, generality, certainty or scope. - `right-source-wrong-claim` – the right passage reports a different value, population, direction or result.

Decision order (binding): locator missing; explicitly superseded statement presented as current; finding absent from the cited work but supported by another identified source; same proposition, less precise; weaker proposition with added force; right passage, different result. Take the first that applies – the **primary** fault, since modes sit at layers (pointer, edition, work, meaning) and one claim can fail at several. The order runs cheapest to costliest. Misattribution is repaired by correcting the citation; right-source-wrong-claim is not, because the claim itself is wrong.

Over-specification sharpens a **quantity**; unsupported inference changes **logical force**. `misattributed` versus `right-source-wrong-claim` turns on whether the cited work contains the finding, never on "relevance". Support is a judgement about meaning, so no program in this course detects these, and tools that assist need checking; the closed list is the discipline.

Reference first. Read the sources before the summary; one row per locator: source and locator, what the passage supports, limitation/scope/version note (where supersession and stated limits are caught). "Could not retrieve" is a finding. Stop diagnosis at the first divergence but check the rest before relying on it. Deliver a scoped statement, not "the summary is accurate". Once a summary is known to be capable of an unsupported claim, later claims looking fine says little. None of the six modes is new – all occur in human work; generation changes volume. No claim is made that generated citations are usually wrong (the course has no measurement of that), and retrieval moves the problem rather than removing it.

Unit 10: Re-prompting, specification, and overreliance

Better prompting is precise requirements work, the same skill as specification, not a search for words that unlock a correct answer.

The vague prompt leaves open convention, indexing, return type, domain and zero terms; unsettled, the output settles them by patterns in what the system was trained and tuned on. Each checker code points at a requirement: `b1-sign` convention; `ada-indexing` modern indexing; `not-fraction` exact Fraction; `raises` defined for every n >= 0; `odd-terms` exact zeros; `off-by-one` anchoring values; `no-function` name and signature; `import-error` clean import, standard library only. None is a wording problem.

Precise prompt. C1 one module defining `bernoulli(n: int) -> fractions.Fraction`. C2 first convention, B1 = -1/2. C3 modern indexing, not Note G's. C4 defined for n >= 0; ValueError for n < 0. C5 Fraction(0) for odd n > 1. C6 exact Fraction only, no floating point anywhere. C7 anchors B0 = 1, B1 = -1/2, B2 = 1/6, B3 = 0, B4 = -1/30, B12 = -691/2730, B30 = 8615841276005/14322. C8 import without side effects, standard library only. C9 no historical claims about Lovelace or Note G in comments unless asked. A specification precise enough to test against is precise enough to prompt with.

The checker tries 0 to 30 and checks the interface, convention, exactness and zero terms; each checker code marks one clause. Reading establishes what code appears to do, not what it does, and a dynamic check cannot show the algorithm used, its scaling, a hidden lookup table or behaviour outside the tested range. No program in this lab reads C9, the inside half of C6, the side-effect half of C8 or the n < 0 half of C4. C7 is also a trap: a lookup table can meet named values and fail elsewhere, so examples are not a specification. No better checker closes this in general: by Rice's theorem [T11, pp. 364-365], [T13, sec. 3.2] no program decides, for every program, whether its behaviour over all inputs has a given non-trivial property; the halting problem, a close variant of the problem Turing proved undecidable in 1936-7 [T10, sec. 8], is undecidable too [T12, sec. 2.4.3]. A particular program can sometimes be proved correct for all inputs, but the proof is about that program, which is why a certificate speaks for one artefact, not a method.

The lookup table is one case of a requirement that something useless also meets; one 2026 preprint notes that its security definition alone would count a protocol that always aborts as secure, which is why it is paired, following earlier work, with a bound on how often an honest run aborts [L18, abstract, sec. III.B]. Ask what

the laziest output that passes a clause would be. Where possible, make the wrong output impossible to write – a type, a schema, a restricted language, as that preprint's authors built after an AI-generated protocol computed a key from both parties' strings combined, which neither held on its own [L18, sec. II]. Say what the construction does not enforce, and validate the representation like the rest of the specification.

The checker **verifies** against the contract; whether B1 = -1/2 was right is **validation**, settled by using the numbers (the sums-of-powers example). Most clauses answer a failure the lab's constructed examples show – C2, C3, C5 (a shifted routine returns -1/30 at n = 3), C6 (float B3 = -1.11e-16); C4 was foreseen.

Overreliance is relying on output from any source or tool past the point where you can check it; the gap is your own understanding, so it is situational. Deriving B4 = -1/30 by hand makes 1/30 recognisably wrong; a reference independent of the output is what does that, and an independently derived value is strong evidence because the generated answer did not determine it.

A correct answer is not evidence of learning. In one field experiment, school students practising maths with a standard chat assistant did far better on practice, then scored 17% lower on the exam without it than those who never had it; a hint-giving version largely avoided the drop [E21, "Experimental Design", "Main Results"] – one setting, and research is early. Attempt first, then check with the system.

When to stop: pause if you cannot predict relevant behaviour, explain the transformation at the level the claim needs, name an independent check, or interpret a failed one. Subjective confidence is not evidence.

Without an oracle: write the specification anyway; ask for checkable evidence; check against independent sources; compare an independent version – "are you sure?" is not one, as in a study published in 2024, of assistants from 2023, those trained on human feedback shifted towards the user's apparent belief and could abandon correct answers [L5, secs. 3.2-3.3]; check metamorphic relations [E20, sec. 2.2]; state what was not verified. Responsibility and judgement do not transfer. "Correct" is not a property a value carries alone: B1 = -1/2 and +1/2 are both correct and only appear to disagree, because the convention each assumes is left out. The Bernoulli case was chosen because it is clean, not because it is typical: most questions have no oracle.

Unit 11: Agents and automated actions

An **agent** acts: it calls tools in a loop, and some actions change things outside the conversation – a file written, a message sent, a row deleted. One thing changes: an action can change the world before you have looked at it. Checking afterwards is still needed, but it describes an error rather than preventing it, leaving a remedy rather than a correction.

The loop: plan (generated); tool and argument selection (generated); execution against the world's state at that moment; a recorded result; the result read back and the next step generated; repeat; a report (generated text about the run). Execution is not the safe step except for tools with no outside world: searches reflect today's index, file operations depend on disk, permissions and concurrent writers, network calls time out or half-finish, tools have bugs, and a tool can do exactly what it was told while doing the wrong thing.

Three failure points: **selection** (wrong tool or arguments; read or approve them first), **execution** (timeout, partial write, stale data; look at the result, not the exit status), **postcondition** (success reported, world not as intended; read back state). `ok` is a claim about a call, not the world.

Prompt injection. Text read back can act as instructions. The indirect kind, inside retrieved content, was shown against a real deployed system in 2023 [L6, abstract]; 2025 industry guidance lists web pages and files as routes [L7, "Indirect Prompt Injections"] and says no fool-proof prevention is known [L7, "Prevention and Mitigation Strategies"]. The defence is limiting what a steered run could do: least access, and approval before steps in the bottom two bands below.

Reversibility is the axis: if this is wrong, can it be undone, by whom, at what cost? - Free to undo (a draft on your machine): check afterwards. - Undoable with effort (commit, scheduled job, shared document): check afterwards, budget for the undo. - Undoable only by someone else (message, form, payment): confirm before. - Not undoable (personal data disclosed, deletion without backup, a secret seen): confirm before, every time.

Remedies: **undo** (no trace that matters); **compensating action** (a refund, a correction; the usual third-band remedy); **recovery** (from a backup or history arranged beforehand); **containment** (revoking a key, notifying those affected). Reliability is deliberately absent: it changes how often a remedy is needed, not its cost; repeating bottom-row actions is choosing an exposure, not calculating a date.

Traces. Calls and results are records only if read from the system's own execution log; a chat rendering may condense, reorder or omit; the system's account is generated text in the same voice. A summary can

misdescribe a run – not by deception, which would need intent, but because it comes from the same process and nothing checks it against the log. Check the tool calls, not the narration; a step's absence from a displayed trace, or presence in a summary, is not evidence either way.

Worked trace ("remove duplicate entries in my references file"): read 412 lines; "6 duplicates" claimed; whole-file write of 380 lines; `ok`; "Removed 6 duplicate entries." Check the file and path; which six (matching by title alone merges distinct works sharing a title); the overwrite, which shows nothing about which 32 lines went; and the report's unstated claim that nothing else changed, which a rewrite can break. The right intervention is a copy before the write.

Practice. Move actions up the table first (copy, branch, send to yourself); bound the run; ask for the plan before actions (approving a plan is not verification and will not catch an error in execution); give least access, which removes possibilities rather than detecting results; use dry runs; arrange evidence beforehand (diffs, logged outputs, tests that existed before the run). The habits are not new; agents add speed, volume and loops nobody reads line by line. Much agent work sits in the top two bands, where running unattended is right; the record is evidence, the narration is not.

Duties and consequences

Unit 12: High-stakes use and escalation

Two axes, asked in this order and kept separate:

- **Kind** of check comes from where the reference lives.
- **Amount** of check comes from what being wrong would cost.

Sorting by how important a task feels is the commonest way to get the kind of check wrong, so importance returns on its own axis, where it cannot corrupt the diagnosis.

What "how much" is made of. Four things that interact rather than add up: a bad enough answer on any one can carry the whole decision. Nothing is scored or multiplied.

- **Consequence** – if wrong and unnoticed, what happens, and to whom. Low-stakes to you can be high-stakes to somebody else.
- **Reversibility** – can you find and fix the error later, cheaply? A number in a report sent to a client is not recoverable in practice.
- **Your own uncertainty** – could you recognise a wrong answer? Outside your competence every surface cue is available to a wrong answer too.
- **Whether it is yours to decide** – competence and authority differ; when the decision is someone else's, escalation is the correct route, not a confession.

Decision table (for a claim you will rely on):

- Small, recoverable, assessable: spot-check once.
- Small, recoverable, not assessable: use as a draft, do not present it as established, keep the way back open, say what you could not check.
- Small, not recoverable: make it recoverable first, or treat it as large.
- Large, recoverable, assessable: full check of the kind the class implies; record it.
- Large, not recoverable, assessable: full check before the irreversible step, plus a second pair of eyes if available.
- Large, not assessable: **escalate**.
- Not yours to decide: **escalate**, whatever you could work out.

The last two rows are the point, and people talk themselves out of the first. An unchecked result labelled as a draft is a different act from one presented as checked. "Can you assess it?" is not "could you have produced it?": you may still check a certificate, such as the passage of official guidance with its location. Experts err too, which is why high-stakes fields built sign-offs and independent review before AI; AI changes how fast a confident answer arrives.

Escalating well means handing over: the claim, stated precisely; what you checked and what it showed; what you could not check and why; what decision is waiting. Not "the AI said this, is it right?". When you cannot check even the certificate, escalation means finding someone who can.

Triggers, any one sufficient for a claim where being wrong would cost someone (easy to reverse and affecting nobody: note the concern and what would settle it):

- acting would decide something for an identifiable person or commit an organisation (regulated areas are worth naming because decisions there fall on people [R5, Annex III], but the trigger is the decision, not the topic);
- the decision is not yours;
- you cannot construct an independent check;
- your checks disagree;
- you notice you are looking for a reason to accept the answer – the easiest to miss. In a study published in 2024, of assistants from 2023, those trained on human feedback shifted towards what the user appeared to believe [L5, sec. 3.3].

Health example. Keep apart "does the source say this?" – yours: open the guidance and read it; its possible presence in training data does not weaken a comparison against the document – and "is this right for this person?", which is the escalation. Similarly, in law the document cannot settle whether a rule applies to this case; in engineering a safety case needs a qualified signature, not "I verified it"; in research, escalate when a claim you cannot verify is load-bearing for a conclusion you intend to publish; in study, your institution's rules apply, and output that arrives without the work has skipped what the work was for.

High-stakes work is not off limits and there is no list of forbidden topics: the triggers concern your situation, not subject matter. The amount of checking scales with the cost of being wrong, and at some point the right amount is no longer available to you alone.

Unit 13: Privacy and data handling

What leaves the device. With a hosted service, assume what you submit may be transmitted to and processed by the provider and whoever it works with, unless that product's documentation and settings establish otherwise – prompt, conversation, attachments, anything a tool fetched. This is not specific to AI; AI changes how much is handed over and how casually, and connectors send material without pasting. Asking the assistant about its data handling is not a check; the policy and settings, read on a recorded date, are the evidence.

- **An attachment goes whole**, tracked changes, comments, hidden columns and other sheets included; whether all of it reaches the model is a separate question.
- **Earlier turns keep going**: history is processed again each turn, in whatever form is kept.
- **Deleting a conversation removes it from your view**; provider copies, backups and logs follow the service's policy.

Three questions for any service: is input used for training (may be a setting, may differ by tier)? How long is it stored and who can read it? Where is it processed, under what terms? Being unable to answer is itself the finding. One provider's documentation, as of November 2026, describes memory carried into later conversations, able to be on by default [D2, "What is Claude's memory?", "How Claude stores memory"], and shared links readable by anyone holding them, where copies outlive revoking the link [D3, "Frequently asked questions"].

What can happen to a copy – none requires the provider to break its word:

- **Kept and read** in logs, backups, abuse monitoring.
- **Learned from**: where input is used for training, a model can occasionally repeat it; researchers extracted hundreds of training passages verbatim from one model, including names, phone numbers and email addresses [L12, p. 2633].
- **Ordered kept**: in 2025 a provider was ordered, in a copyright lawsuit, to keep chat logs it would otherwise have deleted [D7, pars. 1-3].
- **Exposed by mistake**: in March 2023 a library bug showed some users others' conversation titles, and could show some subscribers' names, billing addresses and last four card digits [D6, "How did the ChatGPT data leak happen?"].

So ask whether it would matter if this copy got out.

Four bands, by who is harmed if the material becomes readable:

- **Public** – with nothing sensitive or identifying in it, settles the privacy question, mostly; not rights.
- **Yours alone** – with no other person's details in it, no third party exposed; not whether you are permitted.
- **Someone else's** – anything identifying another person or given in confidence (test: did somebody give me this expecting it to stay where I put it?); do not paste it as it stands.

- **Not yours to decide** – employer or client material, unpublished research, anything under an agreement (a dull internal spreadsheet often is); ask before.

The bands are one dimension, not a permission slip: rights, the service's terms and approvals, and assessment rules sit alongside.

Working with material you cannot paste. Change what you send:

- **Redact and generalise** – reduces risk but does not by itself make material anonymous or its use authorised; a role, a date and a place can identify someone.
- **Abstract the question** rather than send the document.
- **Bring the tool to the data** (local or approved service) – though a local application may still update, sync or send diagnostics; ask what leaves.
- **Watch re-identification**, especially in small institutions.

Unsubmitted coursework is "yours alone" for privacy; whether AI may be used on it is a separate question for institutional rules, which govern where they conflict with this.

Unit 14: Copyright and attribution

Three separate obligations:

- **Rights** – what may lawfully be done with the work.
- **Attribution** – who should be credited; what an honest account says.
- **Disclosure** – how it was made (Unit 17).

Rights and attribution come apart both ways: lawful use can be dishonestly presented; scrupulous credit can still infringe.

Copyright in generated output is genuinely unsettled, differently by country. Saying so is the answer, not a hedge.

Reasonably settled:

- Copyright protects a particular expression, not an idea, fact or style.
- Inputs keep whatever covers them; licences still bind.
- Output reproducing a substantial part of a protected work falls under the ordinary rules; machine production is no exemption.
- Terms of service bind by agreement and can restrict use of output; they do not themselves show that copyright exists, but may license or assign rights that do exist, subject to applicable law.

Not settled: whether output attracts copyright and whose; whether training on protected material without permission is lawful, including how far an opt-out counts (this unit reports no court outcome; where the law stands depends on country and date); obligations for output resembling a work without copying it.

Other rights. In the UK there is no single image right: a likeness is protected, if at all, through passing off, trade marks, contract and copyright [R1, "What is an 'image right'?"]; elsewhere a publicity right may exist [R2, "Legal protection of image rights"]. A copyright licence need not cover trade marks: the Apache licence, for one, grants no permission to use the licensor's trade marks [R4, sec. 6]. Most practical questions are answered by readable institutional or client rules and terms of service.

Attribution is a practice, not a permission:

- **Credit what a person made.** A summary that led you to a source does not make it uncited.
- **Never cite a source you have not opened**, and if you read only a report of it, say so, as the course's reference list does. Its `scripts/check_claims.py` shows that citations in the claim register resolve, not that sources were read.
- **Keep credit and disclosure separate.** Failures: disclosure standing in for credit ("written with AI assistance", no citations); credit standing in for disclosure (bibliography, no mention of assistance). Citations go at the claim, disclosure covers the work. A citation with a locator is a certificate a reader can check; "written with AI assistance" cannot be confirmed from the text and says nothing about correctness.

As of 2026, content credentials [V1, sec. 1.3], [V2, secs. 7.2.2-7.2.3] and text watermarks [V3, p. 823], [V4, "Limitations"] can label provenance. A label records origin, not truth; credentials can be stripped; watermarks mark only participating providers and weaken under rewriting or translation; a missing label shows nothing, and a label does not replace your own disclosure. None of these failures began with AI; it adds plausible

text and citations at volume. Whatever their origin, references to non-existent works appear in peer-reviewed papers, and one audit of biomedical papers found them more common since 2023 [L10, secs. 2.1, 4.1], [L11].

Do not credit the system as an author: an author is answerable for the work, and nothing in the system can answer for its claims. Record it as an instrument; if your referencing style asks for an entry for a tool or a retained chat, follow it, which is not authorship and does not replace citing the underlying sources.

Worked case. An overview with six citations; you read four and use two. Cite the two used; cite neither the unread two nor the read-but-unused two; disclose the AI overview, that you read the sources yourself and that the text is yours; follow your referencing style for the system. If the framing came from the summary, say so in the disclosure.

Unit 15: Bias, representation and accessibility

Ask not "is this biased" but: what would I compare this against, and what would that establish?

Where distributions come from. Output reflects training-data patterns, shaped by post-training.

- **Common in the data tends to be common in the output**: "a nurse", "a surgeon" tend towards what was common in pronouns, names, appearance. "Tends" matters: post-training, standing instructions and filtering can damp or introduce a pattern. Nothing in the system decided this. In a 2023 study of 2022 systems, 99% of one generator's "software developer" images were labelled white by a pre-trained classifier, against 56% in US statistics, and asking for the opposite did not reliably undo it [V6]; the point is the mechanism, not the figure.
- **Rare in the data is thin in the output**: smaller languages, places, minority practices – more likely generic, dated or wrong, with less to check against.

A skew can be visible; an absence does not announce itself. Both show in summaries, examples, and most consequentially in anything classifying people, where Unit 12's escalation row usually applies.

Three checks, each narrow:

1. **Vary one thing** (name, pronoun, place, language) – the nearest to a controlled comparison, a metamorphic relation [E20, sec. 2.2]. Two fresh conversations differ in the attribute *and* in run-to-run variation, so one pair settles nothing. Repeat in both orders and look for persistence. A repeated comparison establishes that outputs differed in these ways, at this frequency, on this system, on this date – not that no difference exists elsewhere, that its size is stable, or the mechanism, and with so few runs even a pattern could be chance.
2. **Compare against a real distribution** – a citable mismatch; a match is not a licence, since the real distribution may be the problem.
3. **Ask who is missing** – absences are easy to miss when your reference omits the same groups, so where you can, also compare with an independently assembled list.

None establishes fairness; a varied output is not a fair one, and whether the task was appropriate is a separate question, worth asking first when the task is about people.

Accessibility is the checkable part:

- alt text serving the image's purpose (generated alt text tends to describe contents); empty alt text for decoration; a chart's point in alt text, its numbers elsewhere;
- headings reflecting real structure;
- sufficient contrast;
- meaningful link text;
- colour never alone;
- keyboard access, visible focus, reading order.

A tool settles contrast, missing alt attributes, skipped heading levels; judgement settles purpose, structure and focus order. The course's `scripts/check_accessibility.py`, run on 2 November 2026, checked 121 pages, 64 animations and 65 contrast pairs, passed, and listed six checks for a human.

No comparative claim is made: such material predates AI. What differs is scale and speed; output often lacks provenance, as much human material also does.

Unit 16: The historical bug and historical honesty

The document. In 1843 Lovelace's translation of Menabrea's account of Babbage's Analytical Engine appeared in *Scientific Memoirs* vol. 3 with her Notes A to G [P2, pp. 666-731]. Note G's table sets out, operation by operation, a Bernoulli computation for her **B7**: she labels only non-zero Bernoulli numbers, so her Bk is the modern B(k+1), and B7 is the modern B8 = -1/30.

Established: the published table contains at least one error – a sentence that deliberately says nothing about where, what kind or whose.

Checked, and it holds. The common account, an inverted divisor and dividend, was checked against a facsimile [P2a]: **operation 4** acts on V5 and V4 in that order, while its stated result (2n-1)/(2n+1) needs V4 divided by V5. The copy is a processed image of the printed plate, in which the division sign is legible too.

Still contested: attribution. Compositor, Lovelace, Babbage – the table cannot settle it; "a typesetting error" is an inference. Unsourced: how long it stood, who first published a correction. "Contested" means sources disagree, or one account was copied forward without an independent check – not "probably false".

Why such errors survive. In print, in a respected series, under respected names, on a computation few would redo: each would plausibly reduce checking – a mechanism, not a finding. An error inside something trusted need only arrive where scrutiny was withdrawn.

The course's own mistake. The web says the table "as printed" gives -25621/630; the course recomputed 139/630 and published it as a refutation. Neither was complete. Besides operation 4, operation 24 as printed omits a sign inversion: the row has been read in the facsimile, and that it is a fault is Glaschick's reading [S7, sec. 5.2]; and earlier Bernoulli values may be supplied or generated by the routine. Under the course's recomputation:

- both repaired, values supplied: -1/30 (the intended computation)
- op 4 as printed, op 24 repaired, supplied: 139/630
- both as printed, supplied: -139/630
- op 4 repaired, op 24 as printed: 1/30
- op 4 as printed, op 24 repaired, values self-generated: -25621/630

Neither the web accounts nor the course had stated which object was executed. The fix: state the interpretation, recompute under it, treat a bare number as unfinished. Retrieval can amplify such figures: one major assistant, as of November 2026, may search the web unasked [D4, opening note], and repetition can look like corroboration.

AI as the newest case. Fluent text shares the protective properties – finished, consistent, no visible seam between right and wrong parts – so style and confidence are insufficient evidence. In 1843 every candidate for the error was a person; generation differs in scale, speed, variability and possible retrieval. What settled the question was a certificate: the facsimile.

Three buckets (not a probability scale):

- **Established** – supported by a primary source, or by consistent secondary sources with no serious dispute, and you can name the source you, or someone accountable, looked at.
- **Contested** – sources disagree, or one account copied forward unchecked; ten sources tracing to one are one source.
- **Unsourced** – no nameable source; not the same as false.

Most failures are misfilings into "established". Know what would move a claim: a facsimile or physical copy, compared print runs, period manuscripts, a study stating its copy.

Three strengths: "at least one error" (supported); "inverted operands in operation 4" (now supported [P2a]); "Lovelace made an arithmetic slip" (not made) – the most readable, hence the dangerous one.

Standing behind the work

Unit 17: Defence, disclosure, and responsible use

A **disclosure** answers how the work was made; a **defence** answers why it is right. Provenance is not evidence: an honest disclosure on a wrong answer is an honest wrong answer. Omitting a required declaration breaks the rule, but a reader holding your evidence can check the result whoever produced it, so evidence is worth more.

Ownership joins **understanding** (predicting what the thing does; checks count where their outcome was predicted first) and **correctness** (the thing meets the specification wanted). They come apart, and a result can also be mislabelled: the course's makers ran a faulty Note G reading, got 139/630, and called it "the table as printed". Pause when you cannot predict relevant behaviour, explain the transformation, name an independent check, or interpret a failed check.

A defence states what you checked, what it showed, and which specification it was against – the item usually left out. Verification: meets the specification; validation: the specification is right. Never cite the system's own verdict, confident or polished output, or the tool's name. Specially elicited or trained confidence measures can discriminate on some evaluated tasks, but calibration varies [L4, sec. 1.1]. **Independent-check principle:** the evidence must not be controlled only by the system under review.

Certificates let someone repeat a check and get the same result without trusting you: a command with range and output; a comparison table with sources; a quotation with locator; a facsimile. Checking is usually quicker than producing (multiply out a factorisation; machine-check a proof), so ask for checkable things. What a system supplies is a candidate until checked against something it did not produce; "I ran the tests" and displayed "reasoning" are generated text.

What a proof checker certifies. That the proof derives the formal statement from its hypotheses and definitions – not that they say what was meant, which is validation and stays with a person. In one 2026 preprint on a proof developed by researchers with AI agents and checked in Lean, review of the definitions caught an AI-generated protocol computing a key from both parties' strings combined, which neither held on its own; the paper notes its security definition alone would count an always-aborting protocol as secure; and it states openly that its protocol assumes an authenticated channel [L18, secs. II, III.B, IV.A]. A list of foundational axioms does not list every modelling assumption; inspect the hypotheses and definitions too (this course's inference). Read what was proved, under which assumptions, and whether each holds where you use it.

Disclosure gives what AI was used for, at which stage, and what you did afterwards, per the applicable rule.

A disclosure cannot be confirmed from the finished work; records may corroborate parts of it, without proving the account complete. As of 2026, detectors return likelihoods and err both ways; credentials [V1, secs. 1.1, 9.3.1], [V2, secs. 7.1.2, 7.2.3] and watermarks [V3, p. 823], [V4, "Watermark application", "Limitations"] mark only participating output and can be stripped or weakened. One 2023 study of seven detectors found false positives falling on non-native English writers [V5, Fig. 1]. Base rates: flagging one honest essay in a hundred gives about ten per thousand. So disclosure is not a control; a no-AI condition must be created by the setting. What carries weight is the work (checkable) and who is answerable (assistance does not move responsibility). Disclosure is a duty you discharge, and the finished work cannot confirm it. None of this is new; AI changes mostly speed and volume.

"Originate." Lovelace wrote the Engine "has no pretensions whatever to originate any thing". Wrong reading one: it settles language models – she described a fully pre-specified procedure; extension needs argument. Wrong reading two: it is obsolete – that depends on what "originate" means. Four readings:

- **Novelty** – Turing took up "Lady Lovelace's Objection" in 1950, replying among other things that machines often surprise us [T14, sec. 6].
- **Creativity** – Boden: new, surprising, valuable, by combining, exploring or transforming [T15].
- **Understanding** – Searle: running a program does not suffice [T16]; Bender and Koller: form alone cannot yield meaning [L13, p. 5185].
- **Consciousness or intention** – no agreed theory; a 2023 indicator assessment suggested no current system was conscious, and no obvious barrier to one meeting the indicators [L14, p. 1] – an answer only as good as the theories, which disagree.

Making them concrete moves the debate to which notion should matter; the course does not resolve it. You remain answerable for generated material you use.

Three words without agreed definitions. A claim that does not say which sense it means cannot be checked.

- **Understanding** – Searle's Chinese Room: a man following a program for Chinese understands none of it, so a program alone cannot give understanding [T16]. The **systems reply**: he is one part; the whole system understands. Searle: let him memorise the program; he is then the whole system and still does not understand. Critics: that examines a part again; and Searle, who holds that brains are machines that think, does not specify the brain's causal powers precisely enough to tell whether anything else could have them (Gardiner, as summarised there: Searle owes a more precise account of intentionality);

"still no consensus" [T21, secs. 3, 4.1, 5.2, "Conclusion"]. The course's working account concerns your own model of your work and does not answer Searle.

- **Intelligence** – some seventy definitions, none standard; Legg and Hutter adopt "Intelligence measures an agent's ability to achieve goals in a wide range of environments" and think one definition possible [T17, secs. 1, 5]. Turing replaced "Can machines think?" with a test of behaviour [T14, sec. 1]. Chollet: task skill alone is insufficient to measure intelligence, since priors or data can buy skill; intelligence as skill-acquisition efficiency over a scope, without claiming it is the "one true" definition [T18, secs. II.1.1, II.2.1]. Of "more intelligent", "human-level", "general": was anything measured, in what sense, on cases not seen?
- **Consciousness** – no canonical meaning [T22, sec. 2.3]; "something it is like" (Nagel) [T19]; the hard problem [T20, sec. 2]; functionalist, global-workspace, higher-order, integrated-information and neural theories [T22, secs. 8.1-8.2, 9.1, 9.5-9.7]; illusionism: phenomenal consciousness, in the philosophers' sense, is an introspective illusion, introspection misrepresenting experiences as having phenomenal properties [T23, sec. 1.1]. **The course's position is on method:** no agreed definition or theory, and the 2023 assessment took its indicators from several theories [L14, p. 4]; a system's report about itself is more output and settles nothing by itself; so "this AI is conscious" and "no AI could be" cannot be checked as stated – ask which theory, which indicators, what evidence; accountability does not wait.

Generally: disclosure carries **provenance**, a defence carries **warrant**, whose support must suit the claim and be separable from it.

Unit 18: Alles Lüge! First- and second-order reasoning

Established, published material, apart from the course's own working account of understanding, labelled as such; the unit proves nothing about language models, and nothing here establishes how often a model is right or wrong.

The movement ("all lies!"), marked by the songs [Q1], [Q3], [Q2]: everything sounds like a lie; the standpoint above is gone and work continues from inside; what is left is a method, warrant assembled locally.

The question: where is the observer standing? Index a claim by **observer**, **context** (convention, specification, limits) and **record** (retained evidence).

Cybernetics – control and communication in the animal and the machine [C9]; the Macy Conferences (ten meetings, 1946-1953, chaired by McCulloch, later titled "Cybernetics") met on circular causal and feedback mechanisms [C4, "Part 1"].

- **First-order** – *observed* systems; observer outside.
- **Second-order** – *observing* systems; observer included.

Von Foerster formulated it [C1, p. 303]; Mead's 1968 "Cybernetics of Cybernetics" is commonly cited as naming the reflexive turn [C2]. Observing from inside costs and can change the observed, so on this view a claim carries an index. Maturana: *anything* said is said by an observer [C3, p. 8] – not "everything", a later essay title propagated by repetition. Not the logical first/second-order distinction. Indexing makes claims more checkable. It is a tradition and **interpretive lens**, not a theorem.

Mapping – a reading, not a proof: the oracle, not the AI, settles the routine (check from outside); B1 = -1/2 and +1/2 both correct (supply the index); Ada's and modern B7 (two contexts, one symbol); oracle trusted by independent agreement, not a view from nowhere; disclosure versus defence (provenance versus warrant). On one working structural account (the course's own, not a published result or a settled definition), your model of something supports understanding when its parts and predictions keep corresponding to the target under checks (applied to the learner's model, never used to say whether a system understands; checks are evidence of it where their outcome was predicted first) – separate from the target's correctness. Where "the system" ends is disputed in the Chinese Room (the man, the man with his program, the program in his head) [T21, sec. 4.1]; for a language model it may be the weights, the weights with a prompt, or a deployed product. Stating the boundary makes clear which question a claim asks, without settling Searle's argument.

Classical results (a formal system has explicit symbols, assumptions and rules; a truth predicate classifies its own sentences as true):

- **Goedel's first incompleteness theorem** (1931): any **consistent** (never proves a statement and its negation), **effectively axiomatised** system proving basic arithmetic has a sentence it can neither prove nor refute; Goedel's "nor refute" [C6, pp. 187-189] used omega-consistency, removed by Rosser in 1936 [C8, p. 87].

- **Goedel's second incompleteness theorem** (same paper): such a system, with a little more arithmetic, cannot prove its own consistency [C6, p. 196], [T24, sec. 1.1]; a limit on derivation inside the system, not on showing its consistency by other means.
- **Tarski's undefinability theorem** (Polish 1933; German text 1935 or 1936, year disputed): arithmetical truth cannot be defined by a formula of that same language [C7].

Conditions are load-bearing: without effective axiomatisation the theorem is false (true arithmetic is consistent, complete, unlistable [T24]). The unprovable sentence's *truth* holds under the intended interpretation; provability belongs to the system, truth is relative to a model.

The missing middle. A language model is a physical system studied empirically. Carrying a theorem across needs: which component is the formal theory, that its conditions hold, that conclusions transfer to the running system. Asserting the conclusion without that argument is the fault; it may be true for other reasons. Turing's result [T10, sec. 8], the halting problem [T12, sec. 2.4.3] and Rice's theorem [T11, pp. 364-365], [T13, sec. 3.2] concern programs, ruling out general procedures, not checking one program over a stated range. A trained model runs as, or inside, a program; these limit procedures meant for all programs, and whether they limit one meant for trained models depends on the class those form (precision, context, the generation loop, tools). It is not a formal theory (no proof rules, can produce a claim and on another run its negation), so Goedel's and Tarski's conditions are not shown to hold. Idealised hard-attention transformers are Turing complete [L15, Theorem 6]; with logarithmic precision, far weaker if L is not P [L16, p. 531].

The middle, for a guarantee. A machine-checked theorem holds for its formal model; carrying it to a system needs the model to be faithful and its assumptions to hold there. One 2026 preprint reports a theorem checked in Lean for a model that assumes an authenticated channel, as it states; whether a system meets that is outside the proof. It also describes gaps in its field between individually valid theorems combined under mismatched assumptions [L18, sec. IV.A, abstract]. A chain of correct steps is only as good as the agreement of the conditions where they meet.

First order, mostly – a reading, not a result: do not assume output records its standpoint (training, cut-off, framing prompt) – ask for it and check what you can; weights stay fixed in ordinary use, though one provider documents, as of November 2026, memory carried into later conversations [D2, "How Claude stores memory"]; its step-by-step explanation can misstate what drove its answer (2023 study, multiple-choice tasks) [L9, sec. 3.2]; self-correction without outside feedback was found (2024 study) to help little on reasoning tasks, sometimes making answers worse [L17, sec. 3.2]. Second-order work is done by the person, or a design placing the check outside the model.

Not licensed: "nothing can be known" (scoped conclusions beat felt certainty); "AI cannot be trusted at all" (the observer question applies to humans and Note G; the response is architectural); "Goedel and Tarski prove AI cannot verify itself" (true halves, unargued join; the second theorem limits a formal system proving its own consistency, not a model checked against an independent reference). Warrant is built from below: observer, context, record, independent agreement – less than certainty, and checkable.

The standard

One standard for any work you submit, whatever produced it: *I understand the claim, I gathered independent evidence appropriate to it, I can hand over that evidence so someone else can check it, I can defend its scoped correctness, and I take ownership of using it.* "I understand the claim" is not a feeling: you can predict relevant behaviour, explain the transformation at the level the claim needs, name an independent check, and interpret a failed one.

Course content is CC BY-SA 4.0 and software AGPL-3.0-or-later; third-party material is outside both.

v0.1.0 (commit 9331b46, 2026-11-05), built 2026-11-05 14:25 UTC. Copyright (C) 2026 Frank C. Langbein and contributors. Licences: CC-BY-SA-4.0 (course content and media); AGPL-3.0-or-later (software).