CSSE1001 — Introduction to Software Engineering

Table of contents

Overview

Introduction to Software Engineering through programming, with a focus on the fundamentals of computing and programming using an exploratory, problem-based approach: building abstractions with procedures, data and objects; data modelling; designing, coding, and debugging programs of increasing complexity. Taught in Python. The GUI module has been removed from the scope of this course to leave more time for fundamentals.

No prior programming knowledge is assumed. Incompatible with COMP1502, CSSE7030, ENGG1001.

Staff

- **Dr Paul Vrbik** — course coordinator & lecturer — CSSE1001@eecs.uq.edu.au — office hours by appointment
- **Dr Kasra Khosoussi** — lecturer

Textbook, if any, isn't in the source material yet — add it here if it comes up.

Course structure (by week)

Week	Topic
1	Course introduction. Arithmetic, types, and variables in programming.
2	Python memory model and primitive data.
3	Functions and if-statements.
4	While-loops and data.
5	For-loops and list comprehensions.
6	IO, string methods, testing and debugging.
7	Exceptions, scope, and intro to OOP.
8	Dunder methods and representation invariants.
9	Composition and inheritance.
10	Unified Modelling Language.
11	Design patterns and MVC.
12	Recursion.
13	Functional programming. Complexity.

Applied classes (weeks 2-13) and practicals run alongside lectures, giving guided/self-directed practice with demonstrator feedback. 12 formative Ed Lessons tasks (immediate feedback) and Gradescope (immediate feedback on assignment submissions) support self-study.

Learning outcomes

After successfully completing this course you should be able to:

1. Apply program constructs such as variables, selection, iteration, and sub-routines.
2. Apply basic object-oriented concepts such as classes, instances, and methods.
3. Read and analyse code written by others.
4. Analyse a problem and design an algorithmic solution to it.
5. Read and analyse a design and translate it into a working program.
6. Apply techniques for testing and debugging.

Assessment

Assessment	Weight	Due	Covers
Assignment 1	15%	19/09/2025, 3:00pm	Weeks 1-4 (functions, lists, tuples, dictionaries, loops, if-statements)
In-semester exam (Saturday)	25%	Sat 13 Sep 2025, 7:15pm, 90 min	Up to week 5
Assignment 2	20%	24/10/2025, 3:00pm	Weeks 1-11
End-of-semester exam	40%	End-of-semester exam period (8/11/2025 - 22/11/2025), 120 min	Entire course. **Hurdle**: exam $\geq$ 45% required for Grade 4 and above.

Both exams are closed-book, identity-verified, in-person, and paper-based (calculator: Casio FX82 series or UQ-approved/labelled calculator only).

Grading formula

With A_1, A_2, EM, EF as percentages for Assignment 1, Assignment 2, the in-semester exam, and the end-of-semester exam respectively:

$$\text{MARK} = \max(M_1, M_2)$$

where

$$M_1 = 0.15A_1 + 0.20A_2 + 0.25EM + 0.40EF, \qquad M_2 = 0.15A_1 + 0.20A_2 + 0.10EM + 0.55EF$$

So a poor in-semester exam result can be mitigated by a strong final exam.

Assignments are submitted online via Gradescope. Extensions: up to 7 days maximum (in multiples of 24 hours), handled by the school (not the lecturer) — a medical certificate is required, with no exceptions. Work must be *received* by the deadline, not just transmitted; there is an undisclosed small grace-period buffer so work only a few minutes late isn't penalised. Beyond that, late assignment submissions receive 0%.

Academic integrity

- Don't cheat — penalties are harsh, and detection tools are sophisticated.
- Students who maximise assessment scores by using ChatGPT, abusing staff support, or cheating typically fail the exam.
- You may discuss solutions to problems with others, but must not share code — including for assignments.

Course resources

- **IDLE** — the IDE officially supported by tutors.
- **Open Help Center** (a.k.a. "ITLC"/"EECSLC") — 78-217 — Mon, Tue, Wed, Fri 10am-4pm.

Setting expectations

- Lectures are only a **first exposure** to a topic — you're expected to do most of your learning by completing assigned tasks, and to revise both before and after lectures.
- Secondary school vs. university, per the source: 20 weeks / 5 hrs-per-week / 100 hrs instruction / 30 classmates (secondary) vs. 12 weeks / 2 hrs-per-week / 24 hrs instruction / 1000 classmates (this course). Teachers *teach* you; instructors *guide* you.
- Course staff are human — they will sometimes make errors, be terse, and are often sleep-deprived, despite trying hard to provide a good learning experience.
- **How to succeed**: (1) practice makes perfect, (2) learn to debug, (3) do well on the exam.

Lectures

- 2025-07-29-python-as-a-calculator[1] — Lecture 1B
- 2025-08-04-python-memory-model[2] — Lecture 2A
- 2025-08-04-primitive-data[3] — Lecture 2B
- 2025-08-11-functions[4] — Lecture 3A
- 2025-08-11-sequence-selection-and-iteration[5] — Lecture 3B
- 2025-08-18-while-loops[6] — Lecture 4A

[1] courses/csse1001/2025-07-29-python-as-a-calculator.pdf
[2] courses/csse1001/2025-08-04-python-memory-model.pdf
[3] courses/csse1001/2025-08-04-primitive-data.pdf
[4] courses/csse1001/2025-08-11-functions.pdf
[5] courses/csse1001/2025-08-11-sequence-selection-and-iteration.pdf
[6] courses/csse1001/2025-08-18-while-loops.pdf

- 2025-08-18-non-primitive-data[7] — Lecture 4B
- 2025-08-25-for-loops[8] — Lecture 5A
- 2025-08-25-comprehensions[9] — Lecture 5B
- [[week-five-exercises|For Loop Practice]] — Lecture 5C
- 2025-09-01-string-methods[10] — Lecture 6A
- 2025-09-01-file-io[11] — Lecture 6B
- 2025-09-01-testing[12] — Lecture 6C
- 2025-09-09-exceptions[13] — Lecture 7A
- 2025-09-09-scope[14] — Lecture 7B
- 2025-09-09-object-oriented-programming[15] — Lecture 7C
- 2025-09-16-dunder-methods[16] — Lecture 8A
- 2025-09-16-representation-invariants[17] — Lecture 8B
- 2025-09-23-composition[18] — Lecture 9A
- 2025-09-23-inheritance[19] — Lecture 9B
- 2025-10-07-uml[20] — Lecture 10A
- 2025-10-07-advanced-inheritance[21] — Lecture 10B
- 2025-10-14-model-view-controller[22] — Lecture 11A
- 2025-10-20-recursion[23] — Lecture 12A

[7] courses/csse1001/2025-08-18-non-primitive-data.pdf
[8] courses/csse1001/2025-08-25-for-loops.pdf
[9] courses/csse1001/2025-08-25-comprehensions.pdf
[10] courses/csse1001/2025-09-01-string-methods.pdf
[11] courses/csse1001/2025-09-01-file-io.pdf
[12] courses/csse1001/2025-09-01-testing.pdf
[13] courses/csse1001/2025-09-09-exceptions.pdf
[14] courses/csse1001/2025-09-09-scope.pdf
[15] courses/csse1001/2025-09-09-object-oriented-programming.pdf
[16] courses/csse1001/2025-09-16-dunder-methods.pdf
[17] courses/csse1001/2025-09-16-representation-invariants.pdf
[18] courses/csse1001/2025-09-23-composition.pdf
[19] courses/csse1001/2025-09-23-inheritance.pdf
[20] courses/csse1001/2025-10-07-uml.pdf
[21] courses/csse1001/2025-10-07-advanced-inheritance.pdf
[22] courses/csse1001/2025-10-14-model-view-controller.pdf
[23] courses/csse1001/2025-10-20-recursion.pdf