Lecture 0: Welcome to Maths

ISE Block 1.2 Maths · Week 1

Why maths?

When will I ever use this?

- Your maths teacher probably promised you'd use algebra in the real world
- Then you discovered that jobs rarely involve factorising quadratics
- Computer science is different: mathematics sits just beneath the code
- Sometimes it's the difference between a system that *seems* to work and one you can show will work

An unbreakable cipher?

- **By the end of week 1:** you'll build a one-time pad and prove why it offers *perfect secrecy*
- When its conditions hold, the ciphertext reveals no information about the message
- Then we'll reuse a key and see what goes wrong
- The maths is sound. The difficulty is meeting its conditions in practice

The maths underneath the code

- **LLMs and neural networks:** linear algebra and calculus, including backpropagation
- **Algorithms:** asymptotic notation to understand how runtime and memory use scale
- **Public-key cryptography:** number theory behind using a public key and a private key
- **Correctness:** logic to show what a program does, rather than just test a few examples

You’ve already met a little of this

- `def total(...) -> int:` says what kind of result a function should return
- `Literal["pending", "paid", "cancelled"]` restricts the permitted values
- A type checker can catch violations before you run the code
- Types don’t prove everything: a function can still loop forever or return the wrong integer
- We’ll learn to state and justify stronger claims about programs

The plan

Weeks 1 to 7

- **Week 1:** knowing you’re right: proof, logic and a perfect cipher
- **Week 2:** the language of specifications: quantifiers, sets and functions
- **Week 3:** linear maps and induction
- **Week 4:** number theory and cryptography
- **Week 5:** graphs and growth
- **Week 6:** counting and chance
- **Week 7:** assessment
- We’ll check the pace after week 1 and adjust if needed

How a week runs

The timetable

Day	09:00–11:00	11:00–12:00	14:00–16:00
Tuesday	Lecture	Lab with lecturer	Lecture
Friday	Lecture	Lab with teaching assistant	

Lectures

- One topic per lecture, following the textbook’s order (mostly)
- The slides contain the material taught and can be used for revision
- Each deck appears on this site on the day of the lecture
- We will read mathematical expressions aloud and, where helpful, run them in Python

Labs

- Work through problems in a notebook: calculate, test, find counterexamples and write arguments
- Tuesday's lab uses that morning's lecture, with the lecturer present
- Friday's lab revisits earlier material, with the teaching assistant
- Download notebooks from this site on the day of the lab
- Use Google Colab, Jupyter or VS Code; bring a laptop
- **Submission instructions, deadlines and marking arrangements are on Brightspace**

Starting points

What I assume about you

- You’ve completed seven weeks of programming and can write small Python programs
- You’ve studied Leaving Certificate mathematics, including algebra, functions and calculus
- …it’s okay if you haven’t remembered everything; we’ll refresh the maths as needed
- Formal proof may be new; we’ll work through it from the start

How to work

- Try each problem before looking at a solution
- Write down your reasoning, not just the answer
- If a proof fails, find the step you can’t justify
- Ask when a definition or step isn’t clear
- Read the textbook sections listed with each lecture

Assessment and academic integrity

Assessment: what to expect

- Continuous assessment on Brightspace, weekly in weeks 1–6 (20%): short retrieval quizzes, lab checks and written proof tasks
- Two in-class tests, in weeks 3 and 5 (25%)
- A capstone project, due at the end of week 6 (15%)
- A final exam in week 7 (40%)
- **In-class tests are closed-book**
- **The final exam is closed-book**
- End-of-block assessment takes place in **week 7, 14–18 December**, during teaching weeks; there is no separate exam week
- Assessment weights, test format and the final exam's precise arrangements will be published on Brightspace
- **Brightspace is the source of truth for assessment and submissions**

Academic integrity

- Submit your own work
- **Do not use generative AI in assessed submissions unless the assessment explicitly permits it**
- Where AI use is allowed, follow the assessment’s conditions and declaration requirements
- Discussion and collaboration are encouraged for learning, but submitted work must comply with the rules for that assessment

AI and learning

AI can help... but it can also hide gaps!

- A model can explain a concept, check code or suggest a proof
- It can also give you a convincing answer that is wrong
- Completing a problem with help doesn’t show you can solve it unaided
- Research on AI-assisted study has found both gains and setbacks, depending on the tool and how it is used (Kestin et al. 2025; Bastani et al. 2025)

Use it as a critic, not a substitute

- Try the problem yourself first
- Ask for a hint, or for a flaw in your proof
- Check the answer against definitions, calculations and the textbook
- Then try a similar problem without assistance
- **Studying with AI does not give you permission to use it in assessed work**

The text

The textbook

- Lehman, Leighton and Meyer, *Mathematics for Computer Science*, 2015 revision
- Written for MIT's mathematics for computer science course
- Free from MIT OpenCourseWare; link on the course website
- Lecture decks identify the relevant sections; we may select or reorder material to suit the teaching sequence

When books disagree

- Mathematical notation is not always uniform
- Example: some texts include 0 in the natural numbers, and others start at 1
- Here, $\mathbb{N} = \{0, 1, 2, \ldots\}$
- Check which convention a source uses, and state yours where it matters

The slides

- Use the slides for revision alongside the textbook
- Find the latest published version at maths.eoin.ai
- For assessment rules, use **Brightspace**

The lecturer

The boring intro slide

- **Eoin O'Brien**, Assoc. Prof. (Practice) in Software Engineering, University of Limerick
- Sat on the Industry Advisory Board for ISE when the course was being set up
- Founder and CTO of Stokka, building software for manufacturing, stock management and production planning.
- Previously co-founded Tracworx while an undergraduate at UL. Developed hospital patient tracking, Bluetooth-based contact tracing (COVID-19) and supply-chain tracking systems.
- Lecturing at UL: software testing, cryptography, data structures and algorithms, AI/ML
- Research: part of the ABCD (AI for software engineering) and STEER (entrepreneurship) groups
- Irish 30u30, InterTradeIreland Seedcorn, UL Gold Medal etc.

References

Bastani, Hamsa, Osbert Bastani, Alp Sungu, Haosen Ge, Özge Kabakcı, and Rei Mariman. 2025. “Generative AI Without Guardrails Can Harm Learning: Evidence from High School Mathematics.” *Proceedings of the National Academy of Sciences* 122 (26). https://doi.org/10.1073/pnas.2422633122.

Kestin, Greg, Kelly Miller, Anna Klales, Timothy Milbourne, and Gregorio Ponti. 2025. “AI Tutoring Outperforms in-Class Active Learning: An RCT Introducing a Novel Research-Based Design in an Authentic Educational Setting.” *Scientific Reports* 15 (1). https://doi.org/10.1038/s41598-025-97652-6.

Questions?

eoin@eoin.ai · eoin.p.o.brien@ul.ie