Welcome to Python Programming!

Dr Charlotte Desvages - 2254 JCMB

charlotte.desvages@ed.ac.uk

Course summary

- First: **programming fundamentals** using **Python**.
- Then, practical applications in **data manipulation and visualisation**.

No previous programming experience is required!

You will learn to use **professional programming workflows and tools** (git + GitHub, VSCode in particular), and you will have the opportunity to **collaborate** with peers to develop your skills.

A typical week

- In your own time: watch 1-2 short **videos**, and complete a **Python notebook** with interactive examples and exercises.
- **Lecture**: Mondays 1.10-2pm. Code-along examples reviewing the previous week. Bring your laptop!
- **Workshop**: Thursdays or Fridays (depending on your group). Work on a programming task in groups of 2-3.
- Short **weekly homework**: either quiz (autograded), or peer-assessed code review.

Assessment

- **4 quizzes**: best 3 out of 4 count for a total of 15% (5% per quiz).
- **4 code reviews**: best 3 out of 4 count for a total of 5% (1.67% per code review).
- **Project 1**: done individually, halfway through the semester, counts for 40%.
- **Project 2**: done in small teams during workshops starting Week 8, counts for 40%. Show off your Python skills by extracting and visualising some interesting information from a given dataset.

Tell me about you!

app.wooclap.com/XFBGBA

Figure 1: QR code for Wooclap

Some advice

There is no exam – but **you should still study**!

- Make yourself a notebook with your own notes; do the practice exercises; revise previous weeks; quiz yourself; etc.
- Prepare for the assignments like you would prepare for an open-book exam.

Some advice

- Your **workload will get high** in the second half of the semester. Making sure you're well-prepared for the assignments means that you will spend **a lot less time** on them.
- This will also help you **develop and retain** your programming skills (and all the other skills you will practice!) much more efficiently.

Some advice for non-native English speakers

Don't translate your course materials, and practice your English as often as you possibly can!

You will improve a lot in very little time. Only use translation tools to look up individual words if you need to (and make notes!).

Generative AI

e.g. ChatGPT, Claude Code, Gemini, etc. First of all:

- Inform yourself about the ethical and energy costs of training and using LLMs.
- Inform yourself about how they work – why they don't "reason", despite claims to the contrary.

Hicks, M.T., Humphries, J. & Slater, J. **ChatGPT is bullshit**. *Ethics Inf Technol* 26, 38 (2024). https://doi.org/10.1007/s10676-024-09775-5

Read the University Guidance for Students on using generative AI

Generative AI

We will **not** be using generative AI at any point in this course. Any use of generative AI for assessment will be considered **academic misconduct**, as per the University disciplinary policies. But as a teacher, this is not what I really care about:

- You are here to **learn** and develop in-depth academic skills which will serve you well beyond programming.
- As explained in the University guidance, there is more and more research coming out to show that relying on generative AI as a student is likely to damage your learning.
- Think about it as **going to the gym**. If you want to get stronger, you need to work out! You have made a huge personal investment to come study here, and have the time, the space, and the support to grow academically and become more independent. You are here to get stuck on challenging problems – and to learn how to get unstuck!

Generative AI

- If you want to use generative AI later in your professional life, this is your choice – and it will not be difficult to learn. When ChatGPT first came out, programmers had to learn all by themselves – and they didn't need a university course!
- GenAI tools *can* be useful for certain tasks in programming work, and in some cases they *can* be useful to help with learning, but only:
  1. ***if you know how to ask the right questions***, and
  2. ***if you know how to critically evaluate the output***.
- Learning Python "the old-fashioned way" is how you get these skills.

Questions?