

TEACHING PORTFOLIO

Trevor T. Spinosa · Louisiana State University
tspino2@lsu.edu · trevorspinosa.com

1. EXISTING COURSE: ETHICS IN COMPUTING

Course and role. I developed my own version of LSU's CSC 3200, Ethics in Computing, including the syllabus, slides, Moodle site, and assessments. I have taught one section each term since Spring 2025; Fall 2026 is my fourth semester. Approximately 400 students have taken the course across those terms. It is required in the Software Engineering and Cybersecurity concentrations.

An actual session. In Fall 2025, the privacy session included a debate about government backdoors in encrypted communications. Students had five minutes to prepare and five to argue. Other sessions connect an ethical framework to computing cases and questions that ask students to defend a position. Four forum posts and an essay extend that reasoning into writing.

Existing essay assignment. Students analyze a computing-ethics issue, choosing AI and bias, privacy and surveillance, or another course topic. The assignment calls for a clear thesis, at least three credible sources, and analysis. I grade the essays by hand against the five posted criteria below. This table summarizes the existing criteria; it does not reproduce a new scoring scale.

Existing criterion	What the essay asks students to demonstrate
Content	Address a computing-ethics issue with relevant support.
Organization	Present a thesis and a coherent argument.
Analysis	Reason about the issue, beyond describing a technology.
Writing quality	Express the argument clearly.
Adherence to guidelines	Meet the assignment's stated requirements, including sources.

Reflection on assessment. In Spring 2025, students took an ethics quiz, generated a song on the topic with Suno, and repeated the quiz. More than 50 songs came back, but the typical pre-activity score was already perfect. The quiz could not establish a learning gain. In Spring 2026, I began adding scenario questions. I presented the activity virtually at IEEE ISEC in March 2026. This experience supports a change in assessment design; it does not establish the activity's effectiveness.

Support in the existing course. After a student was failing because they had missed every forum post, I offered a makeup assignment intended to prompt discussion of favorite course topics. I made it available to every student who had performed poorly on the forums. Both textbooks are optional references, and the Fall 2026 calendar spaces forum deadlines two weeks apart.

2. PREPARED EXAMPLE: FEEDBACK ON AN ETHICS ARGUMENT

Status. The passage and feedback below were newly prepared for this portfolio. They are an illustrative teaching example, not student work or feedback previously delivered.

Illustrative passage

A driving app should be allowed to collect location because it can help an insurance company price policies. Users agree to the app, so the collection is ethical.

Model feedback

You identify a possible benefit and take a position. Your next step is to explain what makes the agreement meaningful. What does a driver understand about the precision of the location data, its retention, and who can access it? Separate the data needed to assess driving behavior from data that could reveal unrelated activities. Use a credible source to establish a relevant fact, then apply an ethical principle to explain why that fact matters. Address the strongest counterargument to your position. A revised conclusion may support collection, limit it, or reject it; it needs to follow from your reasons.

Proposed revision prompt. Write a paragraph identifying one stakeholder, one factual claim requiring evidence, and one ethical obligation. Explain how the obligation applies, then answer a counterargument. Identify a fact that would make you reconsider your conclusion.

What I would look for. A stronger response would distinguish consent from informed understanding, connect a factual claim to an appropriate source, and explain how an obligation supports the conclusion. Merely adding ethical vocabulary would not satisfy the analysis criterion.

Proposed classroom use. Students would first annotate the passage individually, then compare one revision priority with a partner. I would collect a short written response before inviting public discussion. This would give every student a way to formulate an answer. I would use the responses to decide whether to revisit the distinction between a technical description and an ethical argument.

3. PREPARED CS1 LESSON: COUNTING WITH A LOOP

Status and audience. A new 15-minute lesson for students who have seen variables, lists, comparisons, and `for` loops. It has not yet been taught.

Objective. Trace and explain a loop that counts values at or above a threshold; distinguish the current value from the accumulated count.

0-2 minutes: predict. Display `[3, 8, 5, 8]`. Ask how many values are at least five. Students write a count and underline the qualifying values. The answer is three: the qualifying values are 8, 5, and 8.

2-6 minutes: worked example. Show the code and trace each iteration. Explain that the counter is initialized once and increases by one for each match.

```
def count_at_least(values, threshold):
    count = 0
    for value in values:
        if value >= threshold:
            count += 1
    return count
```

Current value	At least 5?	Count after this step
3	No	0
8	Yes	1
5	Yes	2
8	Yes	3

6-10 minutes: guided practice. Pairs trace `[5, 1, 7, 5]` with threshold 5. One student explains each condition; the other records the count. They then switch roles to trace an empty list. Expected results are 3 and 0. Ask why the function returns zero when no iteration occurs.

10-13 minutes: misconception check. Replace `>=` with `>` and predict the first example's result: 2. Restore `>=`, then replace `count += 1` with `count += value`: the result becomes 21. Ask which version counts matches and which sums matching values. Run the examples only after predictions.

13-15 minutes: exit question. Move `count = 0` inside the loop and trace `[5, 4, 6]` with threshold 5. Why does this version return 1 instead of 2? Repair it and explain the repair. With initialization only inside the loop, an empty list would also leave `count` undefined. The repair is to initialize before the loop. If explanations still confuse values with counts, I would use another trace before introducing a second accumulator.

Companion files. `cs1_counting.py` and `test_cs1_counting.py` provide the worked implementation and runnable checks, including boundaries and empty input. The checks support practice; the explanation remains part of assessment.

4. PROPOSED CS2 ASSIGNMENT: SUMMARIZE REQUEST RECORDS

Status and objective. A new assignment on dictionaries, decomposition, and testing. Students build an index from synthetic website request records. It has not yet been assigned.

Task. Implement `summarize_requests(records)`. Each record is a `(path, status)` pair, such as `( '/search', 200)`. Return a dictionary mapping each exact path to `{ 'requests': n, 'errors': e }`. Count every record, including duplicates. An error has status 400-599. Empty input returns an empty dictionary; do not mutate the input. Inputs are guaranteed to contain nonempty string paths and integer statuses from 100-599. Parsing and input validation are outside this assignment.

For `[('/search', 200), ('/search', 404), ('/help', 500)]`, the result is `{ '/search': { 'requests': 2, 'errors': 1 }, '/help': { 'requests': 1, 'errors': 1 } }`. Path comparison is case-sensitive. Output ordering is irrelevant.

Sequence. First, draw the dictionary after each record. Next, implement and test a single path; then handle repeated and multiple paths. Finally, add student-written tests and a short explanation of the representation and running time. Students would receive a checkpoint before the complete submission. All examples are synthetic and require no account, network access, or paid tools.

Proposed rubric	Points	Evidence
Functional correctness	40	Meets the documented behavior, including empty input and boundary statuses.
Representation and decomposition	20	Groups records appropriately and uses a clear, maintainable structure.
Student-written tests	20	Covers meaningful cases and explains what each checks.
Explanation	15	Traces an example and explains the expected linear-time scan and space use.
Readability	5	Uses clear names and consistent formatting.

Public tests give early feedback, but points follow the rubric rather than a count of passing tests. A working program without a defensible explanation would not earn full credit. The companion student starter and public tests are separate from a labeled instructor solution. Students would submit code, their tests, and the explanation.