Introduction

Programming Practice and Applications (4CCS1PPA)

Dr. Martin Chapman
Thursday 29th September

programming@kcl.ac.uk
martinchapman.co.uk/teaching

Q: What does the term `programming' mean to you?

Let’s Talk About A Difficult Relationship…

IN TRADITIONAL HIGHER EDUCATION...

You are expected to attend (long) lectures and **listen** to **theoretical** content presented to you by someone **speaking**.

Teaching is typically done using **metaphors** and **abstract examples**.

The lecturer is perceived to be at the top of a **knowledge hierarchy**. You only learn from them.

A module follows a **set path**, which is pursued by everyone.

Coursework is set centrally, and completed according to a **particular set of goals**.

HOWEVER, WHEN PROGRAMMING...

In order to learn how to program you must **do** some programming. It is **not sufficient to just listen to someone talking about it**.

Often, programming concepts can only be understood by seeing them **used**.

Age and seniority are **no guarantee of experience**.

Everyone learns how to program in **different ways**.

You can only really **learn** and **enjoy** programming, if you are prepared to **set your own goals**.

The Solution?

THE SOLUTION: RECENT CHANGES

Fortunately, we've known about these problems for a while.

PPA is a **newly structured module** where:

- **50%** of your grade is now from practical tasks (previously it was much less).
- Regular assignments **keep you programming**.
- More **laboratory** time.
- There is **no January exam**, to accommodate different **learning rates**, but a **class test** (**10%**) to check where you are up to.

THE SOLUTION: LECTURE FORMAT

From **next week**, aim to minimise the time I talk, and maximise the time **you** program.

- At times more like a **workshop** than a lecture, with the **tutorial time merged into the lecture**.
- Bring a **computer**, a device upon which you can write code, or a pen and paper.
- Because of this format, the **utility of lecture capture is diminished**.

When I do talk, I will aim to focus on **practical examples**.

Aim to breaker the session down and have regular **breaks**.

- Roughly in accordance with attention span (**45 minutes**).

THE SOLUTION: LECTURE FORMAT (FEEDBACK)

But we will, necessarily, still retain some elements of the **traditional** lecture format.

- I do not mind what you do while I am speaking (e.g. independent work), **but do not disturb those who choose to listen**.
- Push yourself to **listen** and **engage** right through to the end.

Tell me what works and what doesn't; **the structure of these three hours is up to you**.

Using our e-learning platform **KEATS**, you will be able to **give feedback after every lecture**.

THE SOLUTION: LABORATORIES

You will be assigned to a laboratory session (from **next** week; details on your timetable), which will provide a much needed **additional** environment in which you can **do some programming**.

Even if you do not wish to attend the lectures, you must attend the laboratory sessions (if nothing else, we check your attendance).

- Although the labs are designed to complement the lecture content as you **complete questions** and **implement code** from the slides.
- Also an opportunity to work on **coursework** (more later).

Run by bright, helpful teaching assistants.

- When I was a Teaching Assistant, I was told I was more helpful than the module lecturer. Whether this is true or not, it shows that I was able to provide a **fresh perspective** on the material.

THE SOLUTION: THE PEOPLE AROUND YOU (1)

Even with 15 years of programming experience (8 in Java) and experience in a range of different programming languages, **I will still not be the best programmer in this room**.

- Every year students come to us from a range of **impressive backgrounds**.
- Many of you will be familiar with languages that employ conventions that I am not (overly) familiar with, such as **memory management.**
  - These things will often not be covered in this module, but are **relevant** and **interesting**.

THE SOLUTION: THE PEOPLE AROUND YOU (2)

We can **capitalise** on this.

- Learn from **each other**, not just from me.
- I can also **learn from you.**
    - If I don’t know the answer to a question, I won’t pretend, I’ll cover it in the next lecture (if relevant), or **try it out** with you.

This dynamic is a **good thing;** a phenomenon **unique to our discipline**.

THE SOLUTION: THE PEOPLE AROUND YOU (3)

Given this dynamic, it might be useful to view me as a **coach** rather than a teacher, who:

- Nurtures **existing** skills.
- Prioritises **you** in the learning process.
- Allows you to **practice** in a structured environment
- Encourages you to **challenge your own knowledge**.

THE FINAL SOLUTION: YOU

We have setup the course to try and help you as much as possible, but in the end **you must take responsibility for your own learning**.

Learning how to **approach topics yourself** is a skill that is just as important as any topic in Computer Science, going forward.

- If the order of topics we have chosen doesn't suit your learning style, use **other resources** to approach the topics in a different way (see the KEATS support procedure).
- Unsure about something? **Try it out**.

There is no expectation that you will understand something the **first time** it is presented to you; **go over concepts again in your own time**.

Seek **support** as needed (again, see the KEATS support procedure).

Set yourself goals: **`today I want to build X'**.

SUMMARY: WHAT THIS DIFFICULT RELATIONSHIP TELLS US

Programming is a very **unique** discipline, and you have made a **good choice to study it.** The most **creative** thing you can do while remaining **scientific.**

I am **not your teacher**, you are a **group of adults** learning together with me.

- Learn from **each other.**
- I can learn from **you.**
- Be **responsible** for each other.

University is very different from College or 6th Form.

- Please don't call me **`sir'**.

Ultimately, I cannot teach you programming, I can only enable you to teach yourself.

To Summarise And Formalise...

THE MODULE

4CCS1PPA (Programming Practice and Applications)

2 Semester, 30 credit module

- Coursework (**50%**)
- Class Test (**10%**)
- Exam (**40%**)

Lectures every Thursday, 3pm - 6pm, **except this week**, when we will **finish at 5.00pm and resume for the last hour tomorrow at 2pm in the same place.**

The KEATS page is the hub for information about this module. Search `4CCS1PPA' on keats.kcl.ac.uk and enrol yourself.

THE CONTENT (1)

Semesters 1 & 2

Java (C++ in second year). General problem solving.

Semester 1 (`Practice' and `Practise')

Object-Oriented design (including combining objects), program control flow, library classes, arrays, data structures and errors.

Semester 2 (`Applications')

Graphical User Interfaces, Human Computer Interaction, regular expressions and software development tools.

https://www.kcl.ac.uk/nms/depts/informatics/study/current/handbook/progs/modules/4CCS1PPA.aspx

THE CONTENT (2)

Who is PPA for?

Those with no programming experience. Yes. The majority of this course will be aimed at you.

Those with some programming experience. Towards the beginning of the course, yes, we will explore **object-oriented programming** which is typically unfamiliar to students. Other topics will be trivial.

Those who have been working for Google for 15 years. Not so much, but we will regularly post `challenging' problems to a forum on KEATS. **Attend APT**.

Remember: Challenge your own knowledge.

- Prior experience is **good**, but is often just the **start**.

THE COURSEWORK: OVERVIEW

Semester 1 (25%)

10 short coursework exercises, each worth **2.5%** of your final grade.

Semester 2 (25%)

4 short coursework exercises, each worth **2.5%** of your final grade. **1** major piece of coursework, to be completed in groups, worth **15%** of your final grade.

THE COURSEWORK: DEADLINES

Semester 1

One piece of coursework will be released every week, on the **Monday following the lecture**, and will be due the **Monday** after.

- The first piece of coursework will be released **on Monday.**
- However, it's only likely that you will be able to complete it until **after** next week's lecture, so you will have just over **two weeks** to complete it.
- The remaining pieces of work must be completed within **one week** to ensure you are **practicing programming regularly**.

Exact deadlines are given on KEATS.

Semester 2

Pending.

THE COURSEWORK: CONTENT

Semester 1

Assignments are designed to **test your understanding of the most recent lecture**(s), and can be completed in the labs, with the assistance of the TAs.

Semester 2

We will continue to test your knowledge incrementally, and then **bring the learning of both semesters together** in the major piece of coursework.

- This piece of coursework will also enable you to explore a wider range of **software engineering tools** including **version control**.

THE COURSEWORK: MARKING AND FEEDBACK (NEXUS)

Semesters 1 and 2

We are going to ask you to submit your code using a specialist piece of software called **Nexus**.

- There will be a link to the Nexus submission page for each assignment on KEATS (via King's **Github**). **Guide on KEATS**.
- Nexus will allow you to submit your code multiple times, and after each submission will **check** that your code **compiles on *our* system** (so that if it doesn't you can fix it!), and may provide you with **hints about formatting**.
- However Nexus will **not** provide you with a mark.
- Don't keep submitting your work **after** the deadline. This is for **late submissions only**.
- Nexus is **experimental**. Give us **feedback**.

THE COURSEWORK: MARKING AND FEEDBACK (MARKING)

Semesters 1 and 2

After the deadline, we will take the last piece of code you submitted through Nexus, and mark it **offline** using **another** piece of software that will **recommend** a mark to us.

- We will share the output of this software with you, as **quantitative feedback**, before the deadline of the next assignment, but any mark given is **provisional**.
- Don't upset this software! If it cannot compile your code, you will receive a mark of **zero**.
  - Code that **works** is better than code that is **finished**.
  - Check carefully before submitting, and using Nexus, that your code can be compiled.
- Still, this mark is only a **recommendation**, so we will use our discretion, and formalise your marks at the **end of the semester**.

THE COURSEWORK: MARKING AND FEEDBACK (CODE DESCRIPTION)

Semesters 1 and 2

We will ask you to submit some **documentation** of your code along with your submission, so that we know you understand what your code does.

- We will check this documentation at the **end** of the semester and use its **quality** to turn your provision mark into a **final** mark, for each assignment.
- Without submitting documentation along with your code, you will receive a mark of **zero** for that assignment.
- A sample piece of documentation will be **posted on KEATS** to guide you.

You must also comment your code, and the quality of your comments will also be used to determine your final mark.

Any code or documentation that is found to be similar will result in a harsh penalty (more later).

We aim to trial a more **interactive version** of the Nexus submission and marking tool in Semester 2.

THE COURSEWORK: MARKING AND FEEDBACK (TA FEEDBACK)

Semesters 1 and 2

During the laboratory sessions, while you are working on a piece of coursework (e.g. CW2), the TAs will chat to you about the work you just submitted (e.g. CW1).

- This discussion does not gain you any marks but is your **main opportunity** to receive **qualitative** (e.g. code structure, style and efficiency) feedback on your work.
- **This discussion is much more important than your mark itself**.

THE EXAM

Class Test (10%) on Wednesday 14th December.

- Designed to informally check your progress.

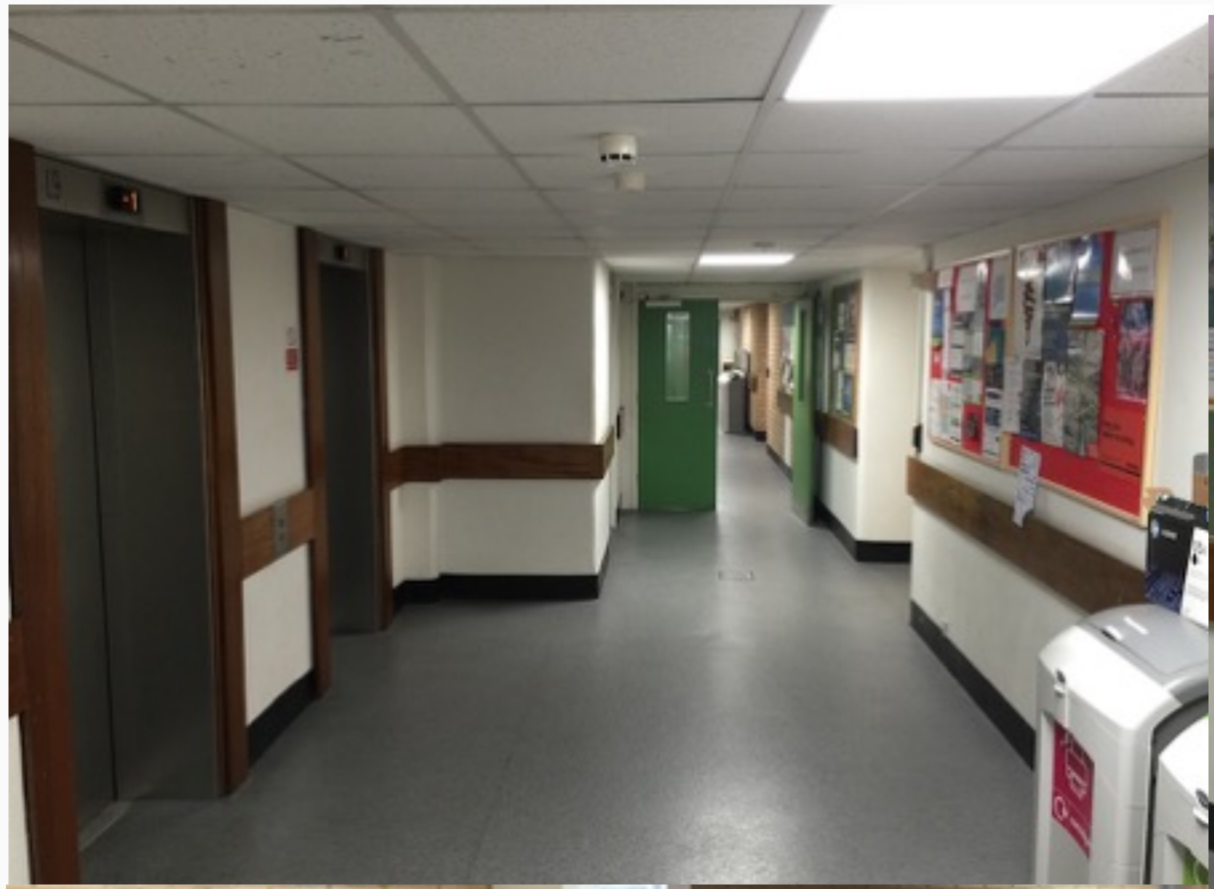
Exam (40%) in May 2017.

- Designed to test your programming abilities after the whole year.

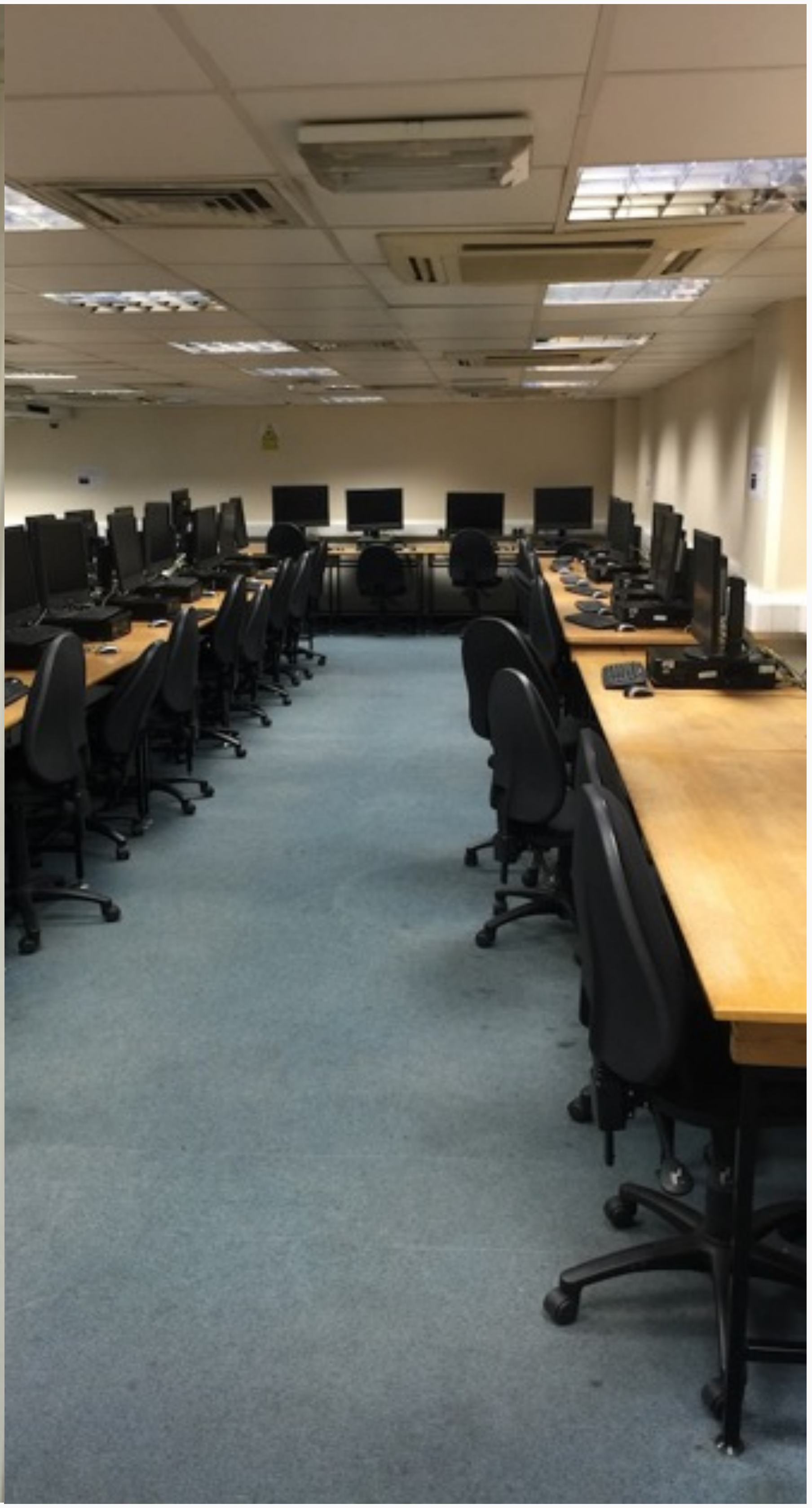
Both are multiple choice, but **do not underestimate this format**.

You must achieve at least 40% (except in exceptional circumstances) in both the examination and the coursework (which includes the class test) components in order to pass the module.

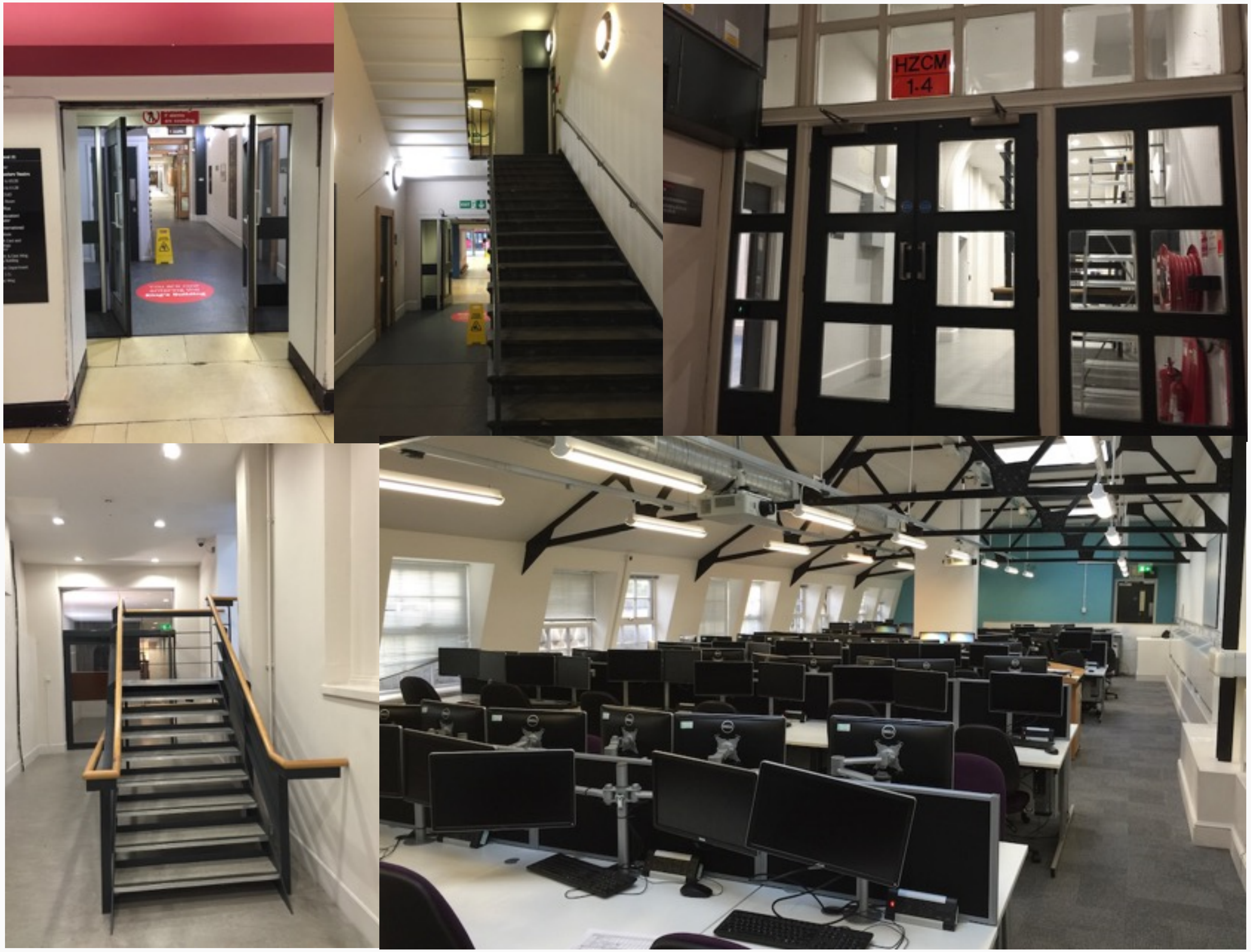
THE PLACES: S5.33 (5TH FLOOR, STRAND BUILDING)



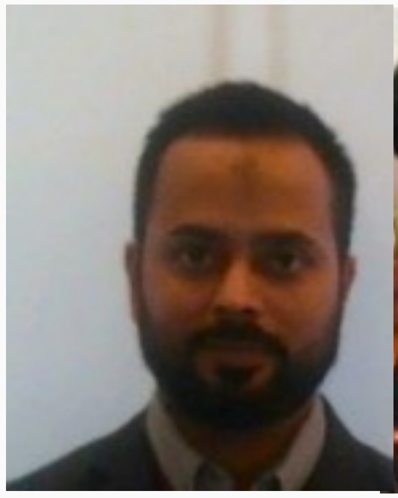
THE PLACES: S4.01 (4TH FLOOR, STRAND BUILDING)

S4.01 Computer Lab
Department of Informatics
Faculty of Natural and Mathematical Sciences
Informatics Computing Labs
Faculty of Natural and Mathematical Sciences
Informatics Computing Lab S4.01


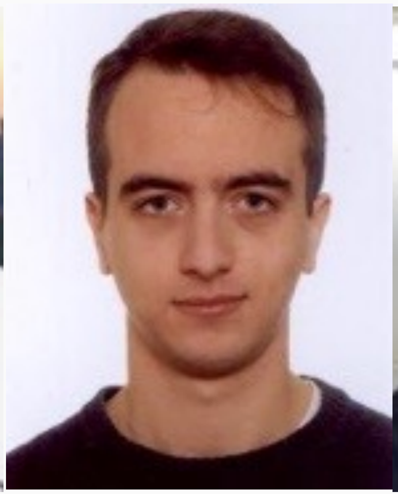
THE PLACES: K4U.13/14 (4TH FLOOR, KING'S BUILDING)

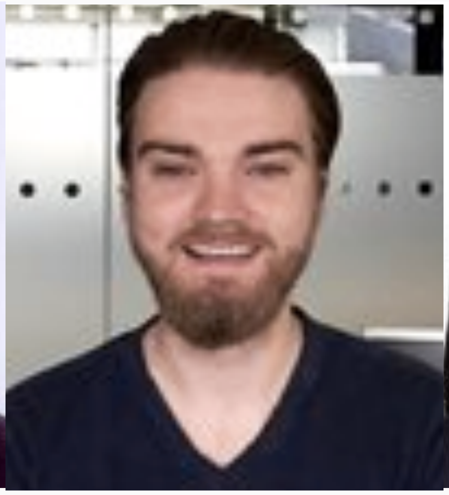


THE PEOPLE


Syed

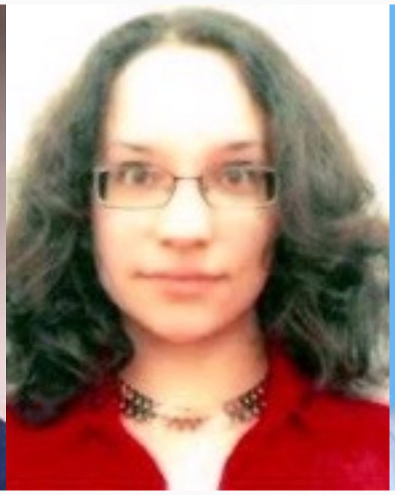
Asad


Kostas


Martin

Kelly

Lukas


Liana

Cheng

Sundararam

Diego

Shanti

Tomas

Steffen

SUPPORT

As mentioned, you will find a detailed support procedure on **KEATS**.

Books from the reading list are a good source of practical exercises, but for support for theoretical topics, in all honesty, **Google** is more efficient.

Email queries, if necessary, should be sent to **programming@kcl.ac.uk**. Our personal email addresses are for **administrative issues only**.

Come and see me during my office hours (generally on Fridays; booking link and exact times on KEATS); I want to meet every single one of you and I want **every single one of you to make a good start at learning how to program**.

- My office hours start **next week** (3/10), because of tomorrow's **one-off lecture**.

Introduction

Programming Practice and Applications (4CCS1PPA)

Dr. Martin Chapman
Thursday 29th September

programming@kcl.ac.uk
martinchapman.co.uk/teaching

These slides will be available on KEATS, but will be subject to ongoing amendments. Therefore, please always download a new version of these slides when approaching an assessed piece of work, or when preparing for a written assessment.