

# Beyond Mnemonics

## Teaching for mastery through PCK – a GCSE Computer Science booster

*Starting at 14:10 in IT2*

Alan Harrison, author of  
“How to Teach Computer Science”

- Email [alan@https.online](mailto:alan@https.online)
- Website <https.online>, Twitter [@MrAHarrisonCS](https://twitter.com/MrAHarrisonCS)



# About me

---

- Professional Development Lead for the NCCE
- CAS master teacher & community leader
- Content author for Craig'n'Dave's SmartRevise platform
- Teacher and lecturer in Computing & CS.
- I wrote the books “How to Teach Computer Science” and “How to Learn Computer Science” in 2021 and 2022, see [https.online](https://www.mrharrisoncs.com/https.online) for more details.



# Introduction

---

## Beyond Mnemonics – teaching for mastery through PCK – a GCSE Computer Science booster

- Do you feel you are teaching for “surface learning”? Are you using tricks and schemes such as mnemonics to get them through the exams, and would rather teach for mastery but don’t know how? Alan’s book “How to Teach Computer Science” is all about the hinterland, the background knowledge that illuminates the subject and helps you teach it with confidence, and pedagogical content knowledge (PCK) – the “how to teach” knowledge that helps you succeed. Alan will explain why this “hinterland” is important and what PCK is and how to acquire it, and how to use both for mastery learning.

# Some Mnemonics and other Aide Memoires

I love mnemonics, here are some favourites.

Lemon  
Meringue  
Really Delicious

is

## BASIC LIFE SUPPORT



OSMOSIS.org

## In the Dining room We:

**R** Clean up after ourselves

**O** Use our time wisely

**A** Use manners

**R** Listen to adult

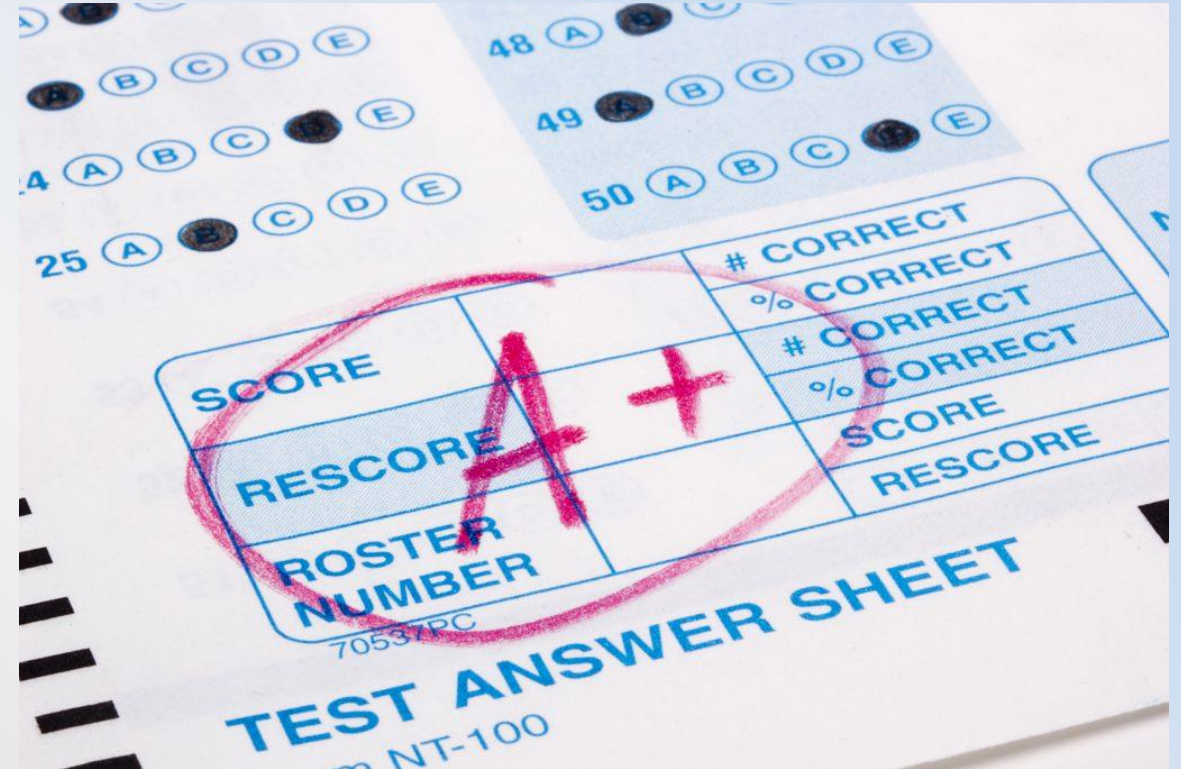


# When Acronyms Go Bad



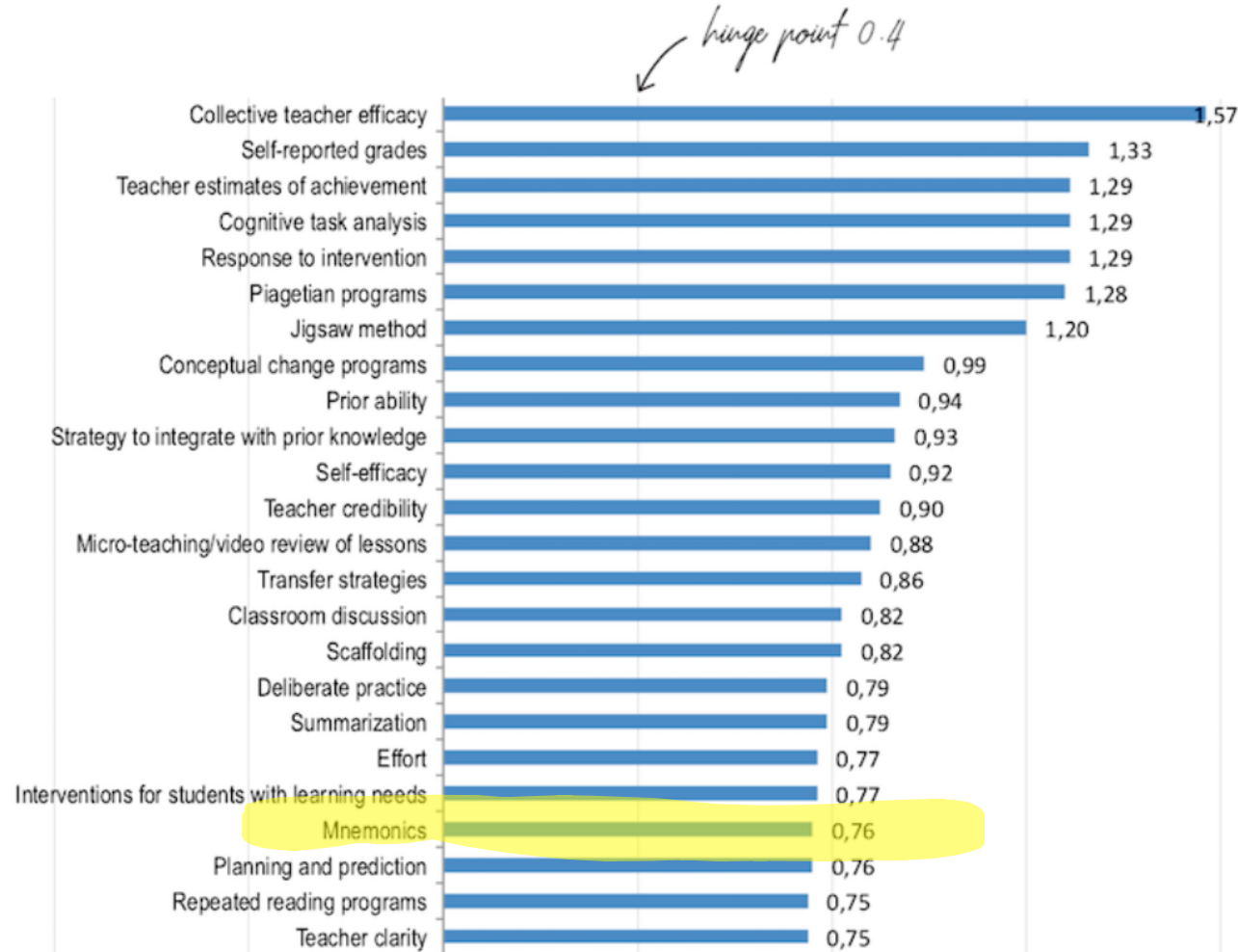
# So what's wrong with Mnemonics?

- A good **supplement** to a sound understanding
- Can be especially good for **ordered** lists e.g. the protocol stack  
(A Tree Is Large = Application, Transport, Internet, Link)
- Aids **short-term retention** which is useful to pass exams.
- But...



# Hattie Effect Size 0.76

Source: J. Hattie (December 2017) [visiblelearningplus.com](http://visiblelearningplus.com)  
Diagram: S. Waack (2018) [visible-learning.org](http://visible-learning.org)



## Mnemonics rank above

- Peer-tutoring
- Group learning
- Highlighting

## But below

- Deliberate practice
- Summarization
- Classroom Discussion.

<https://visible-learning.org/hattie-ranking-influences-effect-sizes-learning-achievement/>

# Before we go any further... your questions?

---

Scan QR code to open the Padlet.

Add a question you have about  
computing pedagogy

The link is...

[padlet.com/MrAHarrison/questions-em2ag3r4ga3u4plg](https://padlet.com/MrAHarrison/questions-em2ag3r4ga3u4plg)



# The case for mnemonics

“How exactly do they help us remember? They are simple; they chunk down complex, overloading or forgettable knowledge and they make it memorable, cheating the limitations of our working memories; they give us a way of self-checking that we have remembered all the content and in the right order.”

## Four Principles for Making Memorable Mnemonics

**1. Selection:** Select knowledge that lends itself to be turned into a mnemonic.

For example, **serial** information (like planets in order of proximity to the sun) especially lends itself to acronym mnemonics; so does **numbered** information (five pillars, seven continents, twelve nerves, etc)

**2. Cue:** Use the **first** ... as the ‘raw material’ for cues.

...orable **phrase** that reduces the

...ing the mnemonic to the  
...ent to the mnemonic.

Crucially, this works for **LISTS AND SEQUENCES** very well. But it can obfuscate more complex knowledge

[joe-kirby.com/2016/12/17/mnemonics-making-the-forgettable-memorable](http://joe-kirby.com/2016/12/17/mnemonics-making-the-forgettable-memorable)

# Surprisingly Low Utility

A study by Dunlosky et al (2013 including Daniel Willingham of “Why Don’t Students Like School” fame)...

Showed that mnemonics fared badly against techniques like retrieval practice, interleaving and spaced practice.



Commonly used techniques, such as underlining, rereading material, and using mnemonic devices, were found to be of **surprisingly low utility**. These techniques were difficult to implement properly and often resulted in inconsistent gains in student performance.

Other learning techniques such as .. practice tests and spreading study sessions out over time were found to be of high utility.

- Dunlosky et al 2013

[psychologicalscience.org/publications/journals/pspi/learning-techniques.html](https://psychologicalscience.org/publications/journals/pspi/learning-techniques.html)



# But whatever works, right?

---

Tricks like mnemonics also undermine key teaching strategies:

**Metacognition** aka “**learning to learn**” requires students to plan and evaluate their own learning. Teachers can help by modelling their own thought processes e.g. when live coding. Memory tricks “shortcut” around these processes.

*Metacognition can add **7 months progress over a year** (EEF [link](#))*

## Encouraging Mastery

Tricks say “this stuff is too hard for you to understand”. Mnemonics might thus be self-defeating, causing students to switch off from the subject.

## Avoiding Misconceptions

Mnemonics allow for no nuance or ambiguity. There are several protocol stack models, and all are valid, but A Tree Is Large fits only one., while “Laptops Like this Language” is actually not true, CPUs process LLL but laptops can run HLLs using many compilers or interpreters.

# What to do instead?

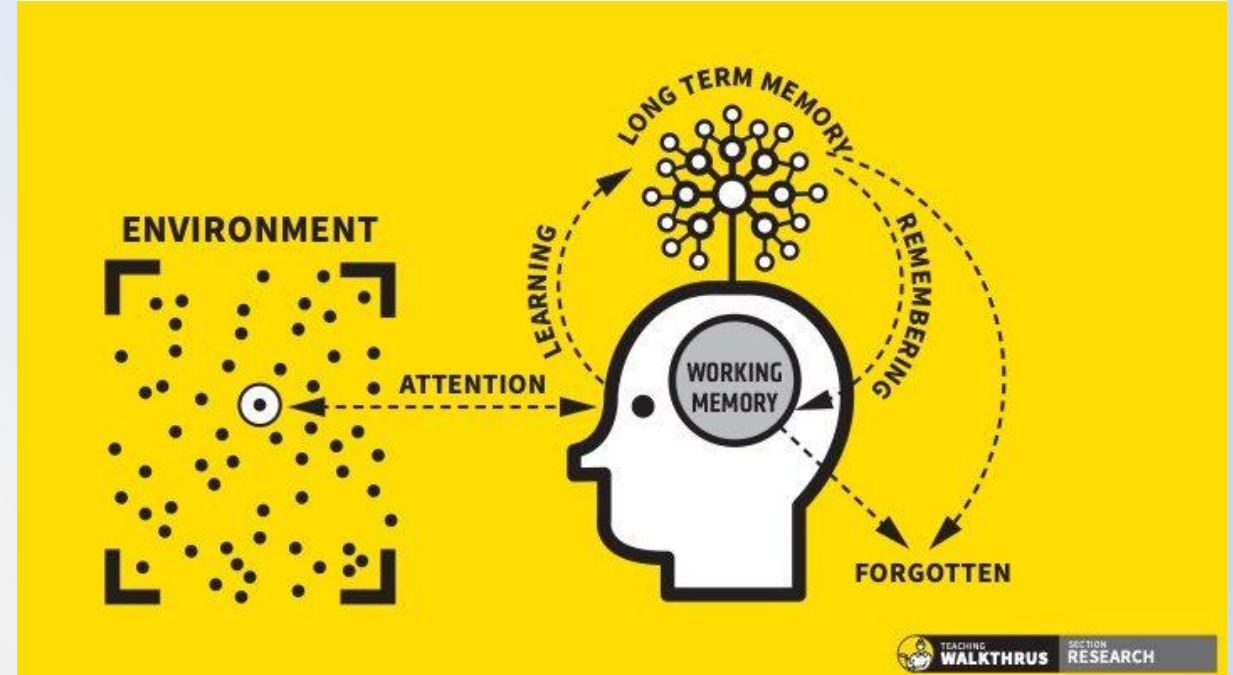
First, nothing beats good teaching.

***“memory is the residue of thought”  
and “understanding is remembering in disguise” – Willingham [link](#)***

Know your subject well, including the “Tier 3” vocabulary, links across topics and across the curriculum and the “Hinterland” so that your explanations are sound, rich and memorable.

Then teach it well, and teach for mastery – i.e. remembering things long term.

Wait, what’s the Hinterland?



# What is Hinterland?

The core is the central thread of the curriculum. Core is what must be retained to pass exams.

But the Hinterland:

- puts the core in context
- makes the subject memorable
- is a vital property that makes curriculum work *as narrative*

Tom Sherrington explains in a blog called “Signposting the hinterland: practical ways to enrich your core curriculum”. that curriculum can be divided into “core” and “hinterland”, where the hinterland is as important as the core and serves the purpose of:

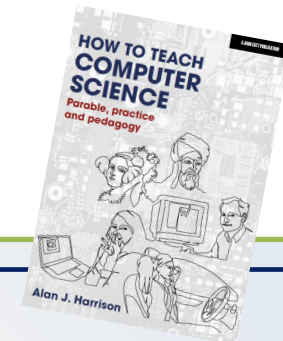
- *“Increasing depth: niche details about a particular area of study that deepen and enrich the core.*
- *Increasing breath: wider surveys across the domain of any curriculum area that help to locate any specific core element within a wider frame.”*

Sherrington quotes from an earlier blog post by Christine Counsell that explains why the hinterland is important:

*“The core is like a residue – the things that stay, the things that can be captured as proposition. Often, such things need to be committed to memory.*

*But if, in certain subjects, for the purposes of teaching, we reduce it to those propositions, we may make it harder to teach, and at worst, we kill it.”*

The original aim of this book was to assist computer science teachers in sharing some of that hinterland with their students, in order to enrich their studies, cement core knowledge in a wider context, and engender an appreciation for the subject that goes beyond what’s required to pass exams and, in many learners, excites a lifelong love for the subject. (page 16 of the book and at <https://www.mrsharrison.co.uk>).

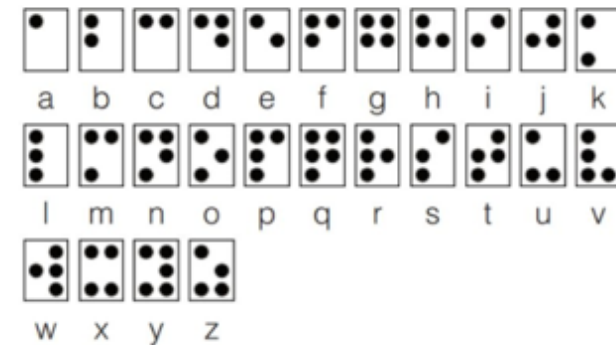


# Inside the book – Data “hinterland” example

## Hand signals

We need to head over to France for an early binary code. Louis Braille injured an eye in his father's leather workshop at the age of three, and the resulting infection caused him to go blind in both eyes by five. At age ten he obtained a scholarship to the Paris Institute for Blind Children, which at the time used a system of raised letters invented by Valentin Haüy. The letter shapes we know and love are not distinct enough to be easily discerned by touch, however, and Braille found Haüy's system hard to learn.

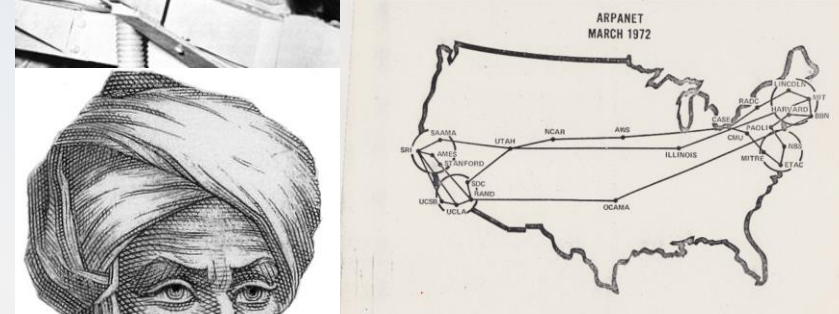
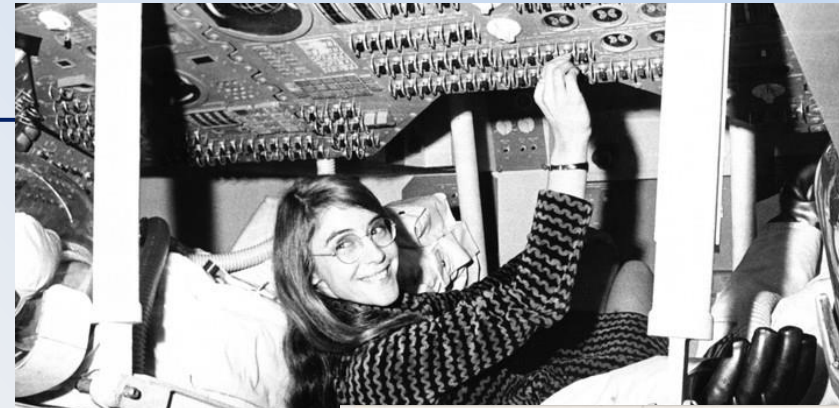
While there, Braille was shown a system of raised dots used by the military to communicate at night. He took this system and improved upon it, using just six dots to represent all the letters of the alphabet plus numbers and some punctuation symbols. Each dot is raised or flat, and a blank space (effectively six flat dots) separates words and sentences. In this way, the grid of six dots could represent  $2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^6$  or 64 different characters.



Braille is therefore a binary code for representing text. If we order the dots, as Braille did, from 1 at the top left, going down then right, finishing with 6 at the lower right, then each of the braille codes can just as easily be written out as a sequence of bumps and flats. So “A” is bump-flat-flat-flat-flat-flat and “H” is bump-bump-flat-flat-bump-flat. Replacing bump with 1 and flat with 0, we can write “A” as 100000 and “H” as 110010. We can now write any text using just two digits, 0 and 1. Braille has created a **binary code** to represent text, and electronic computers have not yet been invented.

# What is our Hinterland?

- History
  - of computing itself
  - in world events
- Impacts and Issues
  - privacy, security, health, environment, ethics and culture
- Theory
  - Computation & info theory
  - Beyond the spec
- Futures
  - VR, AI, ML, automation, bioinformatics, quantum computing, the singularity!



# Hinterland for improved delivery

## Great Explanations

beat fancy resources hands down.

Good teachers must be good communicators.

But to communicate well needs sound subject knowledge.

Learning the **hinterland** helps you teach the **core**.

In my PGCE days I didn't really consider artful explanations as a particularly important area of my practice... it took me a while to realise that a good explanation is worth its weight in gold and is a vital part of every teacher's repertoire – [Adam Boxer](#)

In his book **Why don't students like school?** Daniel T Willingham says **stories** are treated as preferential information, they help with retention. - [Mark Enser](#)

Each of the examples is clear and unambiguous, and explained enthusiastically [by Gary Neville], using self-deprecation and humour [but it is] very unlikely that he could explain something outside his area of expertise to such a high standard. – [Ben Newmark](#)

# Hinterland in the classroom

- Bring the learning to life with a story - as cognitive science shows, narratives are memorable
- Link the topic to other topics (synoptic links) and to other subjects (cross-curricular links)

## Patriot missile rounding error

One of the most serious failures of robust programming resulted in the deaths of 28 American soldiers in the First Gulf War. An Iraqi Scud missile – usually an easy intercept for the Patriot missile defence system – evaded US defences owing to poor coding. The Patriot targeting computer attempted to calculate the time in tenths of a second by multiplying the system clock by 0.1. However, binary cannot exactly represent the decimal 0.1, so the computer makes an approximation, and the fewer bits you have to play with, the less accurate the approximation.

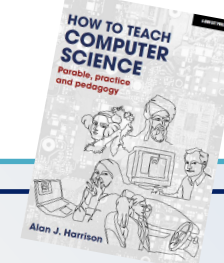
As an aside, you can try this yourself – it makes for a superb classroom discussion. In the Python Shell, or in any Python online IDE (Integrated Development Environment), just type `0.1 + 0.1 + 0.1` and watch your students' eyes light up.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 0.1 + 0.1 + 0.1
0.30000000000000004
>>>
```

Figure 3.3: Decimals must be approximated in binary, leading to rounding errors.

The Patriot targeting computer used just 24 bits to represent 0.1, which meant every calculation was out by 0.000000095 seconds. These errors were cumulative, so after 100 hours of uptime the targeting clock was out by a third of a second. That might not seem a lot, but a Scud missile can travel 600 metres in that time, so when the targeting computer saw empty sky where a missile should be, it failed to launch, and 28 souls perished in February 1991.



# Question...

---

Is hinterland important to

1. the teacher
2. the student
3. both?

Answer with 1, 2 or 3 fingers now!

3. both.

If we know our hinterland we can explain the subject better, use a more narrative style in our explanations, and add interest.

If we share the hinterland with our students, we enrich their experience. Narratives are "preferential information" and more memorable than facts.

# Question...

---

What problem might arise through sharing of the hinterland in the classroom?

Clutter.

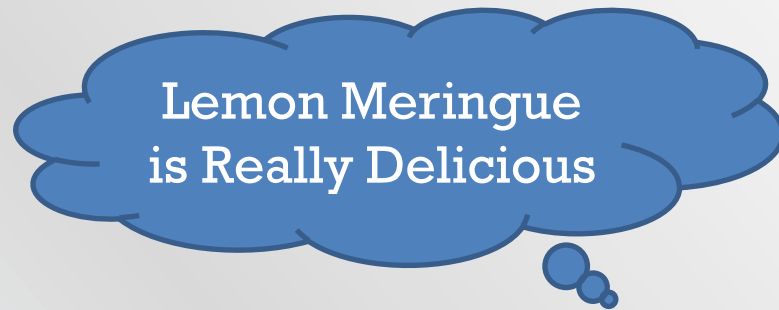
Too much information can prevent the vital "core" being retained. Christine Counsell calls this "clutter" :

in some subjects, reduction to the pure propositions is vital and the last thing one wants is contextual stuff. Even context can be clutter – [Christine Counsell](#)

This is the "hinterland paradox" – we need to ensure the hinterland enriches the core, makes it memorable. Weave a narrative where every story serves a purpose and doesn't distract.

# Back to Mnemonics – an example

This mnemonic helps students remember the effect of binary shifting:



Left = Multiply, Right = Divide

However, using a trick obfuscates the knowledge being taught and fails to build on prior knowledge, that of **place value in maths**. Instead...

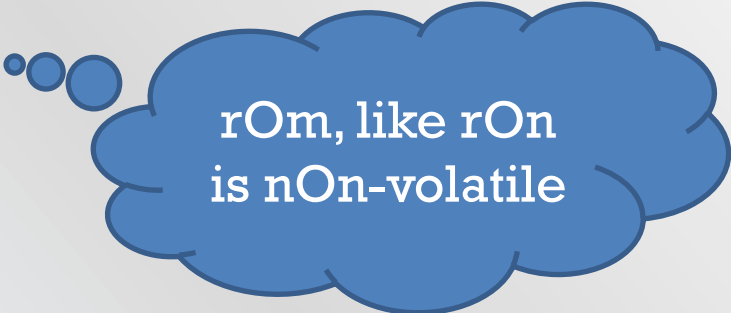
- Recall prior knowledge: place value in maths and practice shifting denary (decimal) numbers:

100	10	1
	3	6
←		
3	6	0
→		
	3	6

- Link this to the new knowledge – binary shifts. Highlight similarities i.e. place value.

# Another Example

This trick helps learners remember that ROM is non-volatile memory



rOm, like rOn  
is nOn-volatile

However a shallow focus on the *words* does nothing to improve **understanding**, and the trick could be misremembered (e.g. rOm is vOlatile!). Instead...

- Recall prior knowledge: what does volatile mean?
  - Evaporates easily (Chemistry)
  - Subject to rapid change
- “If ROM evaporated or changed it would not be much use, would it? So it can’t be volatile”
- “RAM changes all the time, like a volatile person. When the power turns off, its contents evaporate!”

We have made connections across domains, which improves understanding and retention.

# Your turn!

Now pick one of these  
(or your own) and suggest a  
powerful learning episode  
to create lasting  
understanding,  
with no tricks...

Humans Like this  
Language,  
Laptops Like this  
Language

King Midas  
Gained Ten  
Pounds

Killers Must  
Go To Prison

The Crocodile  
Eats the Smaller  
Number: <

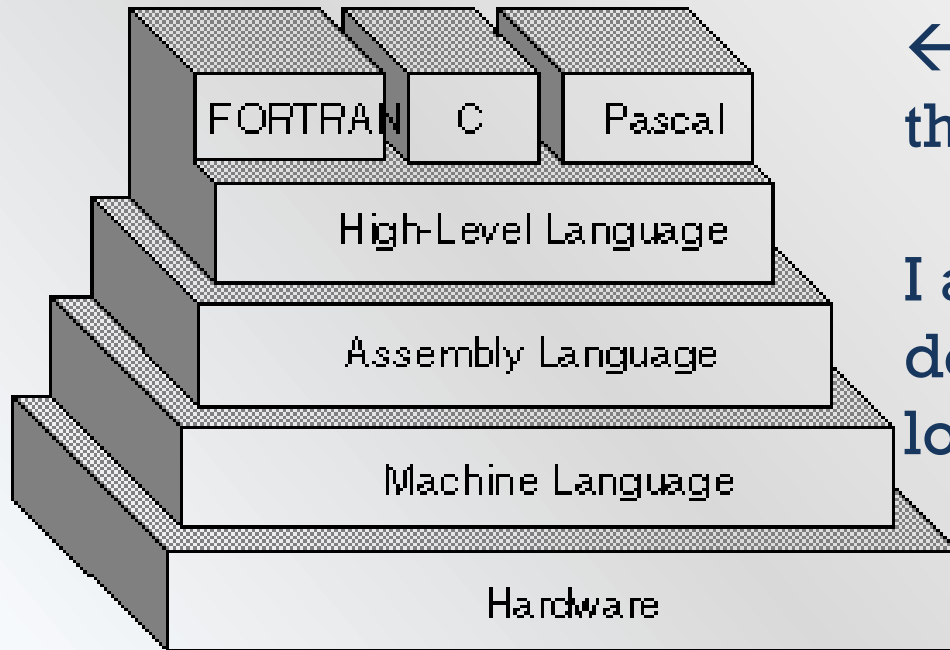
BODMAS

A Tree Is  
Large

Boole Never  
Ate Olives

# My ideas

Humans Like this  
Language,  
Laptops Like this  
Language



High-level and low-level refer to levels of **abstraction** from the hardware. If you teach abstraction well, you can link into this.

Even easier, make sure you use a diagram like this ← to explain the hierarchy of languages. Explain that **low** means nearer to the hardware.

I also stand over a computer in the classroom and declare I am at a high level and the computer is a low level!

# My Ideas...

Killers Must  
Go To Prison

For the quantity multipliers, explain there is nothing special about Bytes or Hertz, that the multipliers kilo, mega etc. are common to all SI units, and we already know grams and kilograms, metres and kilometres...

Unit	×1000	×10 <sup>6</sup>	×10 <sup>9</sup>	×10 <sup>12</sup>	×10 <sup>15</sup>
Byte (B)	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)	Petabyte (PB)
Hertz (Hz)	Kilohertz (kHz)	Megahertz (MHz)	Gigahertz (GHz)	Terahertz (THz)	Petahertz (PHz)
Gram (g)	Kilogram (kg)	100kg = 1 tonne (t)			
Watt (W)	Kilowatt (kW)	Megawatt (MW)	Gigawatt (GW)	Terawatt (TW)	Petawatt (PW)
Metre (m)	Kilometre (km)	not in common use			

# My ideas

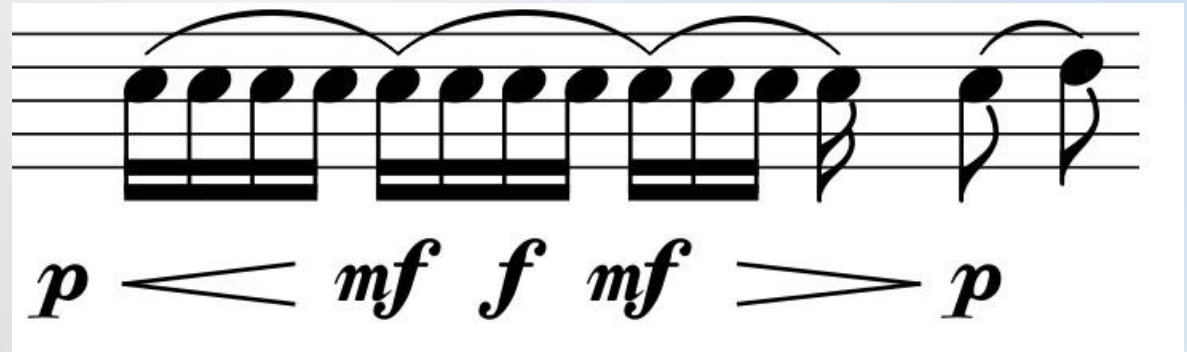
The Crocodile  
Eats the Smaller  
Number: <



I wrote a whole blog about this at <https://online>

The Greater Than > and Less Than < symbols are not accidental. They **already** tell you what they mean by their shapes!

Link this to music, where the diminuendo and crescendo symbols are used:



# How to teach for deeper understanding

- Develop your subject knowledge including
  - Core
  - Hinterland
- Develop your Pedagogical Content Knowledge (PCK)
- Read the Big Book of Computing Pedagogy →
- Attend CPD from the NCCE
- Buy my book!

## Lead with concepts

Support pupils in the acquisition of knowledge, through the use of key concepts, terms, and vocabulary, providing opportunities to build a shared and consistent understanding. Glossaries, concept maps, and displays, along with regular recall and revision, can support this approach.

## Unplug, unpack, repack

Teach new concepts by first unpacking complex terms and ideas, exploring these ideas in unplugged and familiar contexts, then repacking this new understanding into the original concept. This approach (semantic waves) can help pupils develop a secure understanding of complex concepts.

## Create projects

Use project-based learning activities to provide pupils with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Pupils can consider how to develop an artefact for a particular user or function, and evaluate it against a set of criteria.

## Challenge misconceptions

Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.

## Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make) and Use-Modify-Create. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.

## Work together

Encourage collaboration, specifically using pair programming and peer instruction, and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.

## Model everything

Model processes or practices — everything from debugging code to binary number conversions — using techniques such as worked examples and live coding. Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.

## Add variety

Provide activities with different levels of direction, scaffolding, and support that promote active learning, ranging from highly structured to more exploratory tasks. Adapting your instruction to suit different objectives will help keep all pupils engaged and encourage greater independence.

## Make concrete

Bring abstract concepts to life with real-world, contextual examples and a focus on interdependencies with other curriculum subjects. This can be achieved through the use of unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in pupils' lives.

## Read and explore code first

When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.

## Get hands-on

Use physical computing and making activities that offer tactile and sensory experiences to enhance learning. Combining electronics and programming with arts and crafts (especially through exploratory projects) provides pupils with a creative, engaging context to explore and apply computing concepts.

## Foster program comprehension

Use a variety of activities to consolidate knowledge and understanding of the function and structure of programs, including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

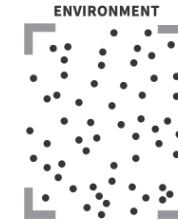
Find out more about our principles and add some or all to your personal pedagogy toolkit.

[nccce.io/pedagogy](https://nccce.io/pedagogy)



# In conclusion

- Tricks like mnemonics are helpful **in certain circumstances but** evidence is **limited**
- Over-reliance on mnemonics and similar memory aids can **limit** understanding, cause **misconceptions** and harm **metacognition**, so use them sparingly.
- More powerful techniques include **storytelling, analogy** and **good explanations, making connections within and across schemas**, and **spaced retrieval**.
- To use storytelling, analogy and cross-curricular links effectively needs good **core** and **hinterland** knowledge.
- I recommend the **Big Book of Computing Pedagogy**, the **Big Book of Computing Content**, William Lau's **book**, Sue Sentance's book, and...



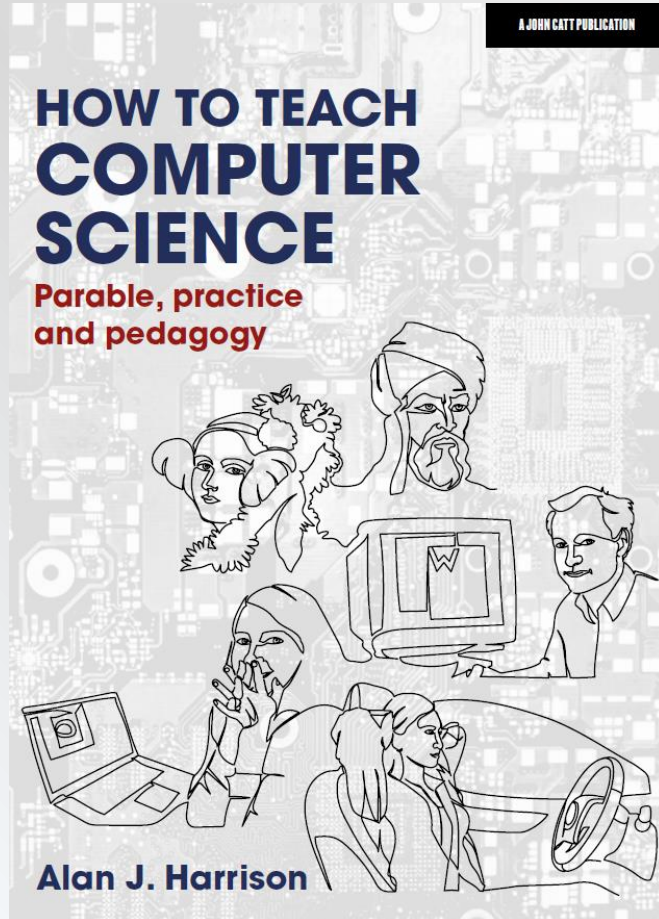
# Your questions revisited

---

Let's look again at the padlet.

[padlet.com/MrAHarrison/questions-em2ag3r4ga3u4plg](https://padlet.com/MrAHarrison/questions-em2ag3r4ga3u4plg)





## How to Teach Computer Science – the book

- That cover art explained!
- What to expect in it
- Sneak previews
- The website <https://online>
- Why I wrote the book
- BREAKING NEWS – the story you would like to see in it

30% off  
with the code

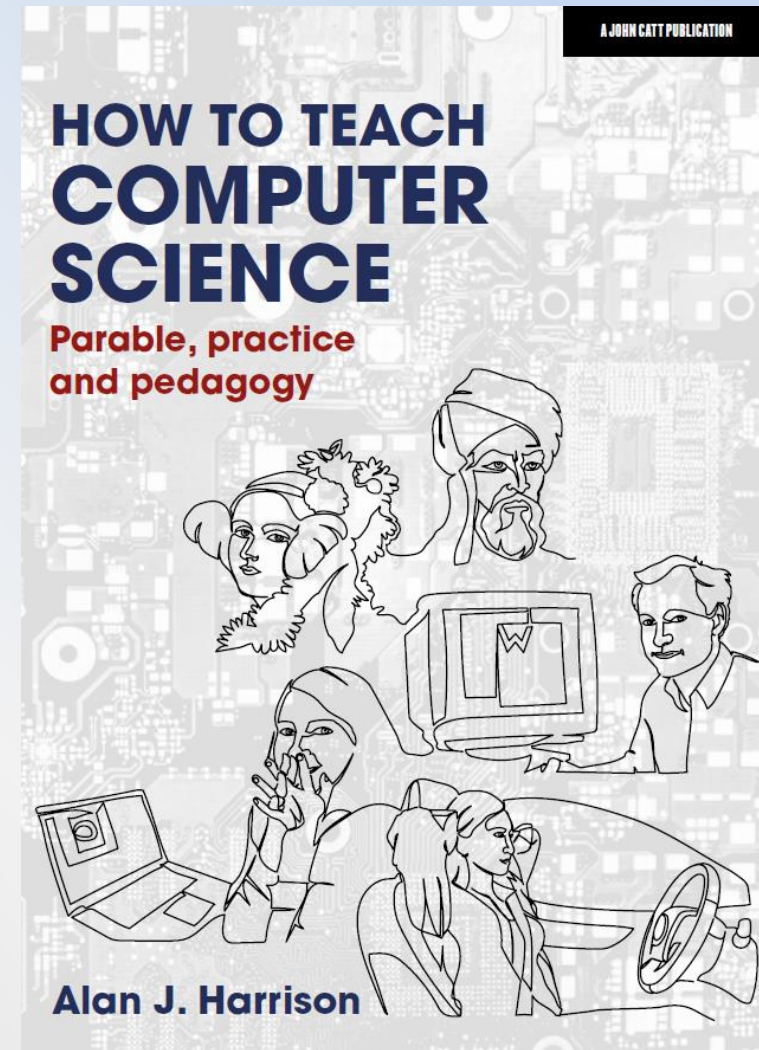
**ilc23**

[johncattbookshop.com](http://johncattbookshop.com)

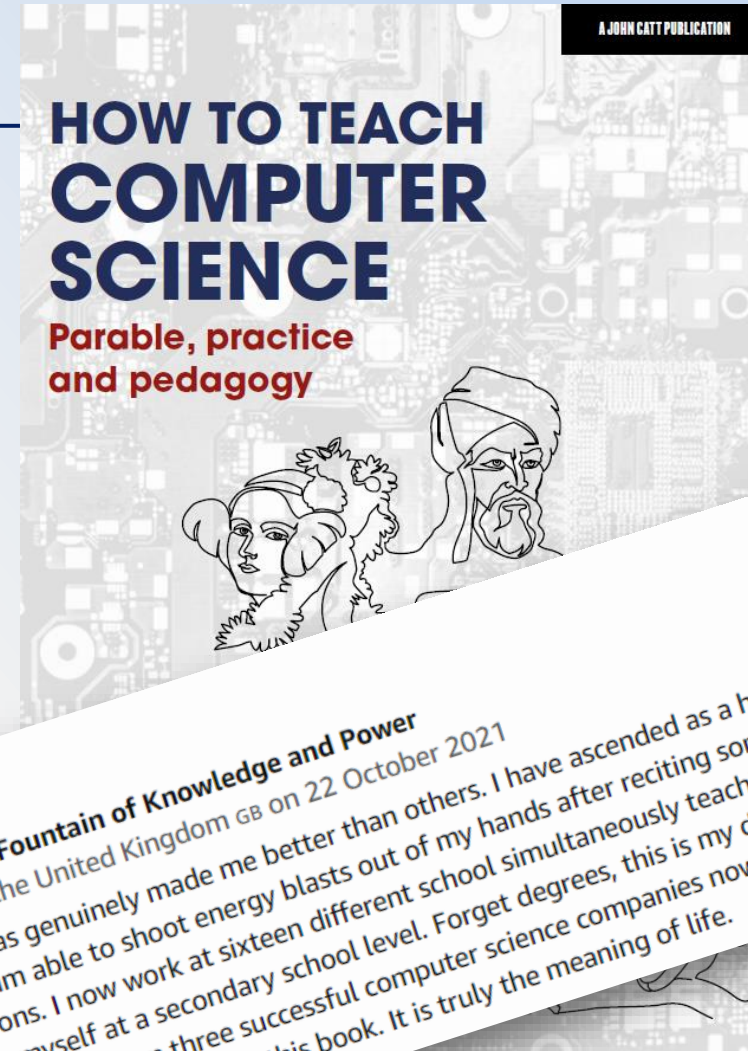


# #HTTCS – Cover Art:


- Top right is Persian scholar **Al-Khwarizmi**, whose name gave us the word algorithm
- Then **Ada Lovelace**, author of an algorithm for Babbage's Analytical Engine.
- Right is **Tim Berners-Lee** at CERN showing off the first ever website
- Lower left is **Katie Bouman**, astrophysicist and computer at MIT whose Python code created the first image of a black hole.
- A woman relaxing in a self-driving car, representing **cutting-edge computing** applications of the 21st century
- A **single line** draws the whole image, reducing the image to its essentials, representing **abstraction**, a key theme of computing which runs throughout the book.




My highlighter's almost run out!



Sawman

 Sawman **Fountain of Knowledge and Power** GB on 22 October 2021  
★★★★★ Reviewed in the United Kingdom  
Book has genuinely made me better than others. I  
want to shoot energy blasts out of my ha  
at sixteen different school s  
er sc

 Sawman **Fountain of Knowledge and Power**  
Reviewed in the United Kingdom GB on 22 October 2021

★★★★★ This book has genuinely made me better than others. I have ascended as a human being and I can talk to CPUs. I am able to shoot energy blasts out of my hands after reciting some hexadecimal combinations. I now work at sixteen different school simultaneously teaching computer science whilst learning it myself at a secondary school level. Forget degrees, this is my degree. I work at Harvard, Oxford and MIT. I own three successful computer science companies now, generating over 53 billion GBP annually in revenue. Buy this book. It is truly the meaning of life.

# Inside the Book 1 – GCSE Content

One chapter per typical GCSE CS topic →

Each chapter covers the “hinterland” of that topic including cultural capital such as

- WW2
- The moon landings
- The “Babington plot”

Then goes on to discuss actionable pedagogy relevant to that topic.

*Introduction*

*1. Data representation*

*2. Programming*

*3. Robust programs*

*4. Languages and translators*

*5. Algorithms*

*6. Architecture*

*7. Logic*

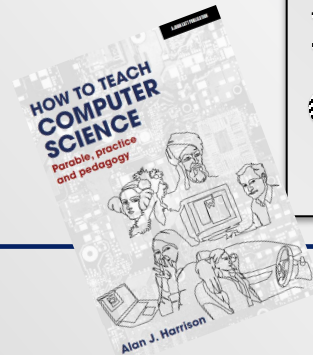
*8. System software*

*9. Networks*

*10. Security*

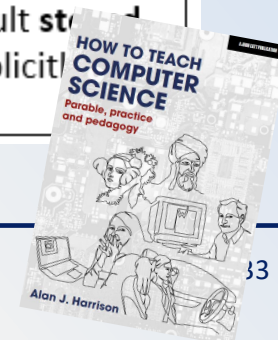
*11. Issues and impacts*

*Conclusion*



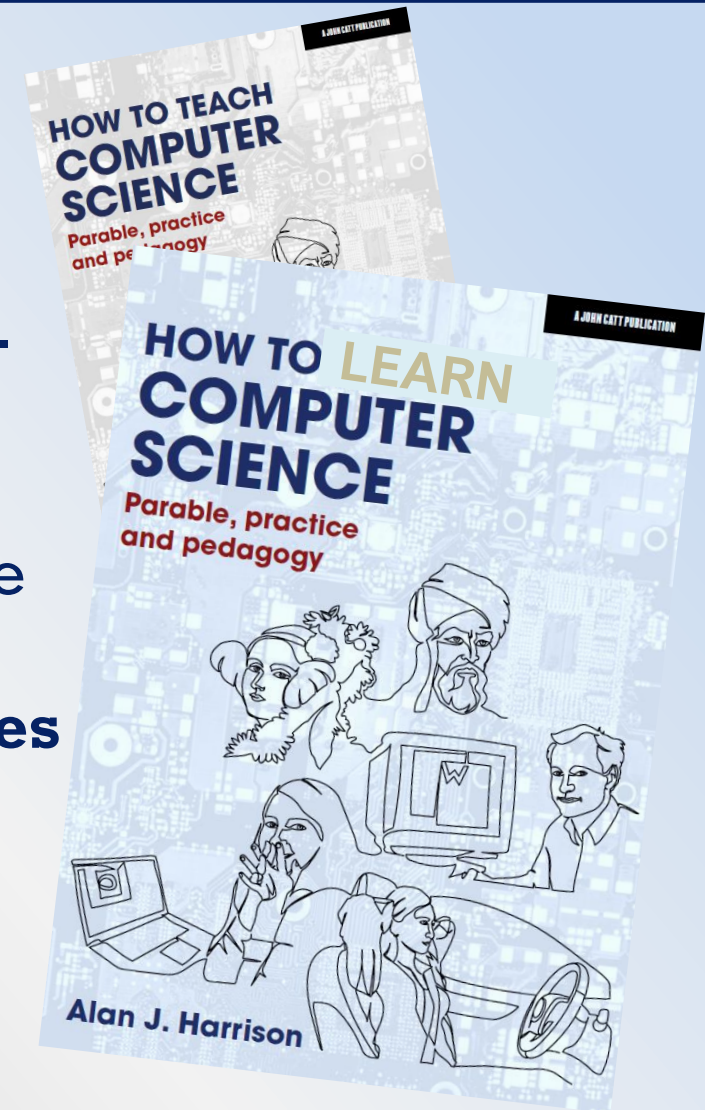
# Sneak preview 2 – Programming misconceptions

Misconception	Reality
Several lines of a program can be simultaneously active, especially a set of assignment statements	Programs are executed sequentially, top to bottom. Only one line of code is executed at a time, and it is not revisited (except through explicit control flow changes, such as loops or branches)
The computer can deduce the intention of the programmer from incomplete statements such as  <pre>score + 1  if age &gt;= 18     "come in"     else "not allowed"</pre>	Statements must be complete and syntactically correct, so the examples given (assuming Python is the teaching language) should be  <pre>score = score + 1  if age &gt;= 18:     print("come in") else:     print("not allowed")</pre>
Assignment statements such as <code>a = b</code> work in both directions, and either swap variables or make them always equal throughout the program execution	Assignment statements are made of two parts, the right-hand side of the assignment operator ( <code>=</code> ) is an <b>expression</b> which is <b>evaluated</b> , and the result stored in the variable on the left. Teaching this explicitly can prevent this misconception



# BREAKING NEWS – student companion

- Entitled “HOW TO LEARN COMPUTER SCIENCE” it contains
  - All the same hinterland content
  - All the “PCK” that is useful to students e.g. core concepts, fertile questions misconceptions.
  - Instead of “teacher PCK” such as cognitive science and gender balance, new **study advice** and **powerful learning techniques** all with concrete examples related to CS



# More information

---

- Find your local Computing Hub or book on courses at [teachcomputing.org](https://teachcomputing.org)
- Start your Computing Quality Framework journey at [computingqualityframework.org](https://computingqualityframework.org)
- Email me [alan@htcs.online](mailto:alan@htcs.online) or Tweet me [@MrAHarrisonCS](https://twitter.com/MrAHarrisonCS)
- Find out more or buy my books at [htcs.online](https://htcs.online)
- Available now at Amazon and JohnCattEd (links on the website)
- Student companion now available also!

Alan Harrison, NCCE PDL and CAS Master Teacher

30% off  
with the code

**ilc23**

[johncattbookshop.com](https://johncattbookshop.com)