COURSE SYLLABUS

CS101 • Introduction to Programming

School of Computational Arts & Sciences • Undergraduate • 6 ECTS

Overview

An introduction to programming through small, complete systems: variables, control flow, functions, and working with files. Strong emphasis on readable code, incremental testing, debugging via tracing, and using version control to iterate with feedback.

LOGISTICS

Credits: 6 ECTS
Level: Undergraduate
School: School of Computational Arts & Sciences
Prerequisites: None listed
Tags: programming, fundamentals
Meeting time: 2×90min per week + 60min lab section
Instruction mode: In-person with structured lab check-ins; hybrid office hours

LEARNING OUTCOMES

You will be able to:
- Write small, readable programs with tests
- Explain and debug program behavior
- Use version control and feedback loops to improve code
- Design functions with clear contracts and test cases
- Use debugging tools (tracing, assertions, logging) to localize faults
- Write small programs that read/write files and validate input

ASSESSMENT

Components
- Weekly labs: 40%
- Problem sets: 30%
- Final project: 30%

Grading emphasizes correctness, clarity, and evidence. Each submission includes a short log:
what you tried, what failed, what you changed. Late work may be accepted with a small
penalty when accompanied by a concrete recovery plan.

WEEKLY PLAN

Schedule
Week 1: Thinking in steps
  - Variables and control flow
  - Tracing
  - Simple tests
Week 2: Functions and decomposition
  - Parameters
  - Return values
  - Refactoring
Week 3: Data and files
  - Collections
  - Parsing
  - Common errors
Week 4: Projects
  - Requirements
  - Iteration
  - Presentation

Extended outline
- Setup + tooling: editor, formatter, test runner, version control workflow
- Data types + control flow: build a small command-line tool
- Functions: decomposition and refactoring with tests
- Collections: lists/maps; common iteration patterns
- Files + parsing: structured text; error handling
- Mini-project: implement and present a small system with a postmortem

POLICIES & RESOURCES

- Academic integrity: cite any external code or ideas; explain your changes.
- Sustainability: you are graded on traceable iteration, not heroics.
- Accessibility: accommodations are welcomed; contact the instructor early.

Suggested resources
- A short style guide: naming, indentation, and small-function discipline
- Testing primer: writing assertions that fail usefully
- Debugging checklist: reproduce, minimize, instrument, verify