# Data Structures and Algorithms

Spring 2026, BSc EFDS, Imperial Business School

**Module Lead:** Ryan Cory-Wright, r.cory-wright@imperial.ac.uk

**Tutorial Leads:** Jay DesLauriers, j.deslauriers@imperial.ac.uk
and Marius Mickunas, m.mickunas@imperial.ac.uk

## Contents

**Data Structures and Algorithms** **1**
- TL:DR . . . 1
- Before you get started . . . 2
- Objectives . . . 2
- Prerequisites and materials . . . 2
- Structure . . . 4
- Assessment . . . 5
  - Assessment marking . . . 5
  - Collaboration guidelines . . . 5

### TL:DR

**TL;DR (read this if nothing else).**

- **Format:** 10 sessions over 10 weeks. Each session has (i) preparation (async), (ii) a live class, and (iii) review (async). Expect to spend most of your time doing exercises.
- **Assessment:** Exam 60% (includes short Python snippets written on paper), session hand-ins 20% (graded most weeks but week one; announced in advance), and two individual homework assignments 20%.
- **Help:** Ask on Ed (monitored by tutors) or ask questions in lectures/tutorials.
- **Collaboration/AI:** Discuss ideas freely, but do *not* share/copy code for individual homework. Do not look up solutions or use generative AI to write code unless explicitly allowed.

## Before you get started

Welcome to Data Structures and Algorithms! The course is organized into ten sessions over ten weeks. Each session consists of a live lecture, asynchronous online materials, and a set of exercises.

## Objectives

This module will introduce you to computational problem-solving through the design of algorithms and data structures. Devising efficient methods for sifting through large datasets is at the core of most modern technological innovation, ranging from search engines and social networks to healthcare, energy and finance. Our objective is to learn to build algorithms: moving from a possibly ambiguous problem statement to formulating a computational solution method. We will introduce the basic principles of algorithm design and analysis through classic problems such as searching and sorting. We will also study algorithms on graphs, with wide applications in areas such as social networks.

We will implement our algorithms in Python, a versatile open-source language widely used in the economics and data science communities for tasks like data analysis, machine learning, web scraping, and natural language processing. Python is designed to be both powerful and easy to read, which makes it a great language for beginners to learn. Our objective is to build a strong foundation in the basic concepts of programming – such as variables, objects, loops, and functions – and learn to apply them in practical problem solving. We will also learn to read and build on other people's code, and introduce some key analytics and business use cases of Python: writing scripts to automate tasks, parsing and interpreting data, and scraping the web.

## Prerequisites and materials

This module assumes no previous programming experience. However, we will cover a lot of material, so if you have not done any programming before, some extra preparation may be helpful. We believe that the following strategies will be useful for most students:

- If you would like to get started on Python before the module, spend a few hours working through an online tutorial, for example the first two chapters of *Automate the Boring Stuff with Python* (see the materials below).

- During the module, make sure to go through the assigned readings and exercises before each lecture.
- If you get stuck while coding, ask for help on Ed!
  Mastering Python will require practice and hard work, but it's no use being stuck on a problem for hours.

Although the module materials are mostly self-contained, we recommend the following books:

- *Introduction to Computation and Programming Using Python* (Third edition, 2021), John Guttag, MIT Press.
  - We will cover approximately the first half of the book.

We will assign readings for each session from the Guttag book, but will also point to alternative readings that are freely available online. These will cover roughly the same topics, but may not always perfectly match the book, so we recommend getting the book.

**Other Python books**

- *Starting Out with Python* (Third edition, 2014), Tony Gaddis, Pearson.
  - Thorough on basic Python topics with lots of exercises.
- *Automate the Boring Stuff with Python*, Al Sweigart.
  - Free online book with lots of practical applications, available at https://automatetheboringstuff.com. Includes great video material too.

**Other algorithms books**

- *Grokking Algorithms* (2016), Aditya Bhargava, Manning.
  - Probably the best algorithms book to go with this module. Covers many basic algorithms in an intuitive way, though skipping mathematical details.
- *Introduction to Algorithms* (Third edition, 2009), Thomas Cormen et al., MIT Press.
- *The Algorithm Design Manual* (Second edition, 2008), Steven Skiena, Springer.
  - These are in-depth computer science textbooks on algorithms, including mathematical proofs.
- Algorithm Design by Jon Kleinberg and Eva Tardos. Pearson, 2006.
  - Harder than what we will see in this class, but good, e.g., if you would like to study up before a summer internship interview

## Structure

The module is divided into ten sessions over ten weeks. Each session consists of self-study online materials and exercises supported by live classes. You will work through each session in three parts:

1. **Preparation** (asynchronous). You will go through preparatory readings, videos, and exercises.
2. **Live class**. We will discuss and review the key ideas of the session in a live class. A part of the class will be devoted to working on exercises, supported by teaching assistants.
3. **Review** (asynchronous). You will complete a set of exercises to practice and review what you learned from each session.

The module is very hands-on, with many Python exercises in the sessions and homework assignments. Based on our experience, **you will likely spend more time working on exercises than reading and watching content**. See the material each week for further details.

If you have any questions on the materials, get stuck working on an exercise, or would like to discuss a topic, the easiest ways to get help are:

- Post a question on the module chat on Ed, which will be regularly monitored by tutors.
- Ask the instructor in a live lecture/tutorial. There will be time devoted to open questions in the end of each class.

Please see below for session details and preparation instructions.

| Session | Live class date | Class type |
|---|---|---|
| Session 1: Introduction to Python | 13/01/2026 | Lecture |
| Session 2: Algorithms and functions | 20/01/2026 | Lecture |
| Session 3: Sequential data (loops) | 27/01/2026 | Lecture |
| Session 4: Complexity | 03/02/2026 | Lecture |
| Session 5: Sorting algorithms | 10/02/2026 | Lecture |
| Session 6: Graphs and Search | 17/02/2026 | Lecture |
| Session 7: Dijkstra's Algorithm | 24/02/2026 | Lecture |
| Session 8: Greedy Algorithms | 03/03/2026 | Lecture |
| Session 9: Generative AI | 10/03/2026 | Lecture |
| Session 10: Mock Exam | 17/03/2026 | Review |

## Assessment

### Exam (60%)

The exam will take place at the beginning of the summer term (date to be confirmed by the programme team). It will cover both theoretical questions and ask you to write out small segments of Python code on paper as if you were going to run them on a computer. A mock exam will be made available toward the end of the module.

### Session hand-ins (20%)

Sessions indicated below have a small graded component. They are typically released the day of the session and due the next week. In the event that you need help on these tutorials, there will be time during the tutorial to ask for it. This should be the easiest 20% of the grade. We will give you plenty of advanced warning on which tutorials need to be handed in and when they need to be handed in by.

### Homework assignments (20%)

There will be two individual homework assignments intended to recap, explore and combine ideas from the course materials.

All assessment dates will be made clear well in advance, in coordination with the programme team.

### Assessment marking

After certain sessions, you will submit a small number of exercises, which will be marked for correctness, commenting, and efficiency. That is, you will typically implement Python functions, and we will check whether they work correctly with different inputs.

### Collaboration guidelines

We strongly encourage you to work together and discuss the course materials and activities with your classmates. In our experience, collaborating with classmates is both a great way to learn and much more fun than working alone.

However, the limit of this cooperation is sharing code on the individual homework assignments. You may discuss ideas and approaches for the homework, but **you should write your own code.** In brief, you should NOT:

- Copy code from your colleagues, or let others see your code
- Look for solutions online, or post your solutions online
- Use ChatGPT or similar large language models to write code except where explicitly stated otherwise

The College takes plagiarism very seriously and it can result in severe penalties. The College's Academic Integrity policy can be found here. Please note that the autograding software we use can run plagiarism checks against both other submissions and online sources.