Best Practice in Computing

Phil Bagge
code-it.co.uk
Computing Inspector/Advisor Hampshire
Teacher Ringwood, Calmore & Otterbourne Schools
CAS Master Teacher

Slides

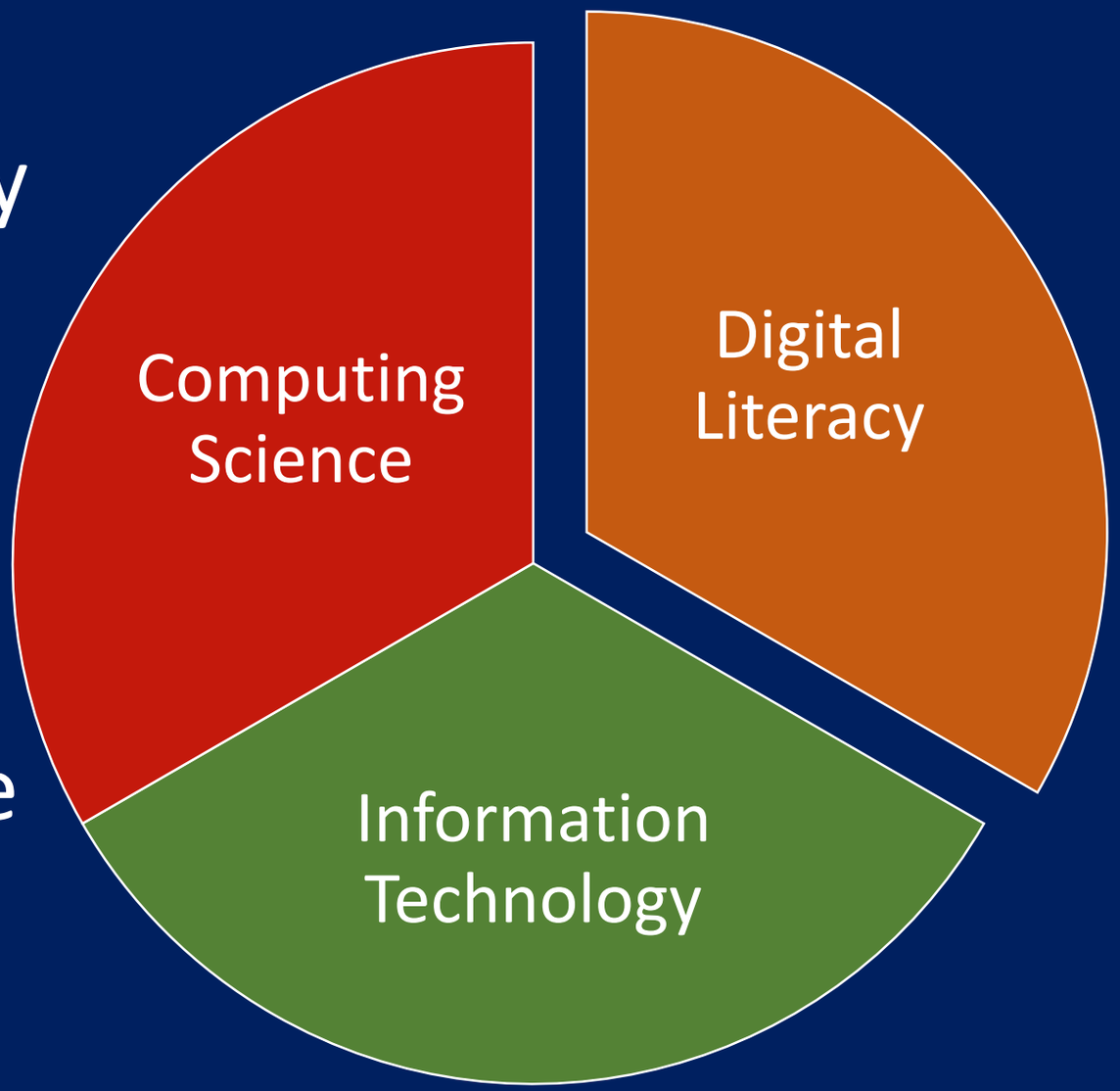
What we believe about a subject affects how we teach it and what we choose to teach

"Pupils are much better at computing than us"

"Pupils are much better at computing than us"

Some pupils who have grown up with technology are more confident when approaching new technology

Let's not mistake confidence for knowledge or understanding



"Pupils are much better at computing than us"

- Encourages us that we don't need to <u>TEACH</u> any computing
- Children know it all already
- Myth of digital native
- <u>https://goo.gl/39cB07</u>

"Pupils are much better at computing than us"

"I'm just learning alongside children"

'I'm just learning alongside children'

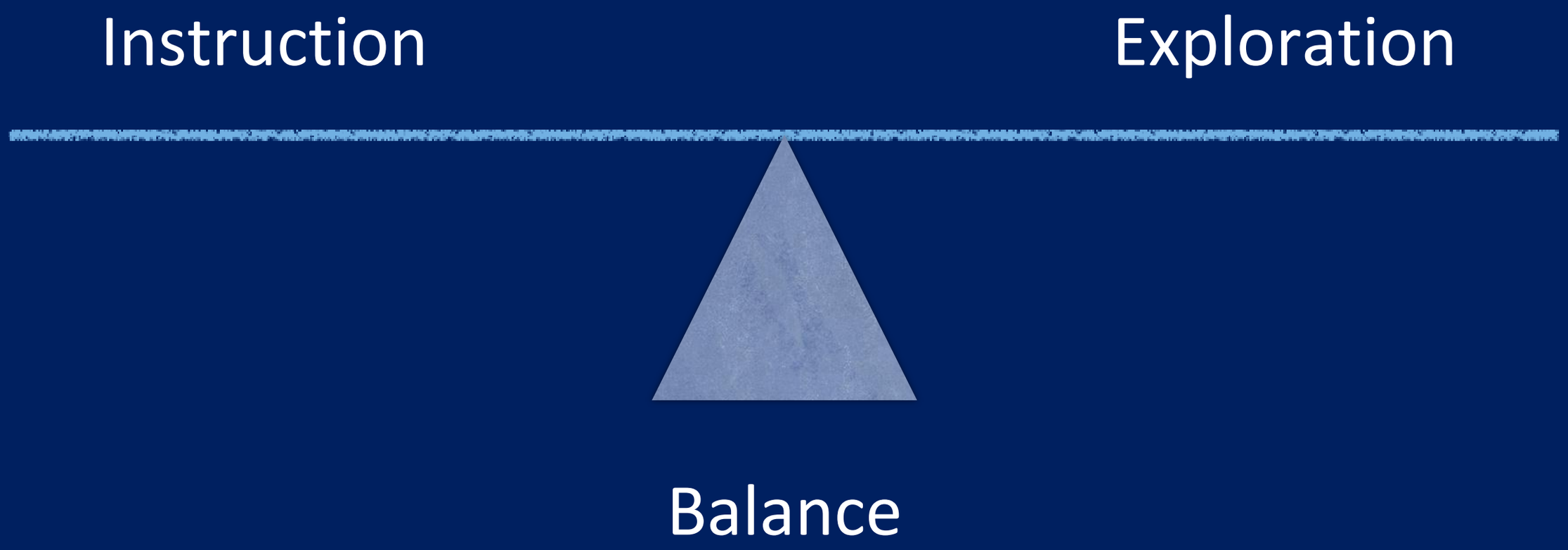
Too much exploration not enough tools or ideas to achieve anything substantial

Instruction

Exploration

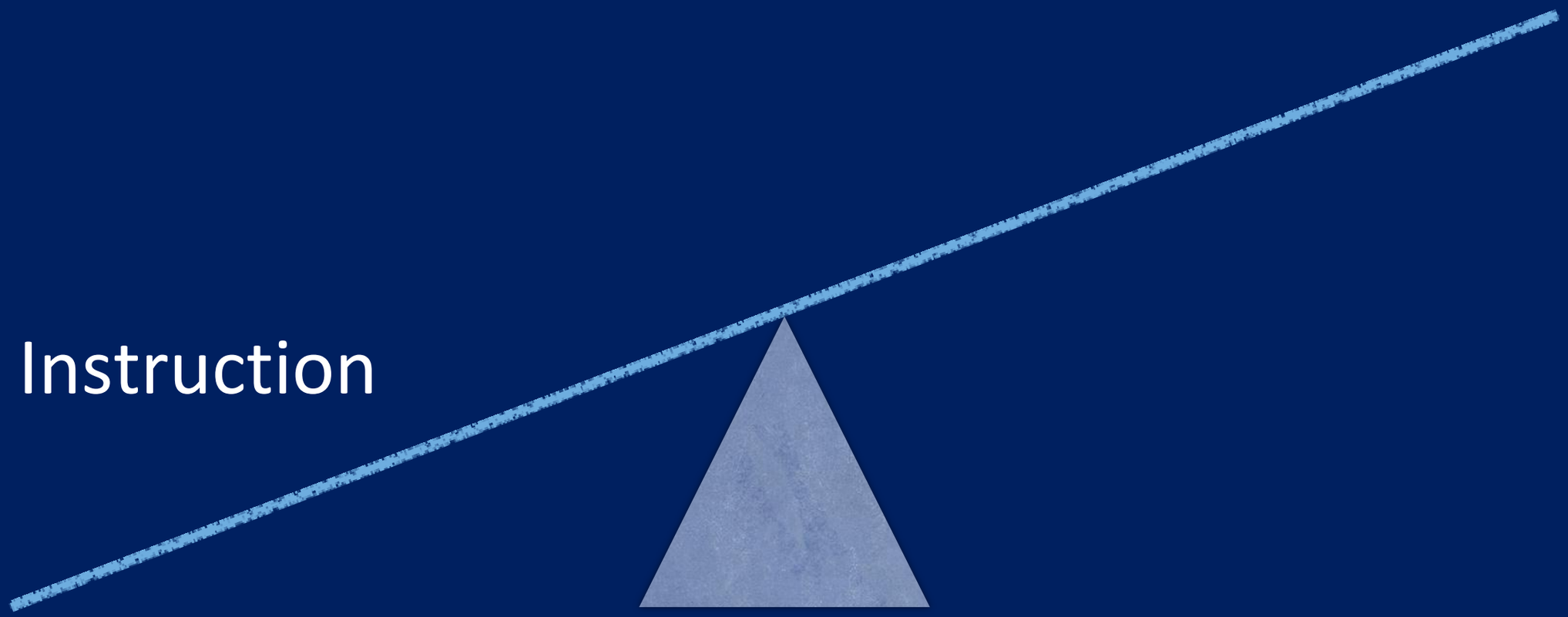
Balance

Great Learning



Too much Instruction shallow learning, concepts grasped at but not internalised

Exploration



Instruction

Balance

Teacher Confidence

'If you don't believe you have anything valuable to teach, why should your pupils?'

Best Practice Grows from…

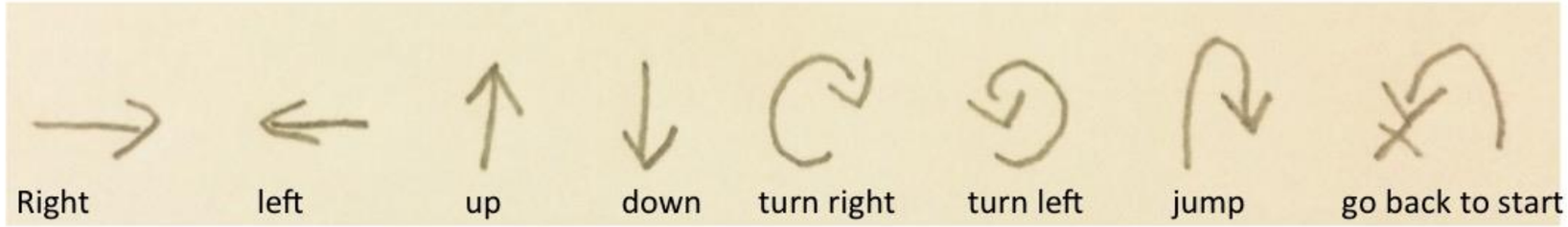
Putting thinking at the heart of your curriculum "Computational Thinking"

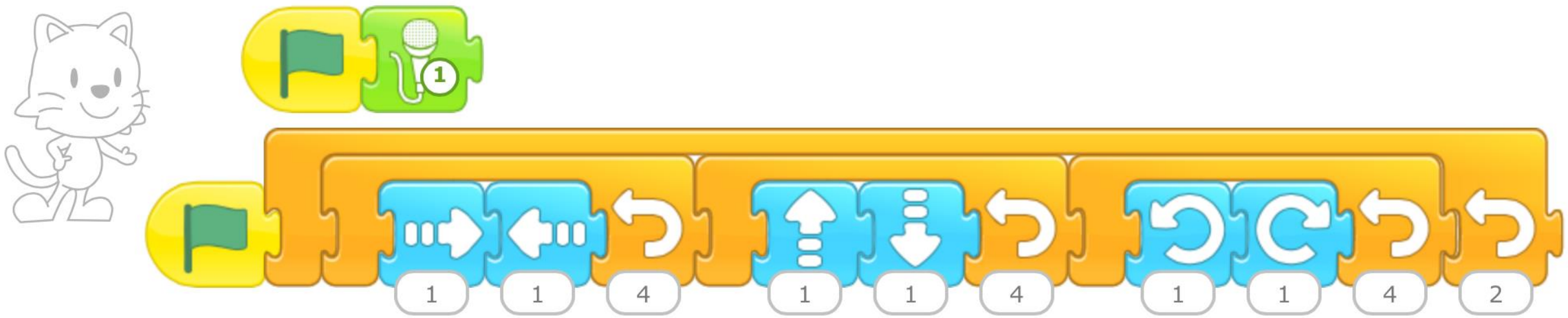
- Algorithmic Thinking
- Evaluating Algorithms
- Generalisation
- Abstraction
- Decomposition

Computational thinking is the ability to think about solving problems with a computer

http://community.computingatschool.org.uk/files/6695/original.pdf

Scratch Jr from algorithm to code





http://code-it.co.uk/scratchjrtravelling
http://code-it.co.uk/scratchjrdance
http://code-it.co.uk/sjmovinggame

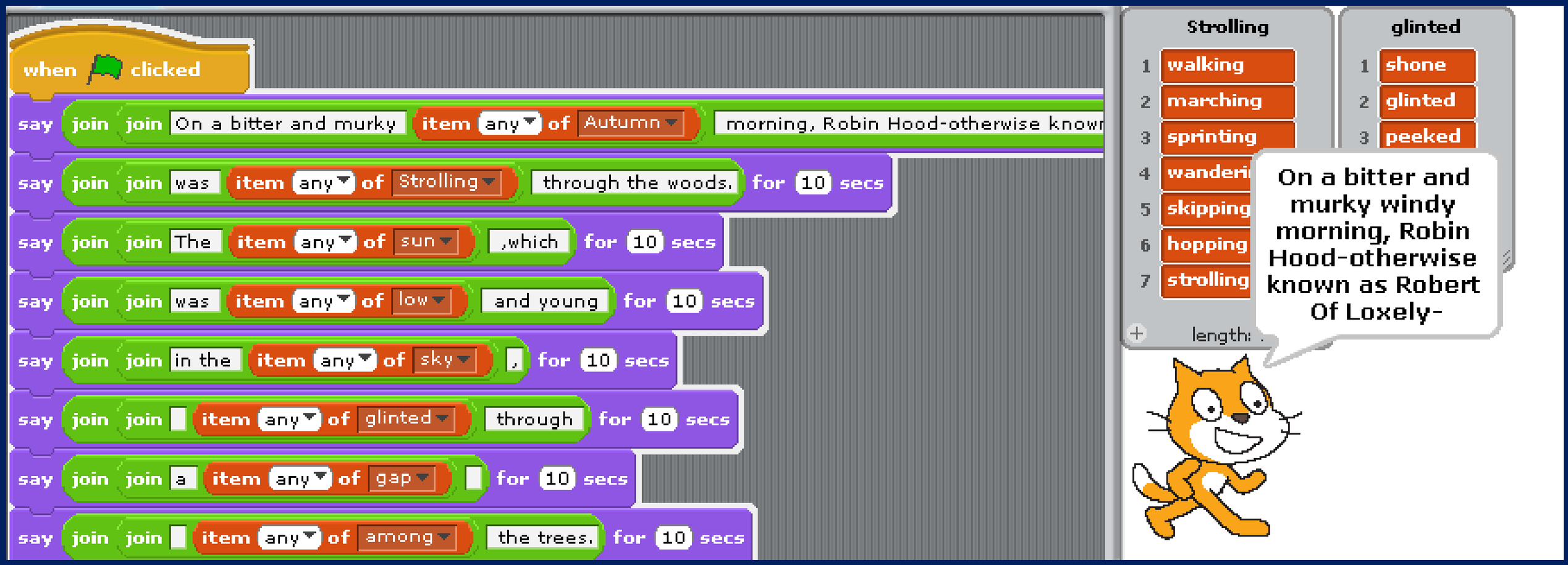
Best practice grows from:

- Building meaningful exciting projects not coding for coding sake
- Not coding by numbers
- Not remote coding

- Remember
  programming is algorithm + code

Best practice grows from:

A variety of programming projects and genre



Laura in Year 5 combining literacy & programming

Best practice grows from:

- Developing good computational doing
  - Developing good problem solving skills
  - Developing good debugging skills
  - Developing pupil independence
  - Developing resilience

My Three Commandments

1. Every programmer makes mistakes
2. Mistakes and debugging are a normal part of the programming cycle
3. Not teacher's job to debug pupil code

Need to encourage independence

Need to liberate pupils into messy problem solving

Opposite to a lot of ICT practice

Not all

Few pupils made mistakes

Every programmer makes mistakes

Mistakes were not normal

Mistakes and debugging are a normal part of the programming cycle

As focus was on other areas on the curriculum teachers felt duty to step in making sure learning could continue in say literacy for example

Not teacher's job to debug pupil code

Need to encourage independence

Need to liberate pupils into messy problem solving

Consequences of previous ICT practice is

Teachers who solve things for pupils & dependant helpless pupils

Helpless Dependant Pupils

- Define Helpless
  - Helpless ≠ stuck
  - Helpless = stuck + no attempt to find solution
- Two types
  - Sweet helpless
  - Aggressive helpless

For Example

What aspect are you stuck on?

Everything

Can you describe the problem?

No answer

Which parts do work?

No answer

Why are they helpless & dependant?

- Most don’t come into school helpless
- They learn it at school
    - Emphasis on finished product over process
    - Teachers & pupils who solve things for them
    - Minimum output for maximum reward
    - Teacher modelling helpless attitude to technology themselves

Pupils do what we do not what we say!

Strategies to overcome learnt helplessness

Challenge the helplessness and reveal it for what it is

"Are you trying to make me do your work for you?"

To understand what you are doing is the beginning of change

With Pupils

Strategies to overcome learnt helplessness

Explain why resilience is really important

If the stick is challenging their learnt helplessness, the carrot needs to be an appreciation of the importance of perseverance and resilience.

Over the last five years I have spent lots of time explaining to pupils why developing resilience, perseverance and a desire to solve problems is important.

Care for their longer term interests and development is important

With Pupils

Strategies to overcome learnt helplessness

Promote bug and debug language

It is much less personal and doesn't indicate blame

What debugging strategies can you use?

Do you have a bug?

You have made an error

How can you fix your mistake?

With Pupils

Strategies to overcome learnt helplessness

It may take time

Recognise that it may take time to change our own habits of fixing things for pupils. Maybe we have become used to the ‘geek’ praise that we get from others when we solve their problems. I know I had!

It took me a few months and I still find on occasions that I desire to just fix something for a pupil or teacher that I am training

In upper KS2 it often took about five sessions to really address this and turn the corner

With Ourselves

Strategies to overcome learnt helplessness

Teacher as hint or strategy provider

NOT solution provider

It is not my job to fix your code

Year 5 pupil

'I know it is not your job to fix this for me but do you have a hint I can try?'

Year 6 Pupil

'You are evil Mr Bagge'

Me

'Why?'

Year 6 Pupil

'Because you won't fix it for us like everyone else'

With Ourselves & Pupils

Strategies to overcome learnt helplessness

Teacher as hint or strategy provider

NOT solution provider

It is not my job to fix your code

It is about good questioning

Does your question help pupils discover the fix for themselves or does it give them the answer?

With Ourselves & Pupils

Strategies to overcome learnt helplessness

Digital Leaders

Get pupils on board with these approaches

Hints not Solutions

If it is code based, they are allowed to say blocks that are useful in Scratch or commands in other programming languages but not share solutions.

Hands Off

If it is a '*how to*' solution, they can point to the place a person needs to go first and if they still need help can describe in hands off manner what to do.

No one learns anything by having it done for them

With Pupils

Strategies to overcome learnt helplessness

Avoid language that describes computers as human

- My computer hates me!
- It never does what I tell it to!
- It never crashes when you are here!

Computers are deterministic machines. This means that the same input always leads to the same output.

Humans are not deterministic. If you humanise the machine, you encourage pupils to think that they may not be able to debug something due to its capricious nature.

With Pupils

Strategies to overcome learnt helplessness

Get other adults on board with these approaches

Hints not Solutions

Hands Off

Provide a few debugging strategies they can use

Some will find this really difficult as their sense of job worth has been tied up with being a solution provider

With Other Adults

Strategies to overcome learnt helplessness

Institutionalise this approach

Share with staff

Share with senior managers

Share with governors

Write it into school computing policies

In doing so you are demonstrating the awesome power of '*computational doing'* as a force for good in your school

This can be part of the legacy computing gives to your school

With Other Adults

Best Practice in Computing

- Teachers believe that computing is worth teaching and that they can add value to pupils knowledge, understanding & skills
- Computational thinking is central
- Meaningful exciting creative projects
- Computational doing builds independence & resilience (pupils work harder than teachers)

Thanks for listening

Come and chat

Phil Bagge

@baggiepr

code-it.co.uk

'How to teach primary programming using Scratch'

http://goo.gl/W4bQ1a