COURSE SYLLABUS

CS120 • Computational Thinking for Everyone

School of Computational Arts & Sciences • Undergraduate • 6 ECTS

Overview

Learn computational thinking as a transferable method: decomposition, abstraction, representation, and systematic debugging. Students solve small problems with simple programs and visual reasoning, practicing how to explain solutions and verify them with tests and examples.

LOGISTICS

Credits: 6 ECTS
Level: Undergraduate
School: School of Computational Arts & Sciences
Prerequisites: None listed
Tags: fundamentals, problem-solving
Meeting time: 2×80min per week + short studio exercises
Instruction mode: Hands-on: small problems, visible reasoning, frequent feedback

LEARNING OUTCOMES

You will be able to:
- Translate real problems into steps, invariants, and test cases
- Communicate algorithms using clear diagrams and structured explanations
- Recognize common bug patterns and apply systematic debugging
- Translate real problems into steps, invariants, and test cases
- Communicate algorithms using clear diagrams and structured explanations
- Recognize common bug patterns and apply systematic debugging

ASSESSMENT

Components
- Coursework: 60%
- Final project: 40%

Assessment is portfolio-based: short exercises, reflections, and two small projects. We
grade clarity of reasoning and your ability to revise after feedback.

WEEKLY PLAN

Schedule
Week 1: Week 1
  - Decomposition: break a task into subproblems and interfaces
Week 2: Week 2
  - Abstraction: data representations and naming
Week 3: Week 3
  - Algorithms: loops, conditionals, and invariants
Week 4: Week 4
  - Debugging: tracing, hypothesis testing, and minimal reproductions

Extended outline
- Decomposition: break a task into subproblems and interfaces
- Abstraction: data representations and naming
- Algorithms: loops, conditionals, and invariants
- Debugging: tracing, hypothesis testing, and minimal reproductions
- Complexity intuition: when a solution will not scale
- Final: a small system with explanation and tests

POLICIES & RESOURCES

- Bring questions early: blocked work is a signal to adjust the plan.
- Revision is expected: resubmissions are allowed for selected work.
- Academic integrity: collaboration on ideas is fine; submit your own artifacts.

Suggested resources
- Problem-solving worksheet: inputs, outputs, cases, invariants
- Debugging checklist: reproduce, minimize, instrument, verify
- Reading: short notes on algorithms as stories