GCSE Computer Science

Classroom Best Practice and Methodology Network

Pearson Edexcel

Introduction

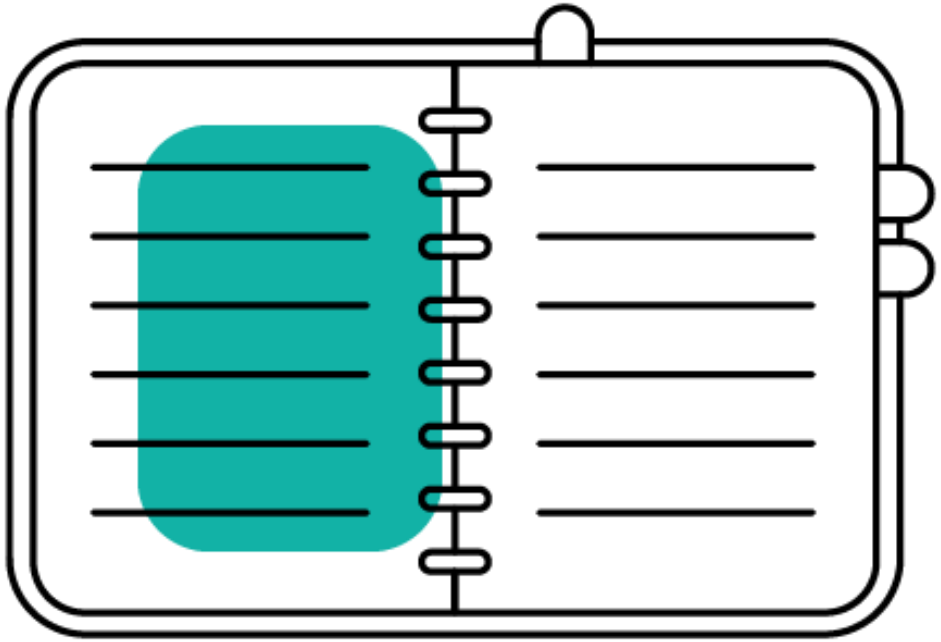
Welcome

Poll

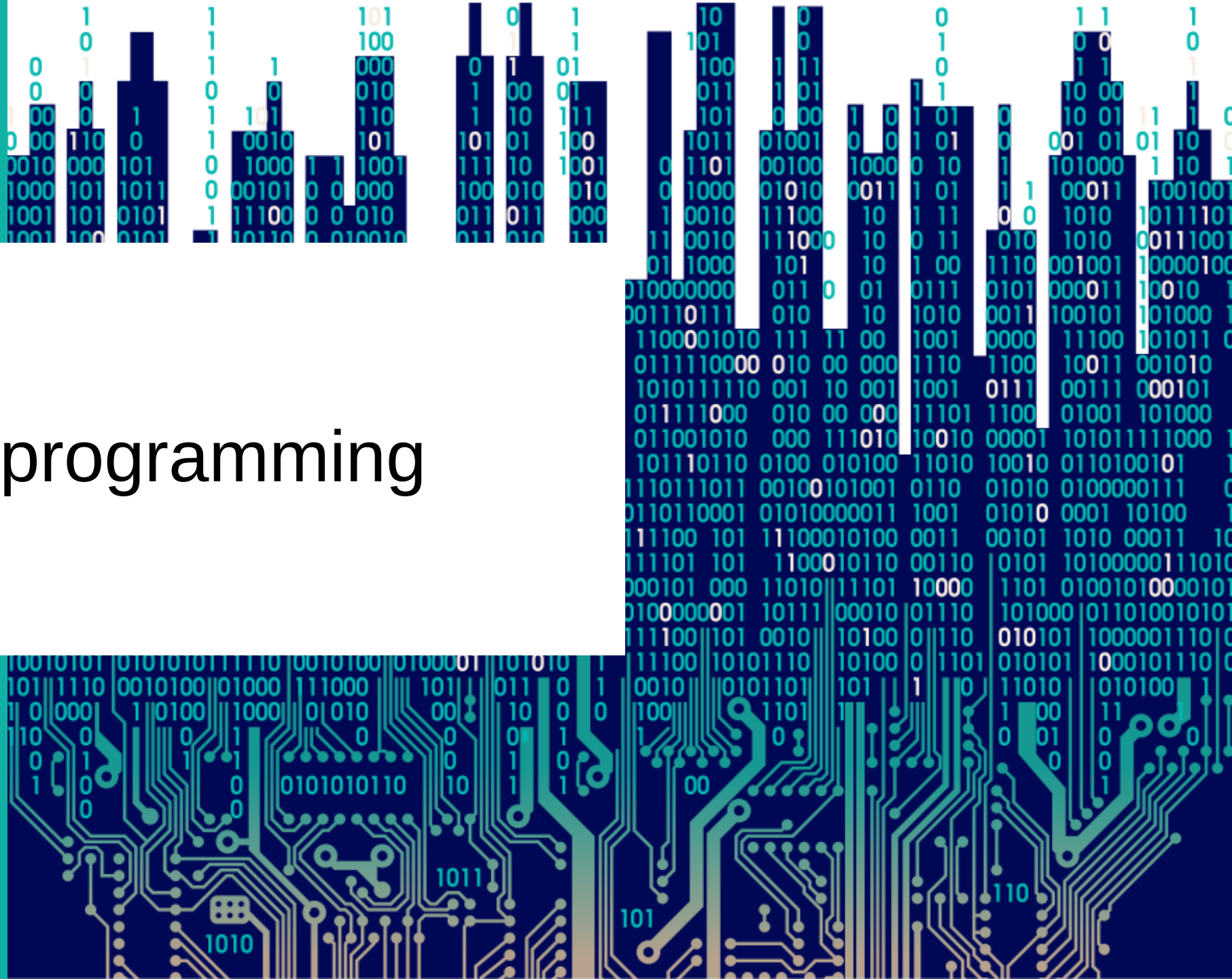
How long have you been delivering this course?

Agenda

- Introduction
- Teaching programming
- Discussion
- Teaching theory
- Discussion
- Support and resources



Teaching programming



Pearson
Edexcel

Integrated Development Environments

Poll

Which IDE do you use?

Poll

Do you offer your students a choice of IDE?

Practical hints for running the programming exam

- Choose an IDE carefully.
- Have your students practise in advance with the exam setup.
- Make sure students can turn on autocomplete, line numbers and other IDE settings that they're used to, as well as showing file extensions in the file manager.
- Check with exams team they are prepared; Train an invigilator or two specifically; Arrange for an IT technician to be available.
- On the day:
  - give students two copies of the code files
  - PLS paper copy is in secure QP package; pdf to be given with code files.
- After the exam:
  - get your IT technician to zip *everything* when uploading students' work.

Preparing students for the programming paper

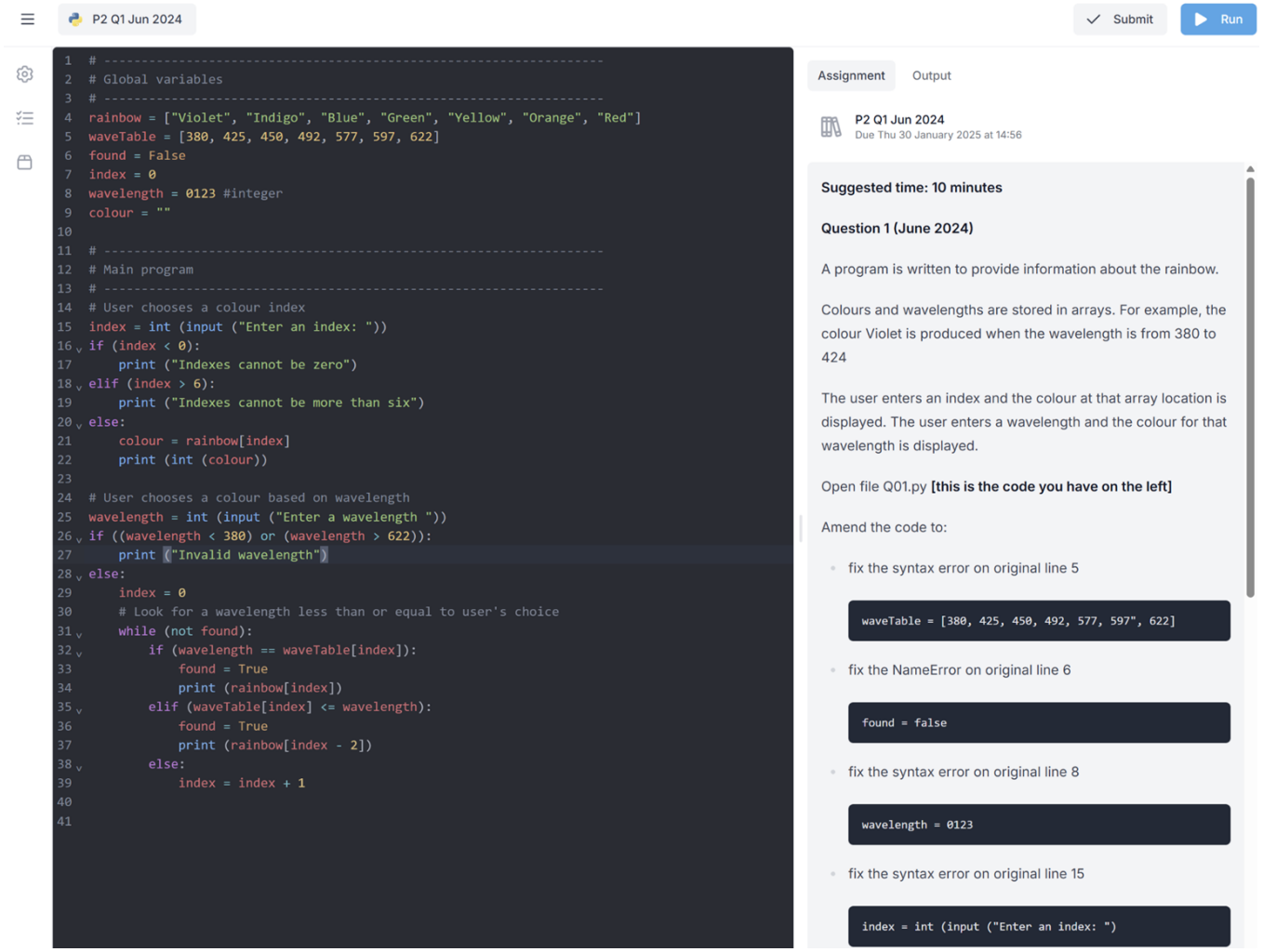
- Use the PLS in class… make using it the “normal way of working”
- Student: “What happens if..?”
- Use the sample questions (e.g. Edublocks)
- Shed Loads of Practice (SLOP)
- Actual papers vs Sample papers

Edublocks

- Not just blocks!
- 100% free and accessible
- 24 / 7 / 365
- Safe and suitable
- Easy to use
- Responsive to feedback / feature requests
- Import libraries
- Exam mode in development
- Engaging environment

Live demo: projects, assignments, starter projects, scaffolding with GPT

Please note: *Edublocks is among various available EDIs, and its usage is not mandatory for delivering the Pearson Edexcel qualification.*



PLS Quizzes

How about a pop quiz to get students used to using the PLS? It helps reinforce key terminology, familiarises them with what's covered in the PLS and gets them used to the <notation> for arguments. E.g.

Q1: Give four examples of relational operators. (4)

Q2: Which kind of loop is used for iteration? (1)

Q3: What is the (official) name of the % operator in Python? (1)

Q4: What is the difference between a procedure and a function? (1)

Q5: What two parameters are needed to open a file for appending? (2)

Q6: What does ord(<char>) do? (1)

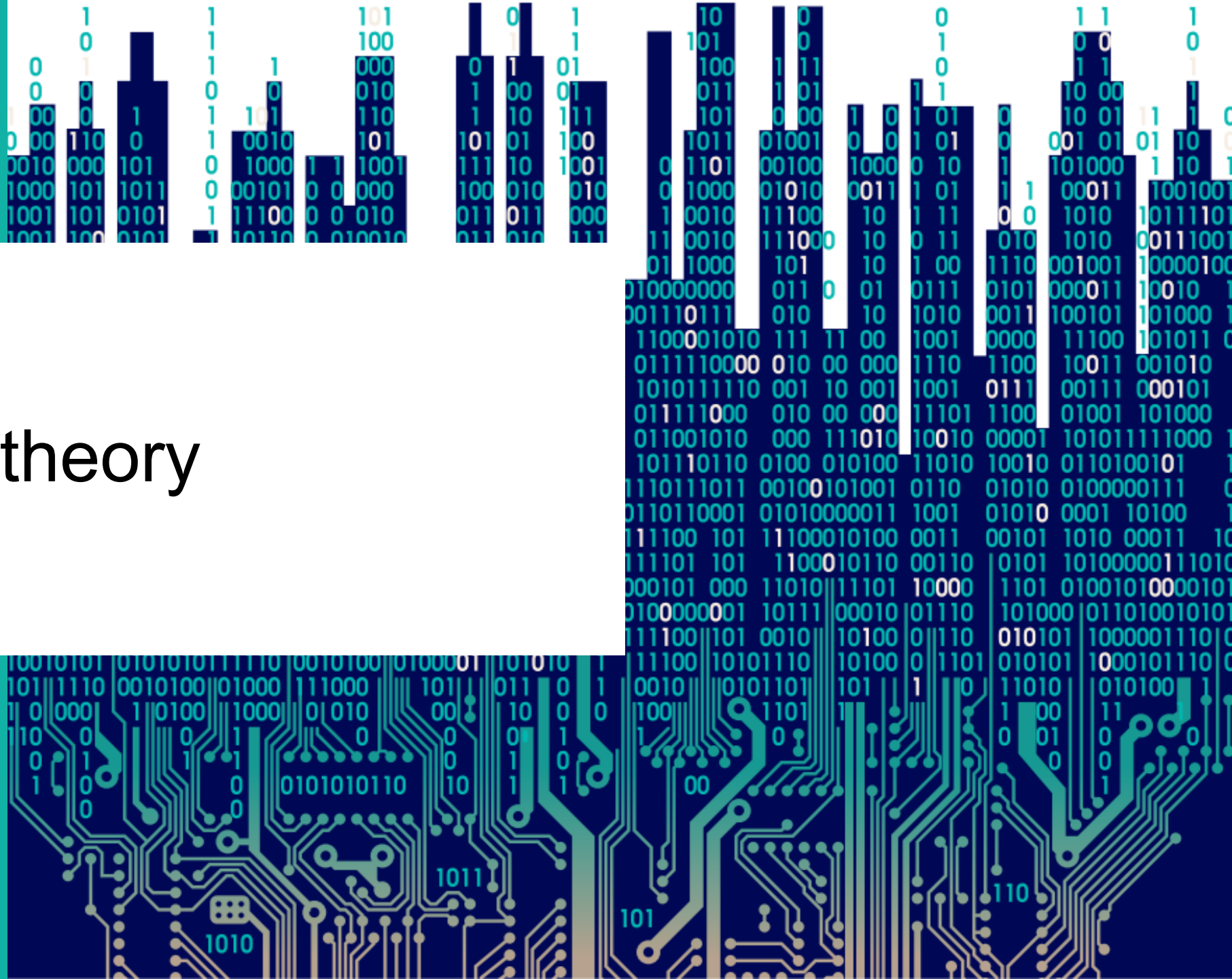
Discussion

Paper 2: Programming (Application of Computational Thinking)

Discussion Focus: Delivering Programming Confidently

- What approach do you take to introduce programming: PRIMM, TIME, Use-Modify-Create, or something else? Why?
- How do you scaffold programming tasks for mixed-ability learners?
- Which concepts (e.g. iteration, logic, subroutines) do students find most difficult? What helps?
- How do you prepare students for the on-paper code questions, especially those who struggle with syntax?
- How do you assess and give feedback on programming skills?
- What’s your favourite coding task or project that builds strong skills while keeping students engaged?

Teaching theory



Pearson
Edexcel

Background in theory

Poll

Which theory topic do you find hardest to deliver?

Poll

How well do you feel KS3 prepares students for this course?

Practical ideas for teaching theory topics

- Analogies
- SLOP
- Exam Questions: Topic booklets
- Reinforce knowledge
  - Ebbinghaus’ forgetting curve
  - Low stakes, retrieval
  - Interleaving
- 'What’s in the bag?’
- Binary bingo
- CPU activity

Practical ideas for teaching theory topics

🔐 Cybersecurity

Analogy: Home Security

- **Firewall** = Security guard checking who can enter
- **Antivirus** = Pest controller checking for infestations
- **Phishing** = Scam phone call pretending to be your bank
- **Passwords** = Keys – strong keys (long passwords) make break-ins harder

💾 Memory & Storage

Analogy: Desk and Filing Cabinet

- **RAM** = Desk space – fast, temporary access while working
- **Hard Drive/SSD** = Filing cabinet – permanent but slower
- **Cache** = Sticky note on your monitor – very fast and small
- **Virtual Memory** = Borrowing space on the floor when your desk is full

Practical activity for teaching scheduling algorithms

🛒 **FIFO (First-In, First-Out)**

Analogy: Supermarket Checkout Queue

Customers are served in the exact order they arrive. No skipping!

✅ *Simple and fair, but slow if someone has a large basket.*

✂️ **Shortest Job First (SJF)**

Analogy: Quick Haircuts at the Barber

The barber picks the person with the **shortest haircut time** first.

✅ *Efficient overall, but longer jobs may have to wait (unfair!).*

🕹️ **Round Robin**

Analogy: Friends Taking Turns on an Arcade Machine

Everyone gets **2 minutes to play**, then swaps out and rejoins the queue.

✅ *Fair for everyone, but lots of switching between players.*

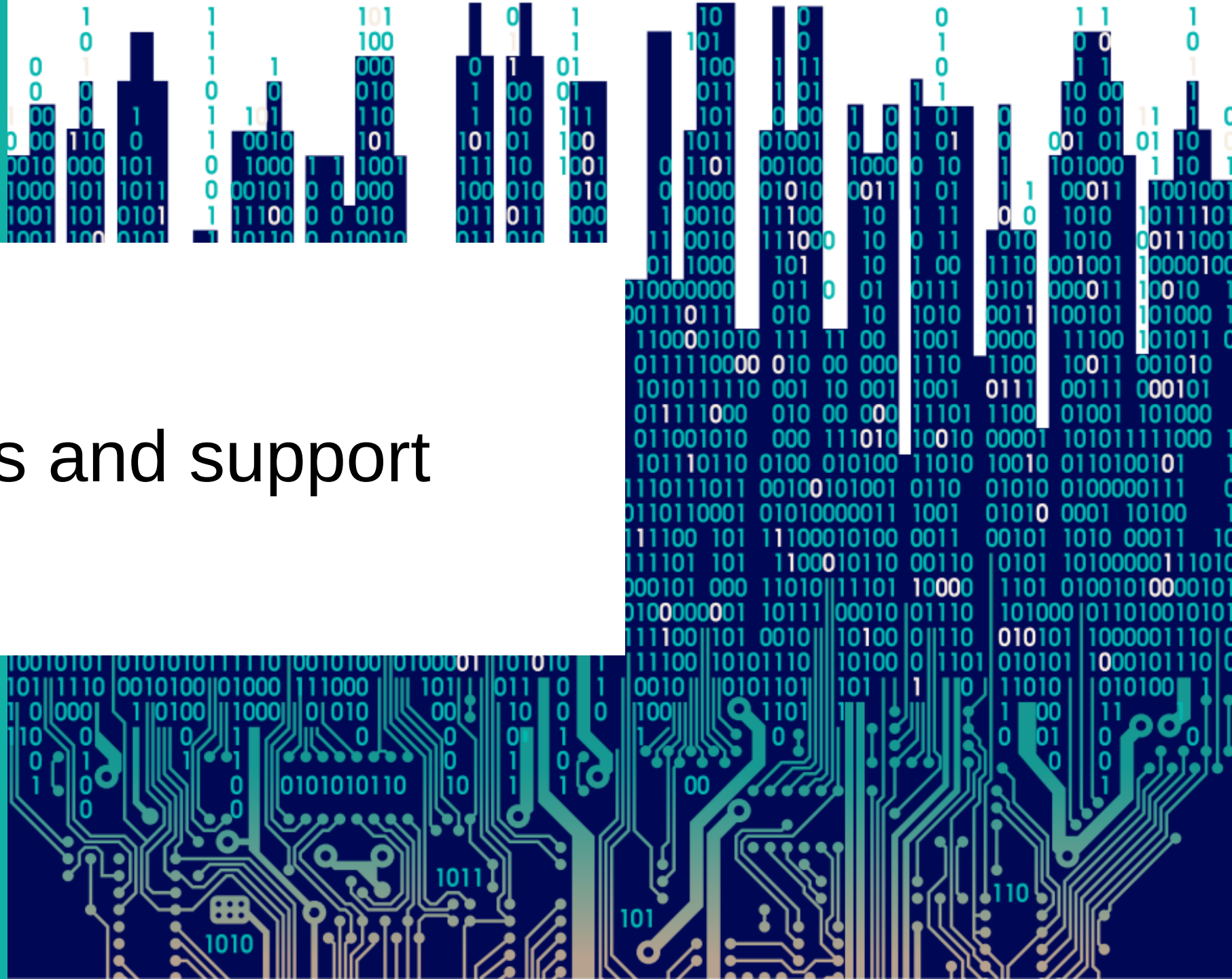
Discussion

Paper 1: Theory (Principles of Computer Science)

Discussion Focus: Understanding and Teaching Theory Concepts

- Which topics do your students find the hardest to understand? Why?
- What analogies or practical examples have helped your students 'get' abstract theory topics?
- Which theory areas lend themselves best to ‘use-modify-create’ or 'misconception spotting' activities
- How confident are students in applying theoretical knowledge to exam scenarios?
- Do you relate programming tasks to the theory topics?

Resources and support

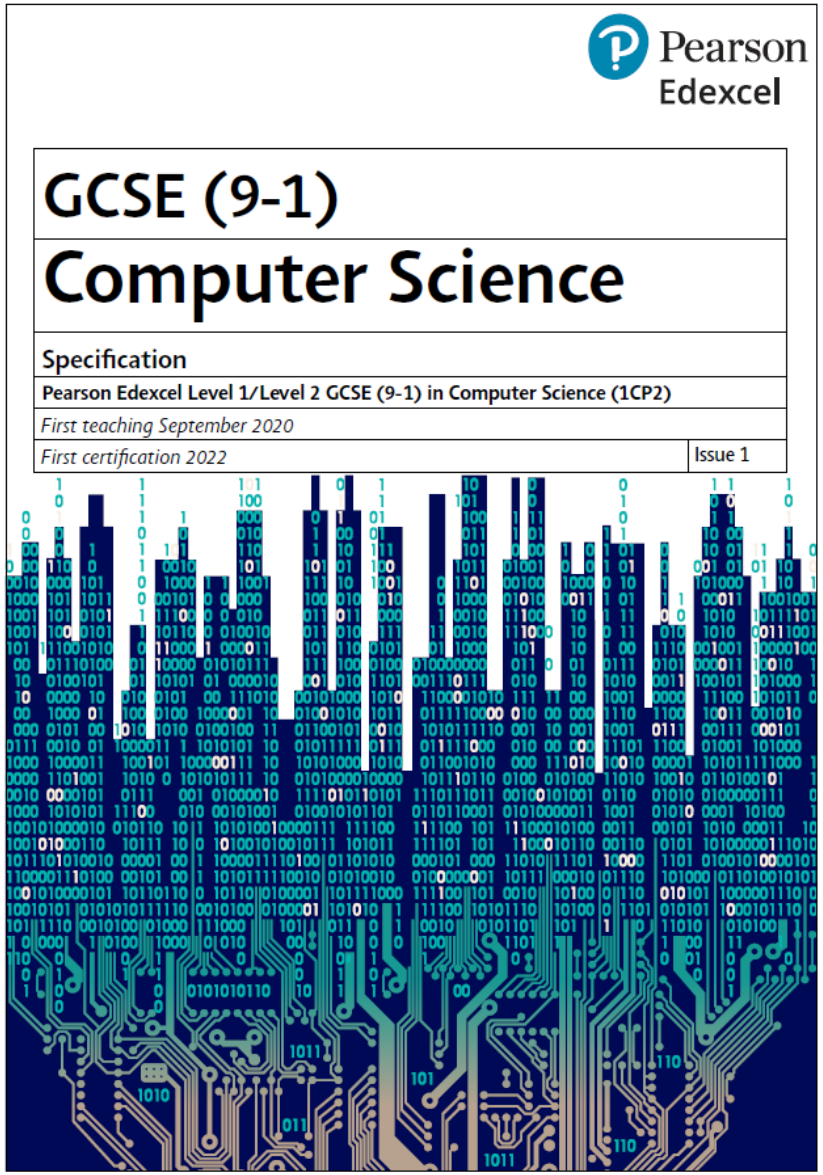


Pearson Edexcel

Resources for teaching programming

Tailored to this particular course:

- Pearson Edexcel's scheme of work
  (fully-resourced lesson plans provided)



Resources for teaching programming

The following links are examples of cloud-hosting IDEs:

- Edublocks (including coding resources to go with above)
- Trinket.io (including coding resources to go with above)

Resources for further programming practice

Online courses:

- W3schools
- Raspberry Pi Foundation
- Coddy.tech
- Sololearn
- Mimo
- Snakify
- Time2Code (free from Craig'n'Dave)
- PythonSponge (free from the team behind PCTC)

An actual textbook:

- "Python by example" (book by Nichola Lacey ISB 978-1-108-71683-3)

Resources for further programming practice

A competitive element:

- Bebras Coding Challenge (competition)
- Perse Coding Team Challenge (competition)
- CodinGame (practice and also competitions)

Resources for teaching theory

- Pearson Edexcel's fully-resourced scheme of work
- BBC Bitesize
- Isaac Computer Science
- Smart Revise, videos and other resources (Paid-for, from Craig'nDave)
- Ultimate GCSE Computer Science (Paid-for, from Paul Long)

General:

- Ada Computer Science
- TryHackMe
- Eedi / diagnostic questions

Resources for professional development

- Pearson Edexcel's free online webinars and meetings
- Isaac Computer Science offers free events for students that are also good for teachers
- Digit<all> Charity offers free events and CPD (as well as resources)
- Computing at School (Community and events)
- NCCE (CPD, curriculum, CQF)

Subject Advisor Support

Our subject advisors are experts in their fields and are here to support you throughout the year.

Computer Science

Email: TeachingComputerScience@pearson.com

Phone: +44 (0) 344 463 2535
(Mon–Fri, 9.00–17.00)

Book an appointment with your Subject Advisor

Sign up to receive regular updates from your Subject Advisor on qualification news and support for your subject.

Tim Brady
Computer Science and ICT

Pearson

Find out more

For more professional development courses please see Pearson's [Professional Development Academy]

Your Feedback Matters

Following this event, you will receive an invitation to share your thoughts about the session. Your feedback is invaluable to us, as it helps us tailor our professional development materials to better meet your needs. Please don’t hesitate to let us know what you’d like to see more of and what areas you think could be improved.

Pearson