Objectives of this session

Increase understanding of control specifically: RAG

- Vocabulary associated with control and the control model
- Examples of real world and school control (including physical devices)
- Using design to represent control
- Progression of control
- Pedagogy for teaching control

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Science, DT and Computing

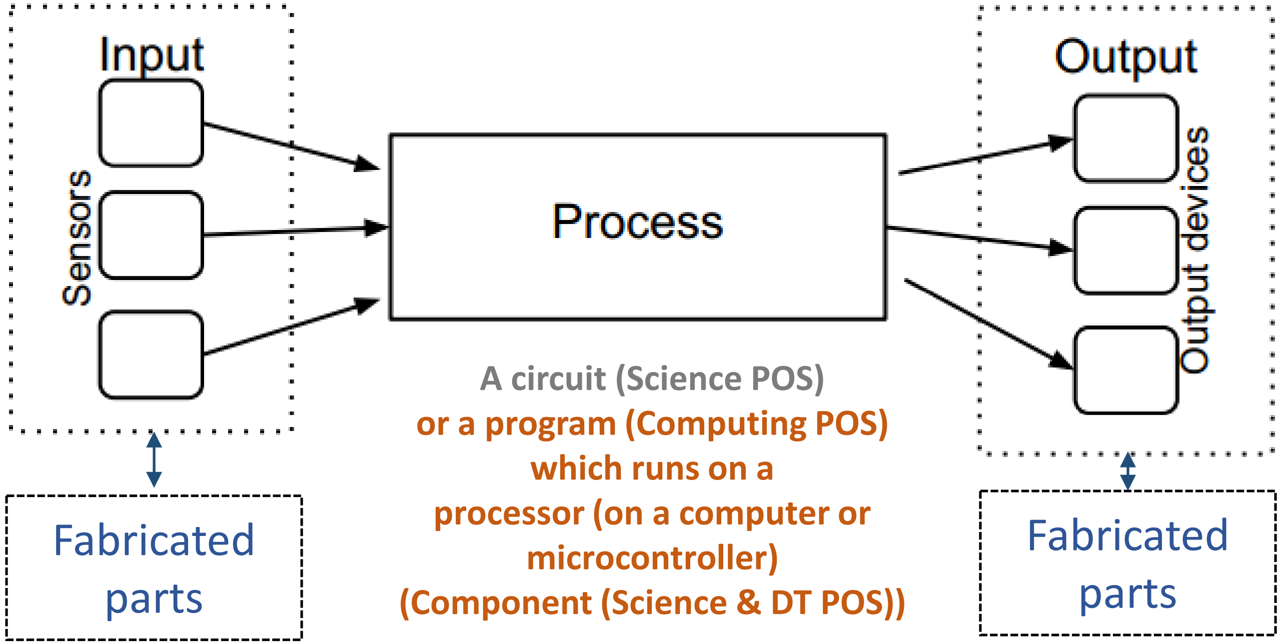
Activity 4

	Science - Electricity	DT	Computing
Year 4	● construct a simple series electrical circuit, identifying and naming its basic parts, including cells, wires, bulbs, switches and buzzers ● identify whether or not a lamp will light in a simple series circuit, based on whether or not the lamp is part of a complete loop with a battery ● recognise that a switch opens and closes a circuit and associate this with whether or not a lamp lights in a simple series circuit ● recognise some common conductors and insulators, and associate metals with being good conductors	● How simple electrical circuits and components can be used to create functional products ● How to program a computer to control their products *(split by data.org)*	● Design, write and debug programs that accomplish specific goals, including controlling physical systems ● Use sequence, selection and repetition in programs ● Work with variables and various forms of input and output
Year 6	● compare and give reasons for variations in how components function, including the brightness of bulbs, the loudness of buzzers and the on/off position of switches	● How more complex electrical circuits and components can be used to create functional products ● How to program a computer to monitor changes in the environment and control their products	

Science, DT and Computing

Physical systems (Computing POS) = Functional Product (DT POS) = Circuit (Science POS)

Input (Computing POS)
Switches (Science POS)
Components (DT & Science POS)
Monitor (DT POS)



A circuit (Science POS) or a program (Computing POS) which runs on a processor (on a computer or microcontroller) (Component (Science & DT POS))

Output(Computing POS)
Bulbs, lamp, buzzers (Science POS)
Components (DT POS)
Control (Computing & DT)

Physical systems include a process which is a hard wired circuit or a program which runs on a processor (on a computer or microcontroller) which monitors or controls inputs and outputs. The inputs and outputs are electronic components which work with fabricated parts (mechanical and user interface).

Designs perhaps include: algorithm design; data structure design; user interface design; electronic component design; fabrication/mechanical design - however some of these are perhaps not design choices but implementation choices.

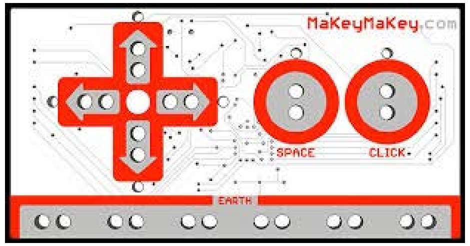
Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

With thanks to Phil Bagge on ideas here

A

B

C



D

E

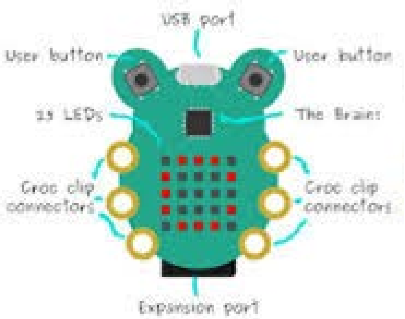
F

G

H

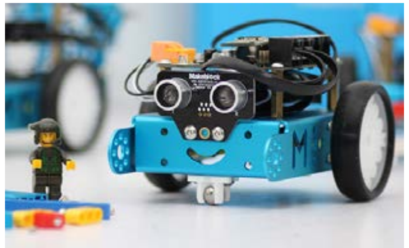
I

J

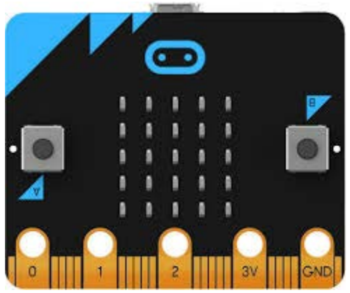


K

L

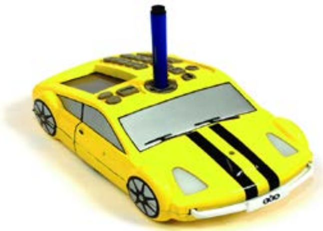


M

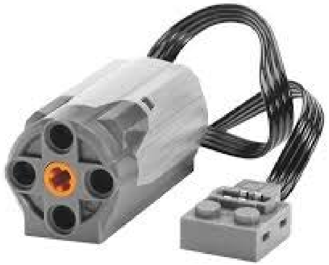


Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

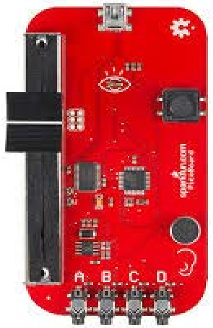
N



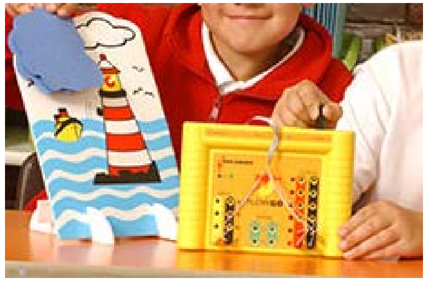
O



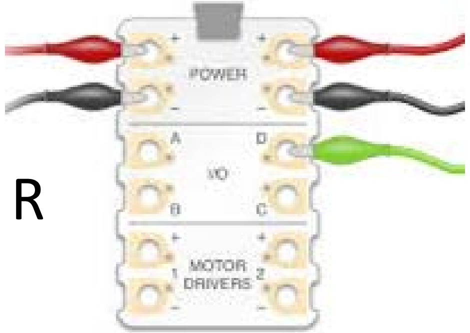
P



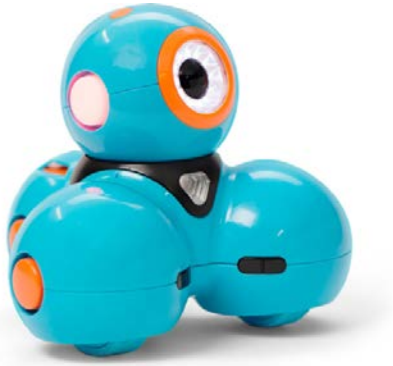
Q



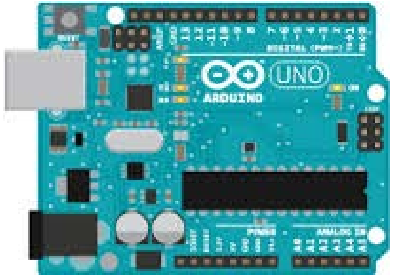
R



S

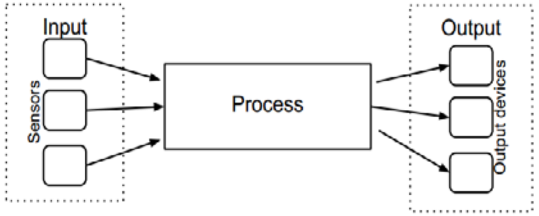


T



U

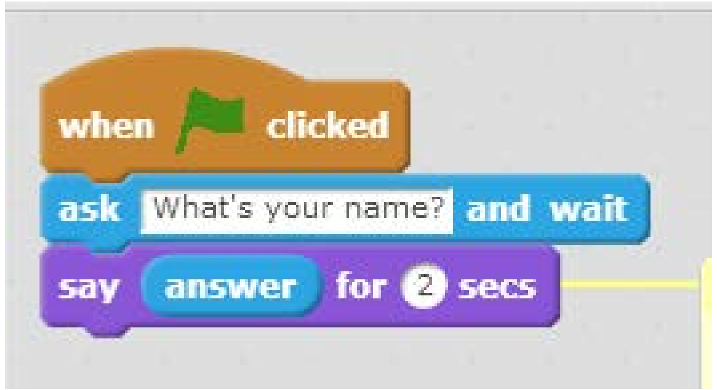
Control



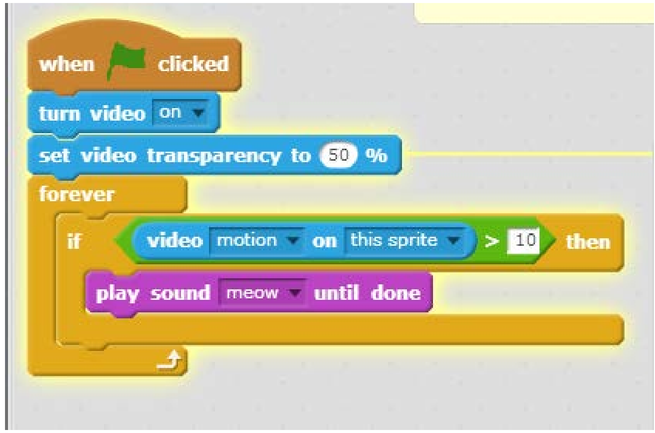
4. What **techniques/ commands** are often used to implement (code) variables?

Predict what this code will do.

Say what are the inputs and what are the outputs.



```
when [flag] clicked
ask [What's your name?] and wait
say (answer) for (2) secs
```



```
when [flag] clicked
turn video [on]
set video transparency to (50) %
forever
    if <(video [motion] on [this sprite]) > (10)> then
        play sound [meow] until done
```

Predict what code has been used.

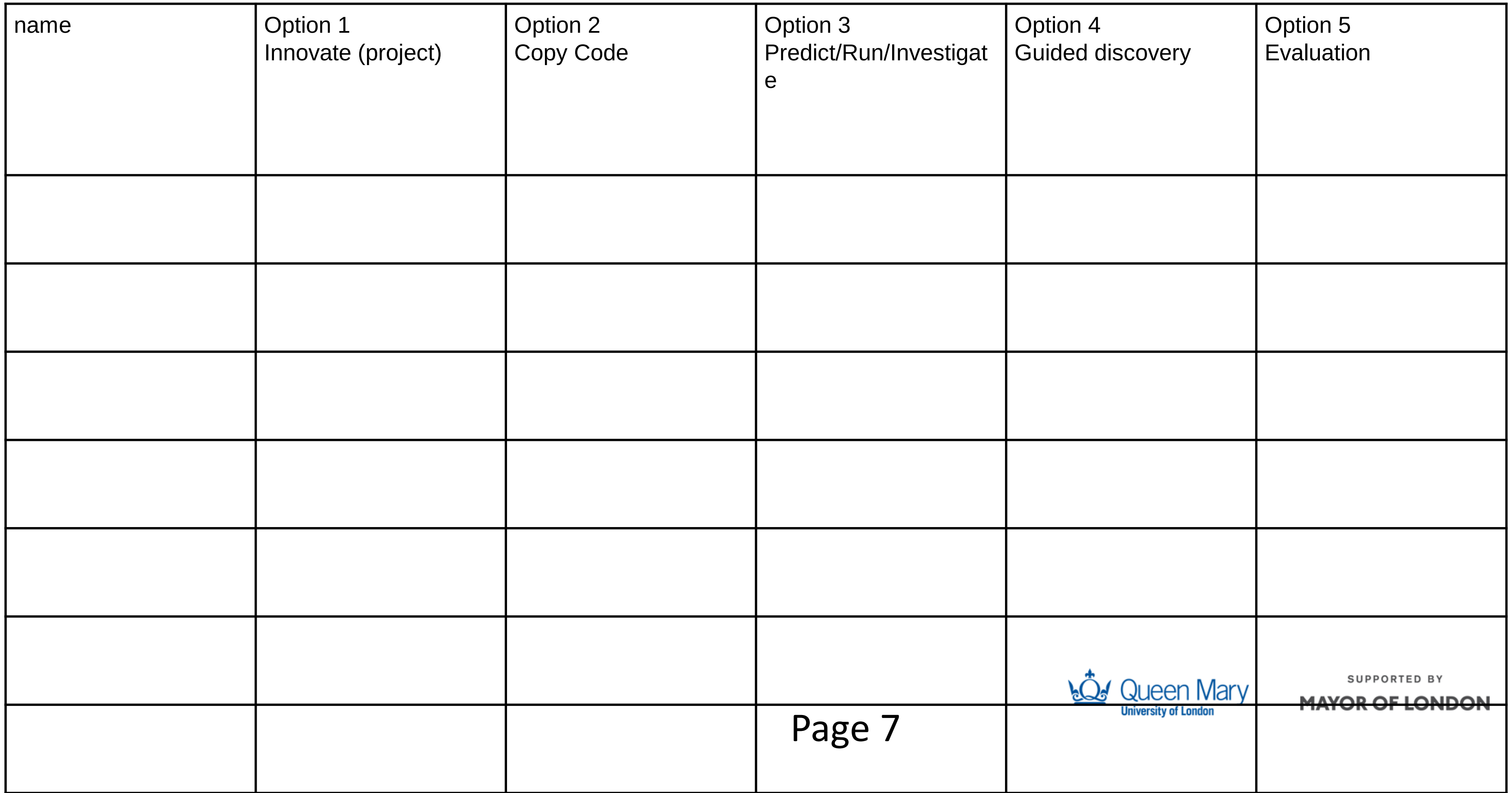
Scratch control code 1 guess what code is used https://scratch.mit.edu/projects/97038386/#player

Scratch control code 2 guess what code has been used *https://scratch.mit.edu/projects/97039379/*

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Comparing instructional approaches in teaching control

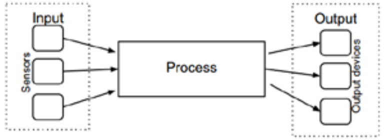
1. Add your name to the table in the google doc
2. Try some of the options.
3. Make a note in the table of advantages and disadvantages of each instructional approach.



name	Option 1 Innovate (project)	Option 2 Copy Code	Option 3 Predict/Run/Investigate	Option 4 Guided discovery	Option 5 Evaluation

Option 1

Control



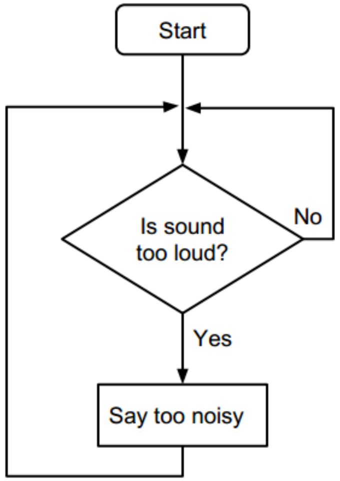
<u>Sound monitor algorithm</u>

If quiet say that is ok

else say that is too noisy

1. What code do you think you would need?

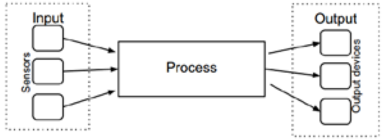
2. Build it!

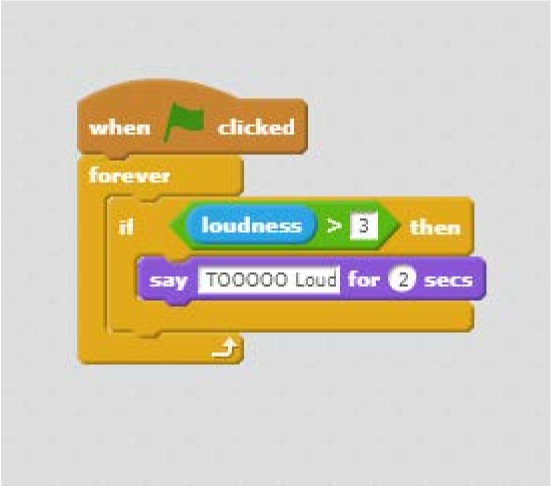


Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Option 2

Control





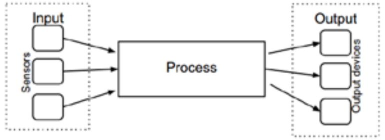
```
when [flag] clicked
forever
    if loudness > 3 then
        say TOOOOO Loud for 2 secs
```

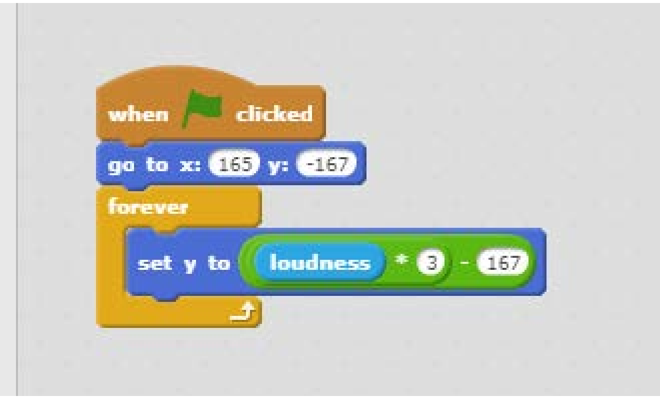
Copy this code.

Well done you have made a sound monitor.

Option 3

Control





```
when 🏴 clicked
go to x: 165 y: -167
forever
    set y to loudness * 3 - 167
```

1. Predict what this code will do

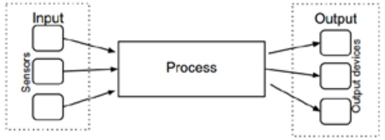
2 Run it https://scratch.mit.edu/projects/36133098/

3. Investigate the values 3 and 167.

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

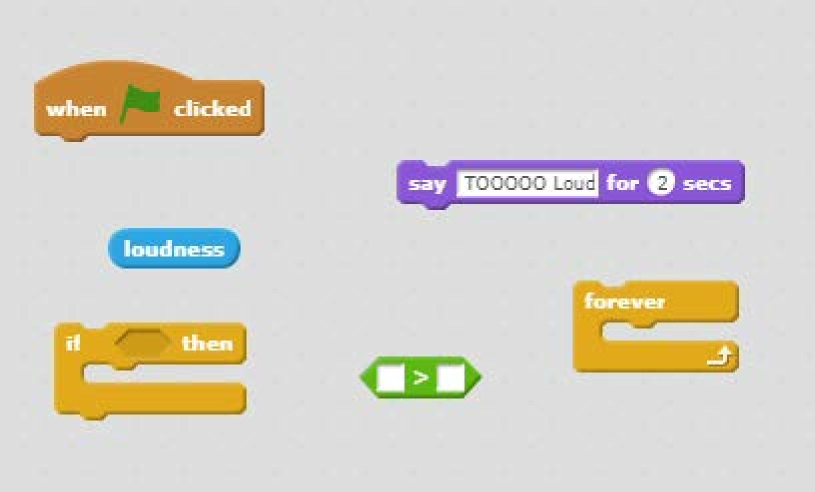
Option 4

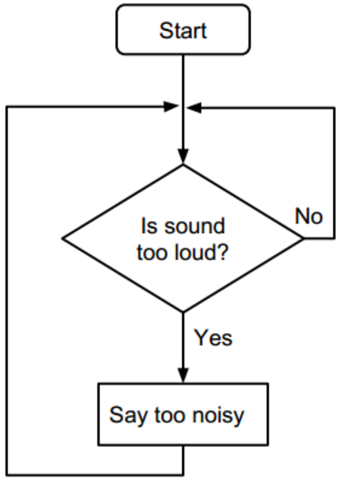
Control



1. Explore these commands

2. Can you make a sound monitor that
3. Implements this algorithm?





Sound monitor algorithm

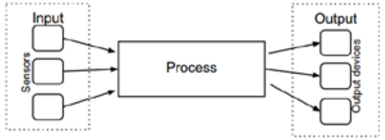
If noisy say that's too noisy

https://scratch.mit.edu/projects/181940153/#player

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Option 5

Control



Look at the code in these programs in this studio, which is the best sound monitor? Why? Explain to the group the difference between the sound monitors.

https://scratch.mit.edu/studios/701450/

Barefoot classroom sound monitor

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Objectives of this session

Increase understanding of assessment, differentiation & progression specifically:

RAG

- Forms of assessment for programming
- Progression in programming
- Approaches for differentiation for programming

Assessment activities

1. Read the assessment working group paper. How does this compare to our ideas? Add to the A3 sheets? (5mins)

2. Compare the ELIM progression statements to the Rising Star assessment books. Do they match? Feedback to group. (15 mins)

3. Can you make a detail progression for one or more of these?

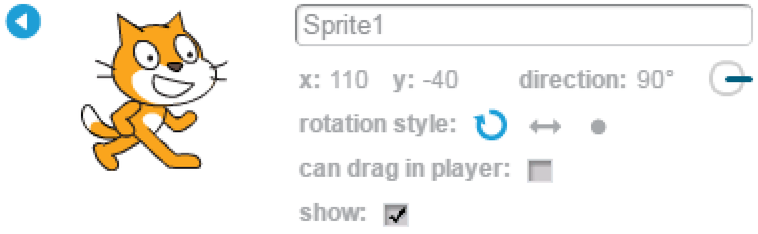
- Sequence
- Repetition
- Selection
- Variables
- Procedures

Compare your progression to Rich et al and Jane’s crazy ideas & compare to your SOW. Have you incorporated general theories? (Bloom/Solo/ KSU) (30 mins)
Where does quality code/bad smells come in?

4. Undo the progression of the Hard/Easy Maths quiz. (10 mins)

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Bad smells



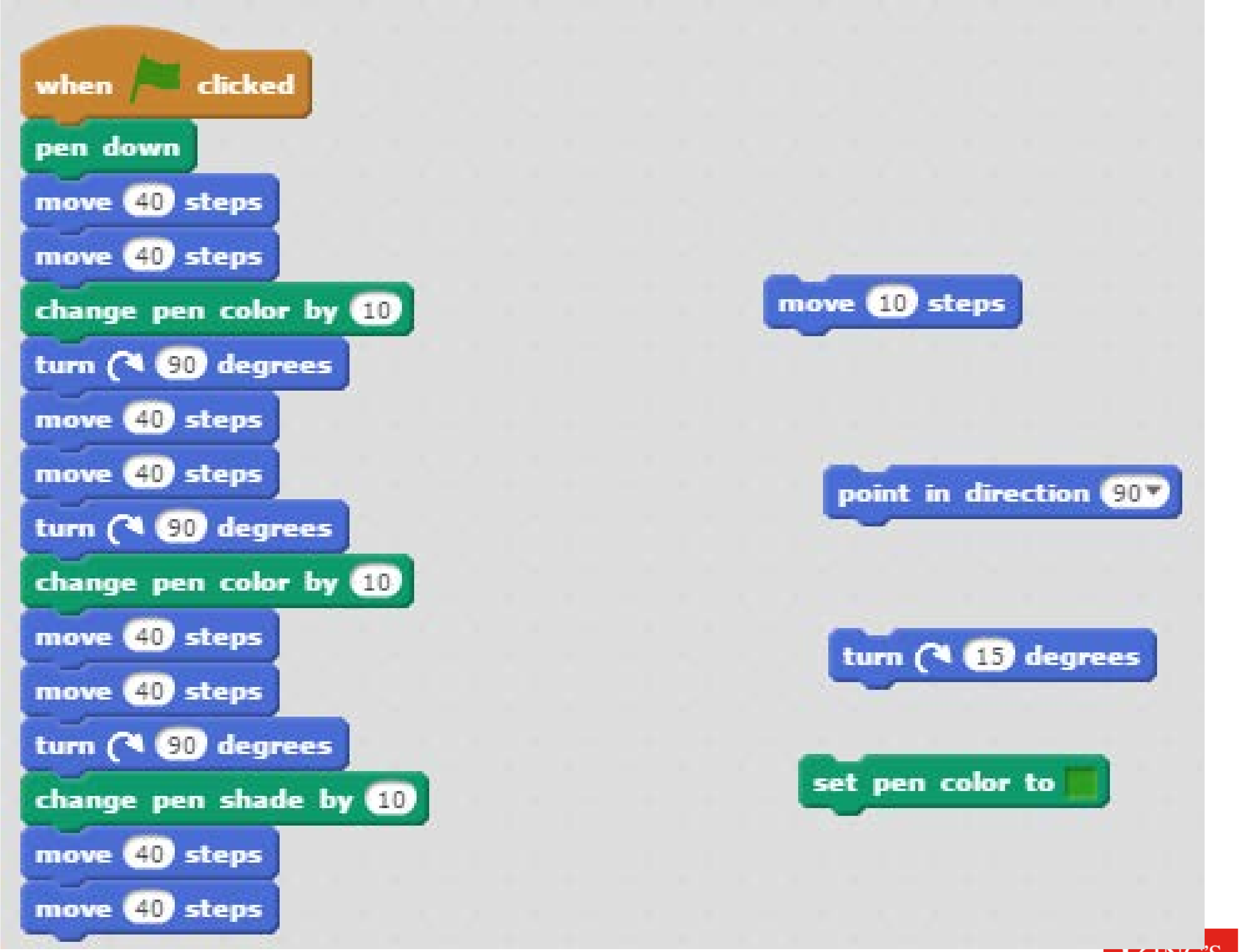
Sprite1

x: 110 y: -40 direction: 90°

rotation style:

can drag in player: □

show: ☑



```
when 🏴 clicked
pen down
move 40 steps
move 40 steps
change pen color by 10
turn ↻ 90 degrees
move 40 steps
move 40 steps
turn ↻ 90 degrees
change pen color by 10
move 40 steps
move 40 steps
turn ↻ 90 degrees
change pen shade by 10
move 40 steps
move 40 steps
```

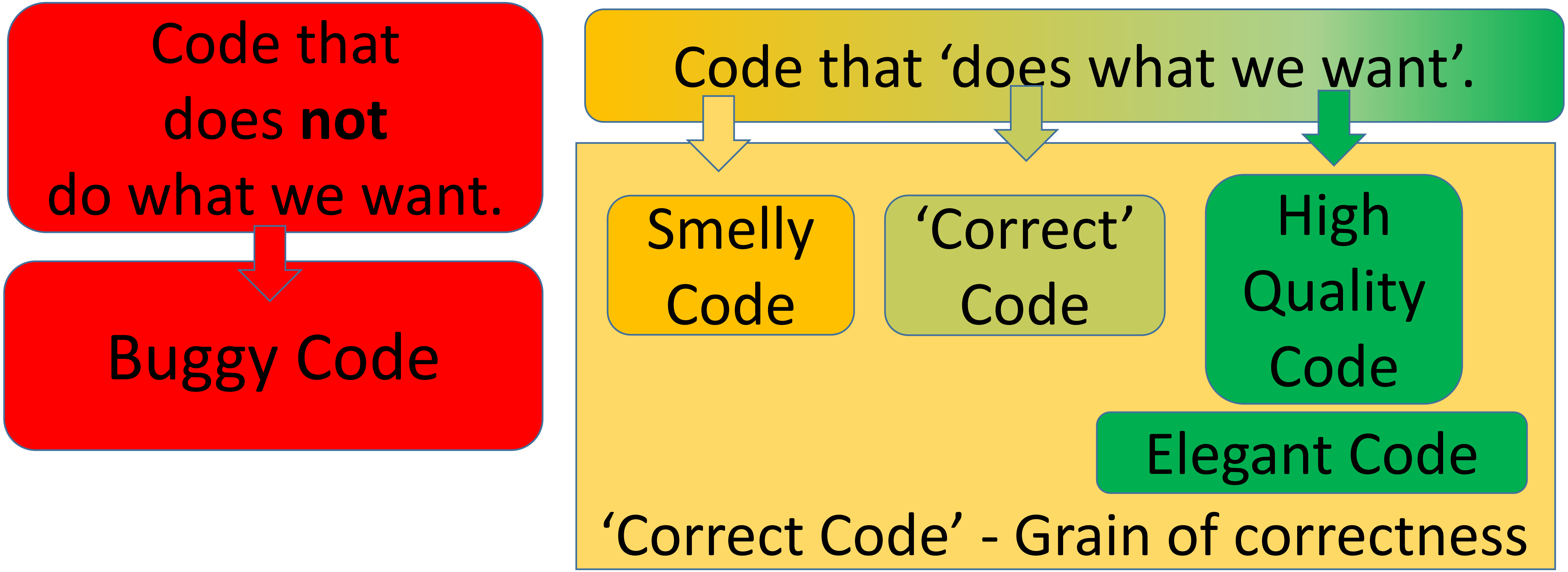
```
move 10 steps

point in direction 90▾

turn ↻ 15 degrees

set pen color to 🟩
```

Handout 2

https://scratch.mit.edu/projects/164525941/

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

What might we call?



Code that does **not** do what we want.

Buggy Code

Code that ‘does what we want’.

Smelly Code

‘Correct’ Code

High Quality Code

Elegant Code

‘Correct Code’ - Grain of correctness

Evaluation. What is correct? What is quality? What is progression?

Which relate to smells, bugs, progress
What year group might you teach these to?

Handout 1

Have you:

1. Cleaned up your scripting area?

2. Cleaned up unused sprites?

3. Cleaned up unused costumes?

4. Removed code that you could combine e.g. move 3 move 5?

5. Renamed your sprites, backgrounds etc so it is easy to know what they do?

6. Made sure your code does what you/ your user wanted/designed?

7. Created a start up ‘make your own block’ for every sprite?

8. Used a loop rather than repeated code?

9 .Used a procedure to decompose your code into parts that are then easy to understand & change?

10. Used a procedure to reuse some code?

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Sequence - progression?

Switch background + no sprites

Single Sprite
- no concurrency & no coordination

Multiple Sprites
- concurrency & no coordination

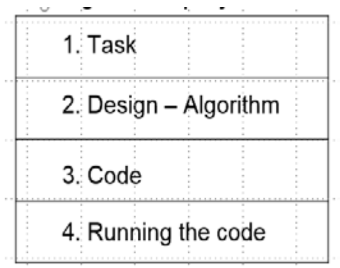
Multiple Sprites
- concurrency - use time to manage coordination

Multiple Sprites
- concurrency - use broadcast to manage coordination

Multiple Sprites
- concurrency - use broadcast & wait to manage coordination

Multiple Sprites
- concurrency - use variables to manage coordination

Use design annotations and ragging rather than copy code to scaffold independence?



1. Task
2. Design – Algorithm
3. Code
4. Running the code

bcs Academy of Computing

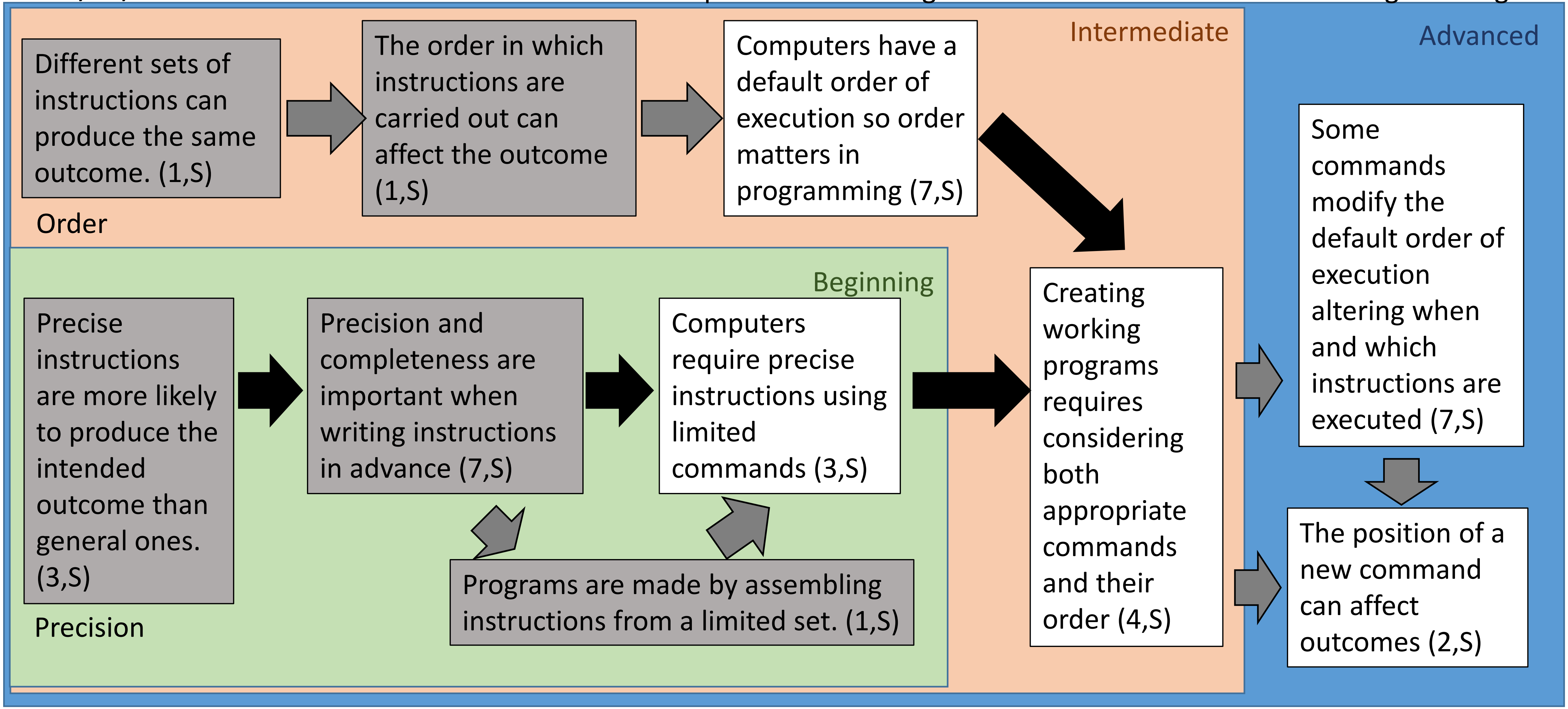
Queen Mary University of London

SUPPORTED BY MAYOR OF LONDON

COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE

KING'S College LONDON

Everyday Computational thinking Programming



Intermediate

Advanced

Beginning

Order

Different sets of instructions can produce the same outcome. (1,S)

The order in which instructions are carried out can affect the outcome (1,S)

Computers have a default order of execution so order matters in programming (7,S)

Creating working programs requires considering both appropriate commands and their order (4,S)

Some commands modify the default order of execution altering when and which instructions are executed (7,S)

The position of a new command can affect outcomes (2,S)

Precision

Precise instructions are more likely to produce the intended outcome than general ones. (3,S)

Precision and completeness are important when writing instructions in advance (7,S)

Computers require precise instructions using limited commands (3,S)

Programs are made by assembling instructions from a limited set. (1,S)

Key

Unplugged	Necessary
Plugged	Helpful

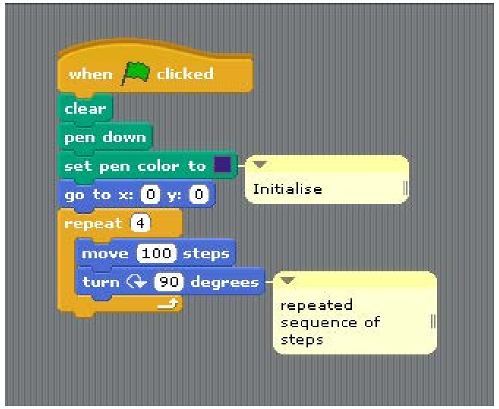
Consensus goal (n, S/T)
n = number of concepts coalesced
S = evidence that students able to engage with the learning goal
T= theory based goal

(Rich et al. 2017)

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Repetition - progression?

Repeat n times

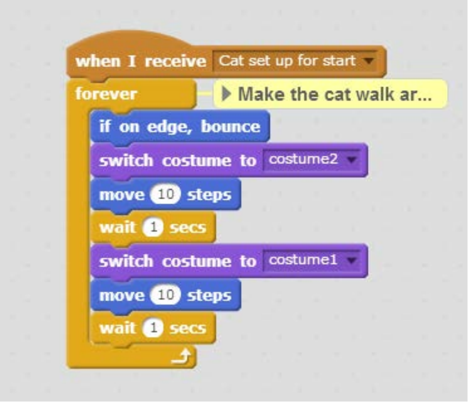


```
when [flag] clicked
clear
pen down
set pen color to [ ]
go to x: 0 y: 0
repeat 4
    move 100 steps
    turn 90 degrees
```

Initialise

repeated sequence of steps

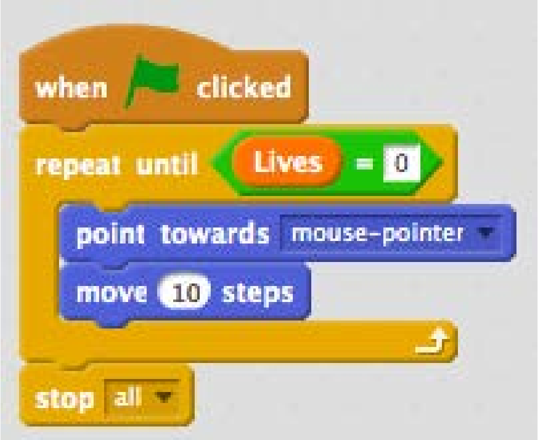
Forever (can be simple or complex - depends on use - spiral progression)



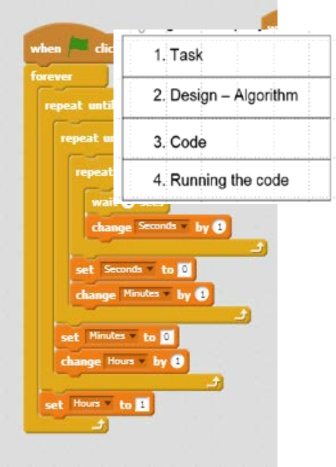
```
when I receive [Cat set up for start]
forever
    if on edge, bounce
    switch costume to [costume2]
    move 10 steps
    wait 1 secs
    switch costume to [costume1]
    move 10 steps
    wait 1 secs
```

Make the cat walk ar...

Repeat until (requires boolean and could use input)



```
when [flag] clicked
repeat until <(Lives) = 0>
    point towards [mouse-pointer]
    move 10 steps
stop [all]
```



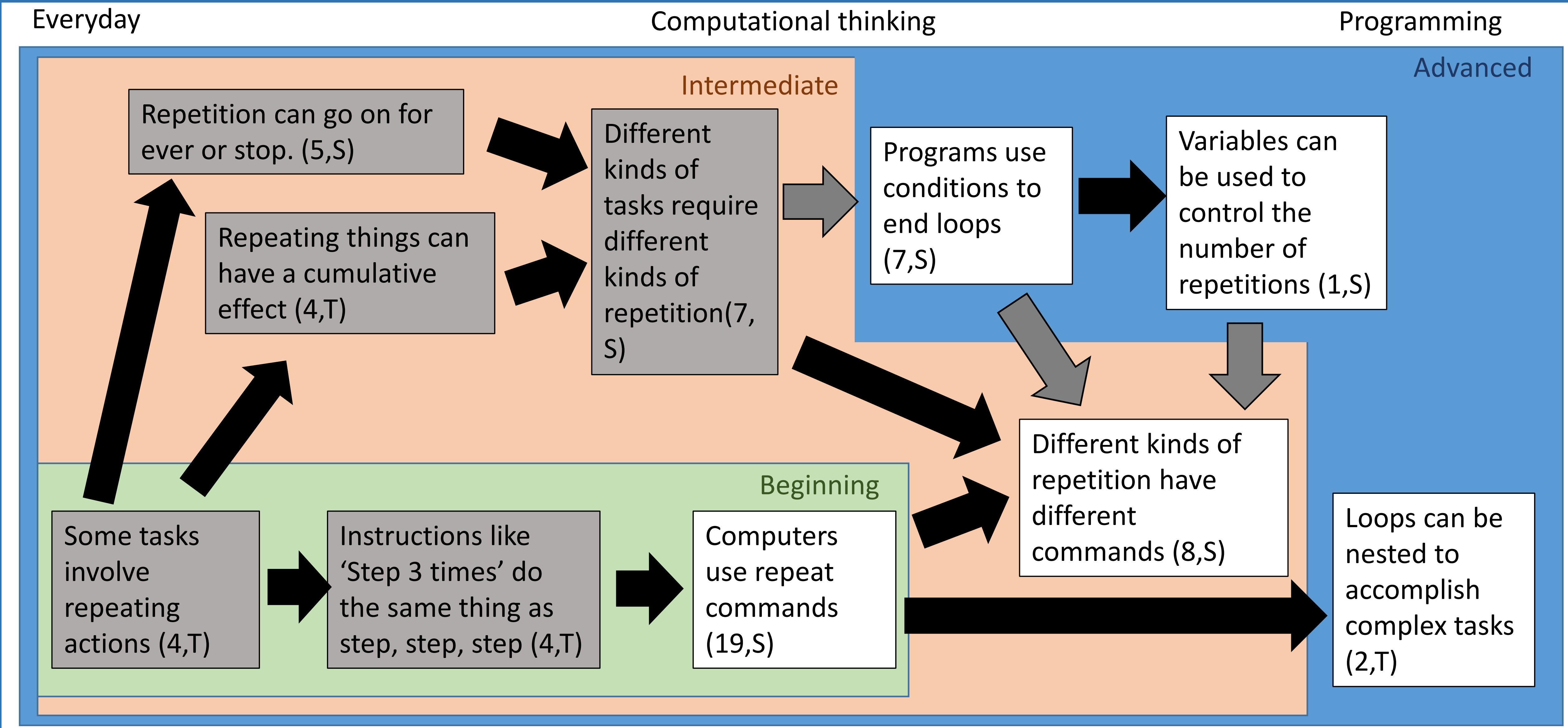
```
when [flag] clic
forever
    repeat unti
        repeat u
            repeat
                wait
                change [Seconds] by 1
            set [Seconds] to 0
            change [Minutes] by 1
        set [Minutes] to 0
        change [Hours] by 1
    set [Hours] to 1
```

1. Task
2. Design – Algorithm
3. Code
4. Running the code

Use design annotations rather than copy code to scaffold independence and ragging

In other programming languages...

for loops / while loop



Everyday

Computational thinking

Programming

Advanced

Intermediate

Beginning

Some tasks involve repeating actions (4,T)

Instructions like ‘Step 3 times’ do the same thing as step, step, step (4,T)

Computers use repeat commands (19,S)

Repetition can go on for ever or stop. (5,S)

Repeating things can have a cumulative effect (4,T)

Different kinds of tasks require different kinds of repetition(7, S)

Programs use conditions to end loops (7,S)

Variables can be used to control the number of repetitions (1,S)

Different kinds of repetition have different commands (8,S)

Loops can be nested to accomplish complex tasks (2,T)

(Rich et al. 2017)

Key

▨ Unplugged

□ Plugged

Necessary

Helpful

Consensus goal (n, S/T)
n = number of concepts coalesced
S = evidence that students able to engage with the learning goal
T= theory based goal

Queen Mary University of London

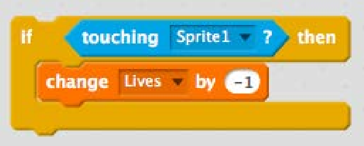
SUPPORTED BY MAYOR OF LONDON

COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE

KING'S College LONDON

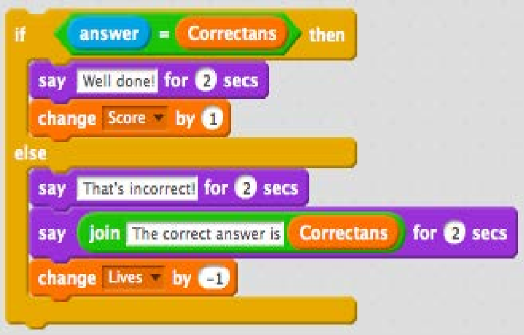
Selection - progression?

If then



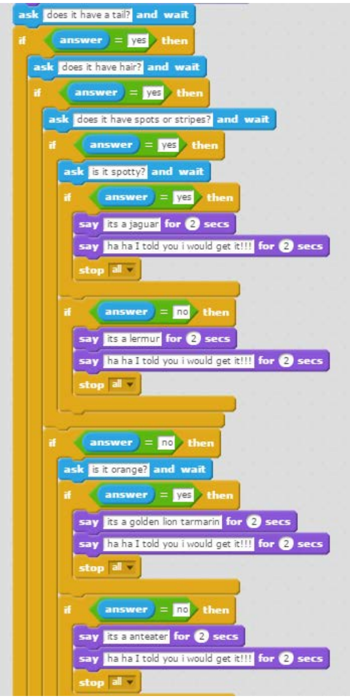
```
if <touching Sprite1 ?> then
    change Lives by -1
```

If then else



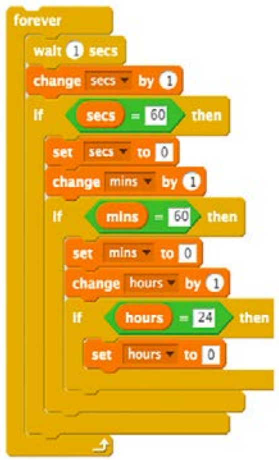
```
if <answer = Correctans> then
    say Well done! for 2 secs
    change Score by 1
else
    say That's incorrect! for 2 secs
    say join The correct answer is Correctans for 2 secs
    change Lives by -1
```

Nested

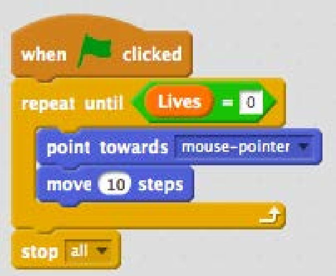


```
ask does it have a tail? and wait
if <answer = yes> then
    ask does it have hair? and wait
    if <answer = yes> then
        ask does it have spots or stripes? and wait
        if <answer = yes> then
            ask is it spotty? and wait
            if <answer = yes> then
                say its a jaguar for 2 secs
                say ha ha I told you i would get it!!! for 2 secs
                stop all
            if <answer = no> then
                say its a lermur for 2 secs
                say ha ha I told you i would get it!!! for 2 secs
                stop all
        if <answer = no> then
            ask is it orange? and wait
            if <answer = yes> then
                say its a golden lion tarmarin for 2 secs
                say ha ha I told you i would get it!!! for 2 secs
                stop all
            if <answer = no> then
                say its a anteater for 2 secs
                say ha ha I told you i would get it!!! for 2 secs
                stop all
```

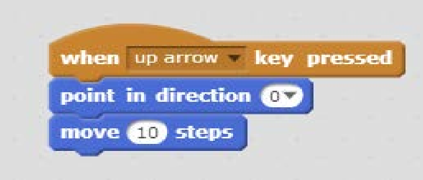
Event handling/ Forever/ Conditional loops?



```
forever
    wait 1 secs
    change secs by 1
    if <secs = 60> then
        set secs to 0
        change mins by 1
        if <mins = 60> then
            set mins to 0
            change hours by 1
            if <hours = 24> then
                set hours to 0
```



```
when green flag clicked
repeat until <Lives = 0>
    point towards mouse-pointer
    move 10 steps
stop all
```



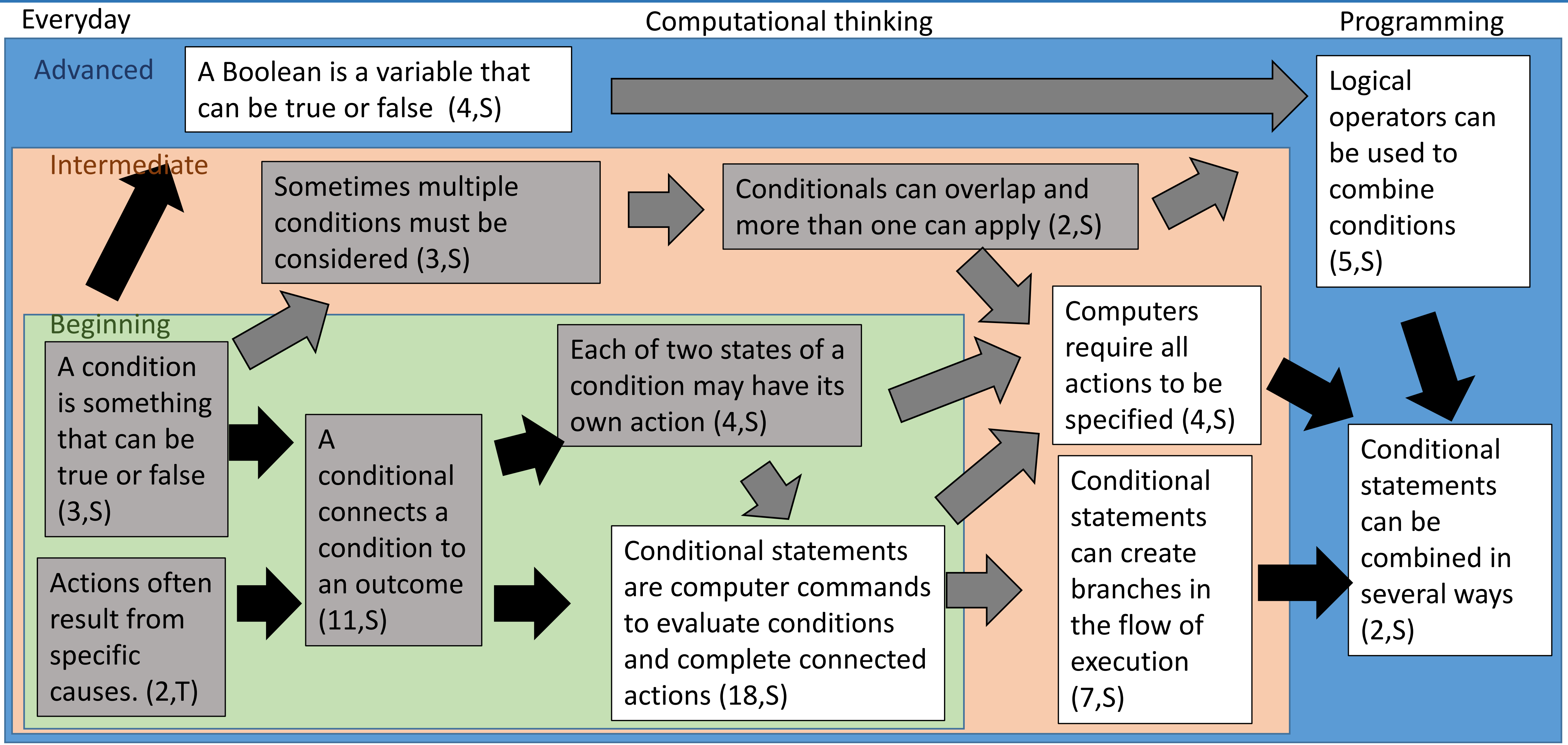
```
when up arrow key pressed
point in direction 0
move 10 steps
```

1. Task
2. Design – Algorithm
3. Code
4. Running the code

Use design annotations rather than copy code to scaffold independence and ragging

In other programming languages...

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON



Everyday

Computational thinking

Programming

Advanced

A Boolean is a variable that can be true or false (4,S)

Logical operators can be used to combine conditions (5,S)

Intermediate

Sometimes multiple conditions must be considered (3,S)

Conditionals can overlap and more than one can apply (2,S)

Computers require all actions to be specified (4,S)

Conditional statements can create branches in the flow of execution (7,S)

Conditional statements can be combined in several ways (2,S)

Beginning

A condition is something that can be true or false (3,S)

Actions often result from specific causes. (2,T)

A conditional connects a condition to an outcome (11,S)

Each of two states of a condition may have its own action (4,S)

Conditional statements are computer commands to evaluate conditions and complete connected actions (18,S)

(Rich et al. 2017)

Key

Unplugged

Plugged

Necessary

Helpful

Consensus goal (n, S/T)
n = number of concepts coalesced
S = evidence that students able to engage with the learning goal
T= theory based goal

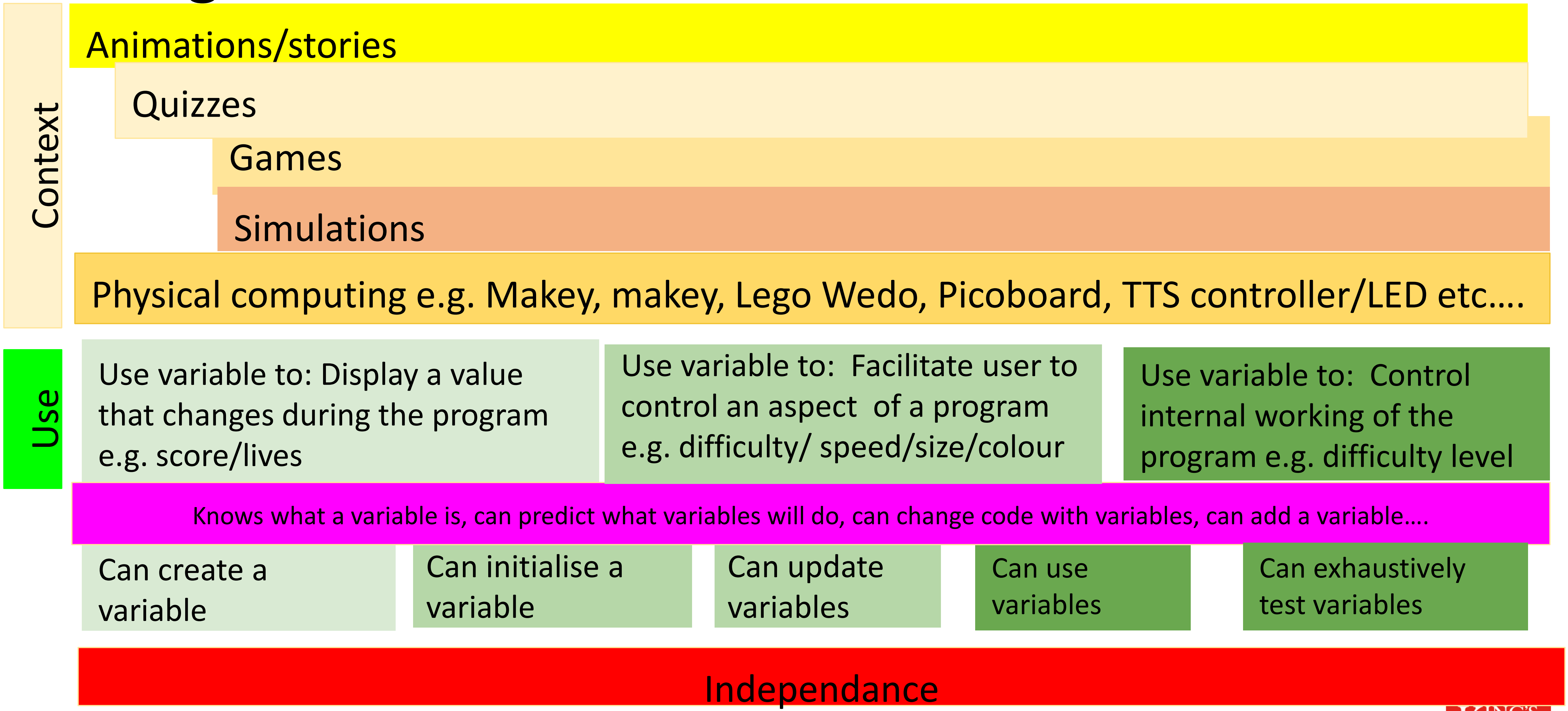
Queen Mary University of London

SUPPORTED BY MAYOR OF LONDON

COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE

KING'S College LONDON

Progression in variables (ideas to spark discussion only)



Context

Animations/stories

Quizzes

Games

Simulations

Physical computing e.g. Makey, makey, Lego Wedo, Picoboard, TTS controller/LED etc….

Use

Use variable to: Display a value that changes during the program e.g. score/lives

Use variable to: Facilitate user to control an aspect of a program e.g. difficulty/ speed/size/colour

Use variable to: Control internal working of the program e.g. difficulty level

Knows what a variable is, can predict what variables will do, can change code with variables, can add a variable….

Can create a variable

Can initialise a variable

Can update variables

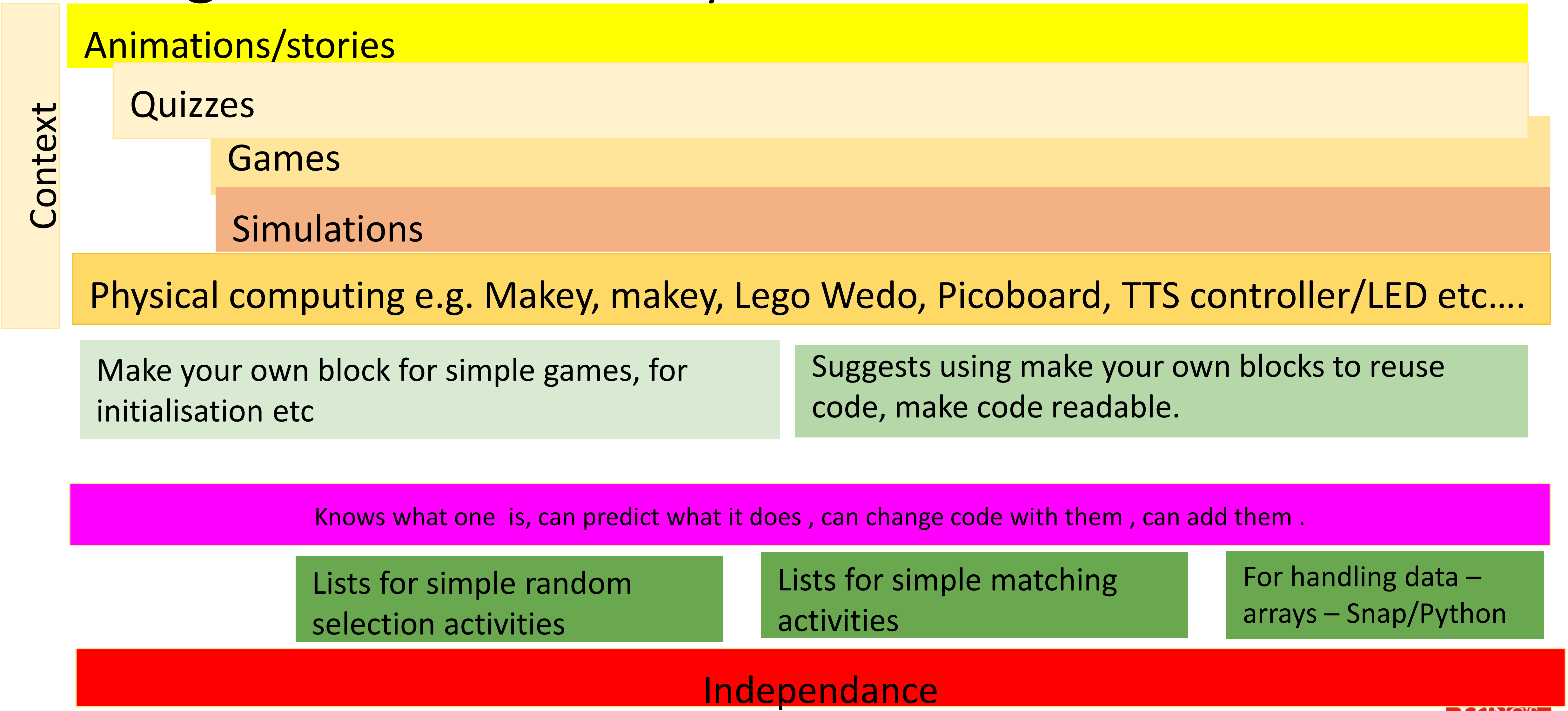
Can use variables

Can exhaustively test variables

Independance

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Progression in make your own blocks and lists



Context

Animations/stories

Quizzes

Games

Simulations

Physical computing e.g. Makey, makey, Lego Wedo, Picoboard, TTS controller/LED etc….

Make your own block for simple games, for initialisation etc

Suggests using make your own blocks to reuse code, make code readable.

Knows what one is, can predict what it does , can change code with them , can add them .

Lists for simple random selection activities

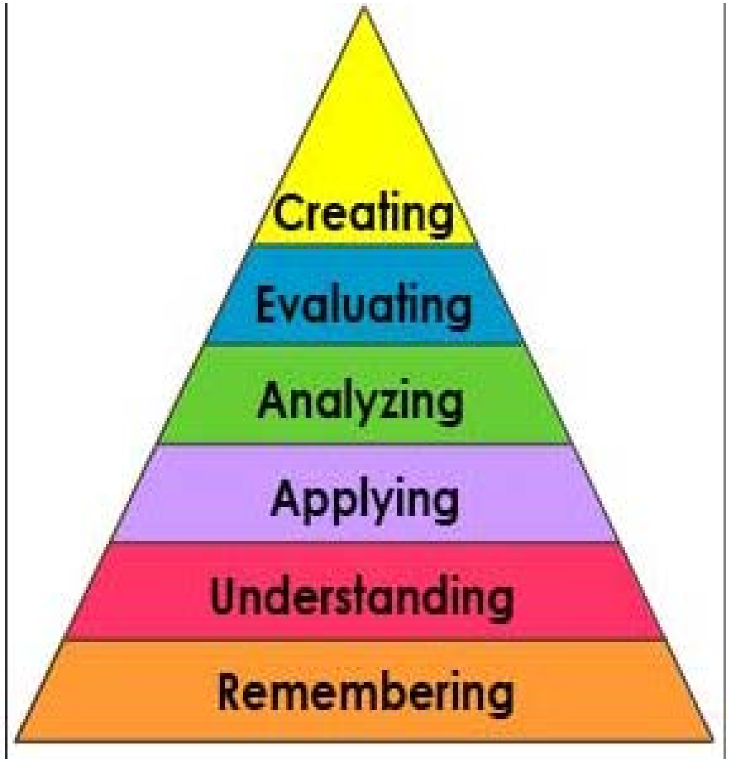
Lists for simple matching activities

For handling data – arrays – Snap/Python

Independance

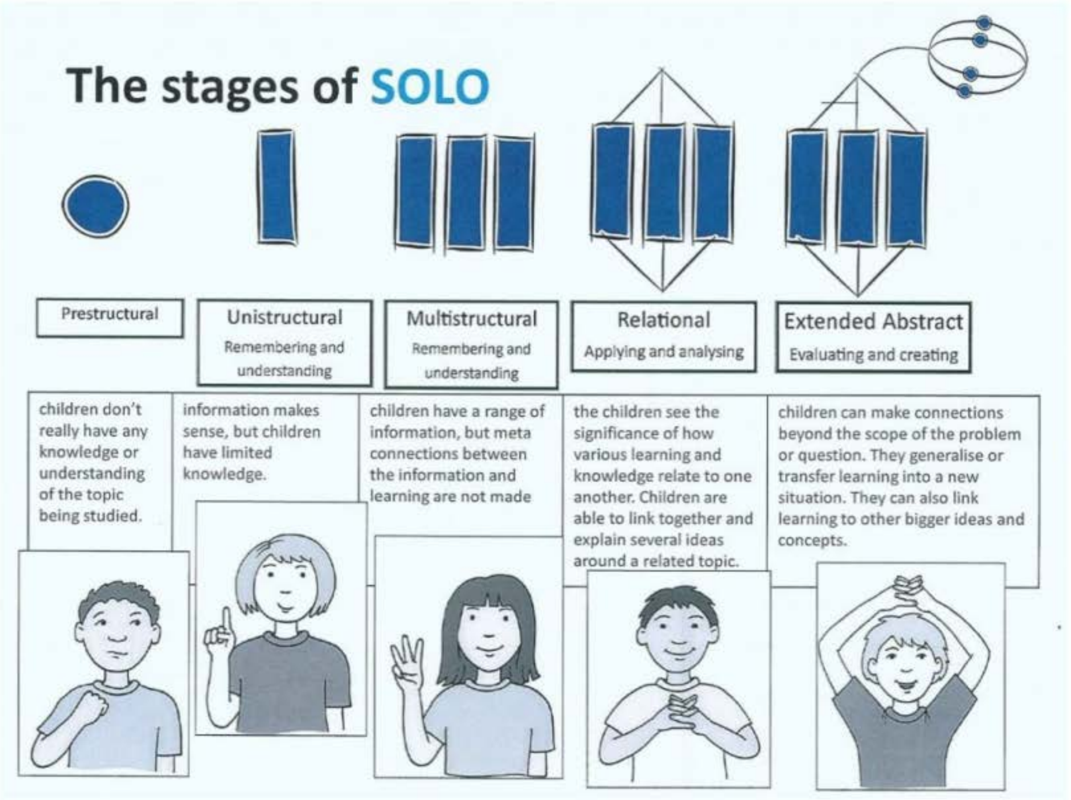
Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Progression – theory – general progression



Bloom et al 1956

Figure 1: Levels of understanding in SOLO Taxonomy



Prestructural: children don't really have any knowledge or understanding of the topic being studied.

Unistructural (Remembering and understanding): information makes sense, but children have limited knowledge.

Multistructural (Remembering and understanding): children have a range of information, but meta connections between the information and learning are not made

Relational (Applying and analysing): the children see the significance of how various learning and knowledge relate to one another. Children are able to link together and explain several ideas around a related topic.

Extended Abstract (Evaluating and creating): children can make connections beyond the scope of the problem or question. They generalise or transfer learning into a new situation. They can also link learning to other bigger ideas and concepts.

(As adapted by Severn TSA from Hook, P & Mills J. (2011) SOLO Taxonomy: A Guide for Schools Book 1 Essential Resources Educational Publishers, NZ)

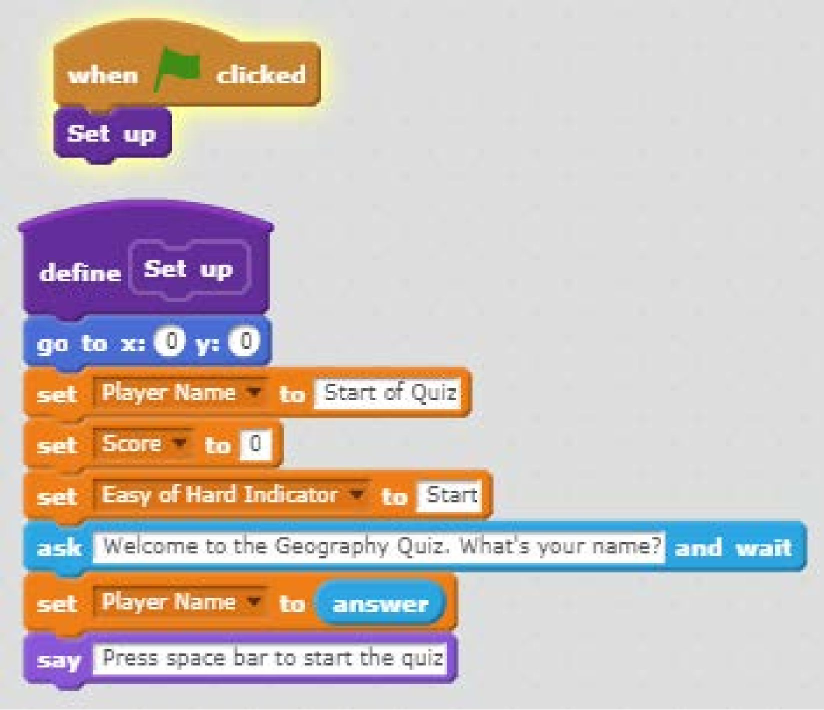
Un-progress the implementation of this design!

Gradually remove progression from the implementation of this design.
The code is shown on the next 2 pages or look online.

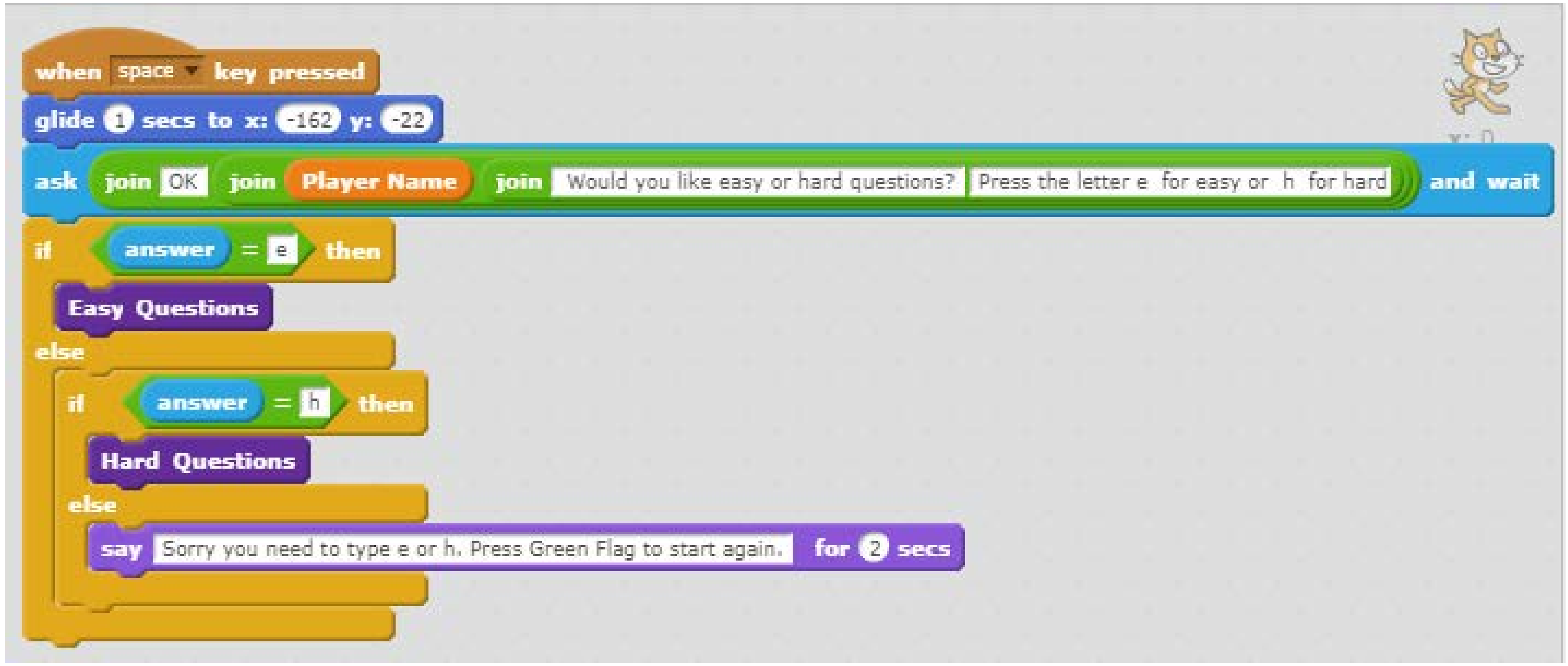
https://scratch.mit.edu/projects/207255264/

The task for this program is : Make a quiz that asks a few questions.
The design is :

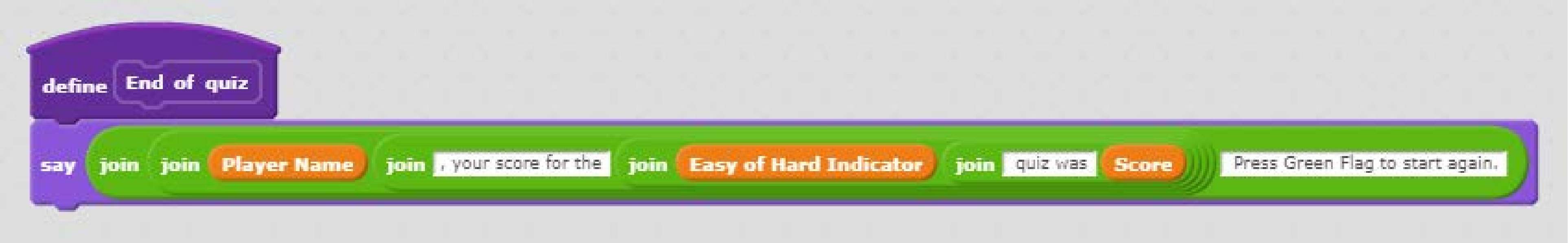
- Ask for the user name and whether they want an easy or hard quiz.
- If the say they want easy questions ask several easy questions and add 1 to the score when the player gets a correct answer but take a point away when they get it wrong. Show the correct answer if they get it right or wrong.
- Do the same for the hard questions.
- At the end, show the player how many points they got and include their name and whether they did the easy or hard quiz.



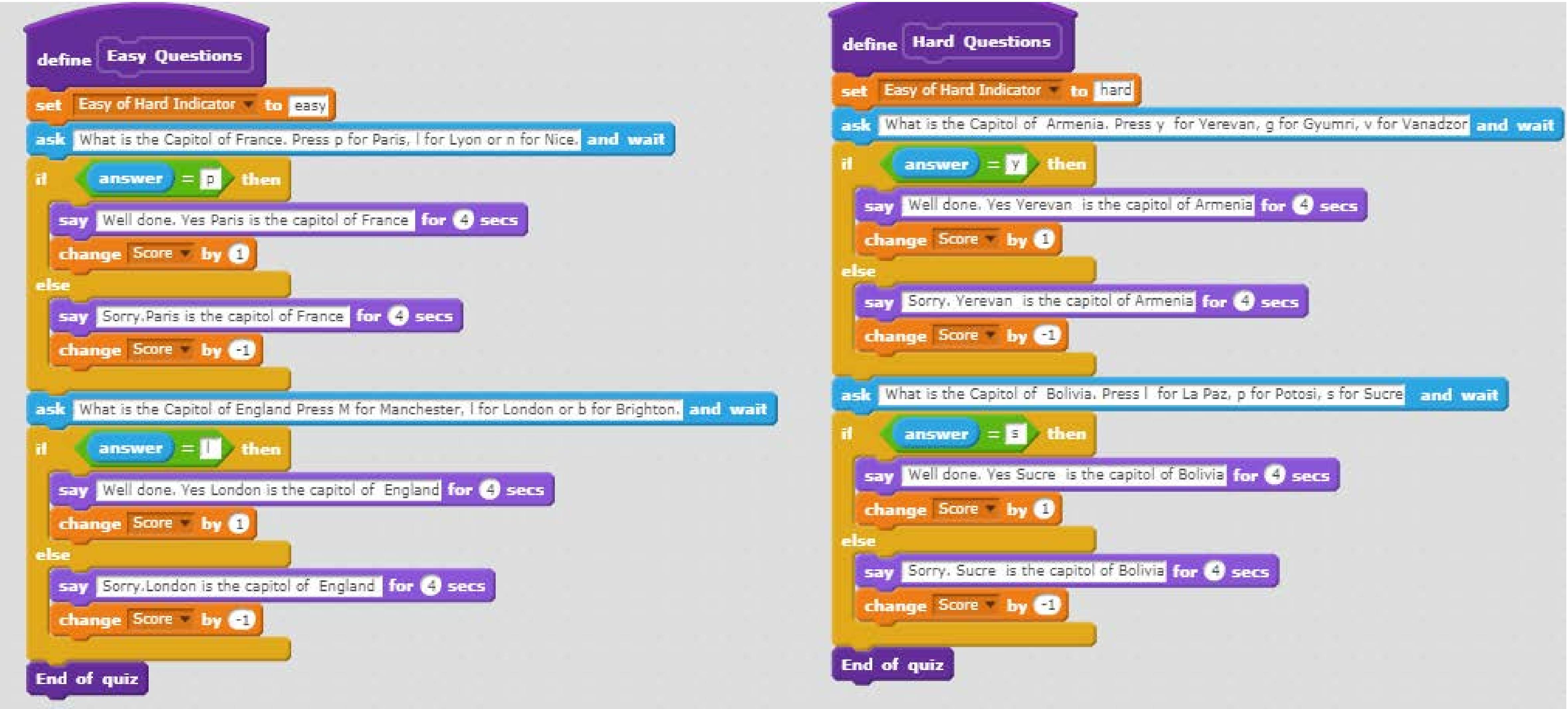
```
when ⚑ clicked
Set up

define Set up
go to x: 0 y: 0
set Player Name ▾ to Start of Quiz
set Score ▾ to 0
set Easy of Hard Indicator ▾ to Start
ask Welcome to the Geography Quiz. What's your name? and wait
set Player Name ▾ to answer
say Press space bar to start the quiz
```



```
when space ▾ key pressed
glide 1 secs to x: -162 y: -22
ask join OK join Player Name join Would you like easy or hard questions? Press the letter e for easy or h for hard and wait
if answer = e then
    Easy Questions
else
    if answer = h then
        Hard Questions
    else
        say Sorry you need to type e or h. Press Green Flag to start again. for 2 secs
```



```
define End of quiz
say join join Player Name join , your score for the join Easy of Hard Indicator join quiz was Score Press Green Flag to start again.
```



```
define Easy Questions
set Easy of Hard Indicator to easy
ask What is the Capitol of France. Press p for Paris, l for Lyon or n for Nice. and wait
if answer = p then
    say Well done. Yes Paris is the capitol of France for 4 secs
    change Score by 1
else
    say Sorry.Paris is the capitol of France for 4 secs
    change Score by -1
ask What is the Capitol of England Press M for Manchester, l for London or b for Brighton. and wait
if answer = l then
    say Well done. Yes London is the capitol of  England for 4 secs
    change Score by 1
else
    say Sorry.London is the capitol of  England for 4 secs
    change Score by -1
End of quiz
```

```
define Hard Questions
set Easy of Hard Indicator to hard
ask What is the Capitol of  Armenia. Press y  for Yerevan, g for Gyumri, v for Vanadzor and wait
if answer = y then
    say Well done. Yes Yerevan  is the capitol of Armenia for 4 secs
    change Score by 1
else
    say Sorry. Yerevan  is the capitol of Armenia for 4 secs
    change Score by -1
ask What is the Capitol of  Bolivia. Press l  for La Paz, p for Potosi, s for Sucre and wait
if answer = s then
    say Well done. Yes Sucre  is the capitol of Bolivia for 4 secs
    change Score by 1
else
    say Sorry. Sucre  is the capitol of Bolivia for 4 secs
    change Score by -1
End of quiz
```

Objectives of this session

Increase understanding of planning particularly looking at

RAG

- What influences us when we set lesson plan objectives
- What other countries are including in emerging curricula
- Scaffolding approaches incorporated in my own planning
- Levels of abstraction incorporated in my own planning
- Detailed progression of objectives in my own planning

Lesson plan objectives influenced by

Or we find a great activity and see what it might teach?

Program of study requirements (high level content)	• Sequence, selection, repetition etc. Computational thinking. Design and write programs.
How children learn to program (theories and models)	• **Level of abstraction ,** Computational Thinking, Progression!!
How we teach programming (approaches & techniques)	• **Scaffolding Copy code-> tinkering.** Pupil contribution. Genre e.g. route based, animation, quiz, game, physical computing, simulation (e.g. Maths modelling).
Specific children's needs	• SEND, gender, cultural, access to kit at home, previous knowledge and experience (ready to learn), interests etc.
Software and hardware available or chosen	• Block based, text or hybrid, tablets or PCs, reliability of kit, ease of installing software
Topic chosen (context)	• Cross-curricula or one-off.
Teacher expertise & CONFIDENCE	

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

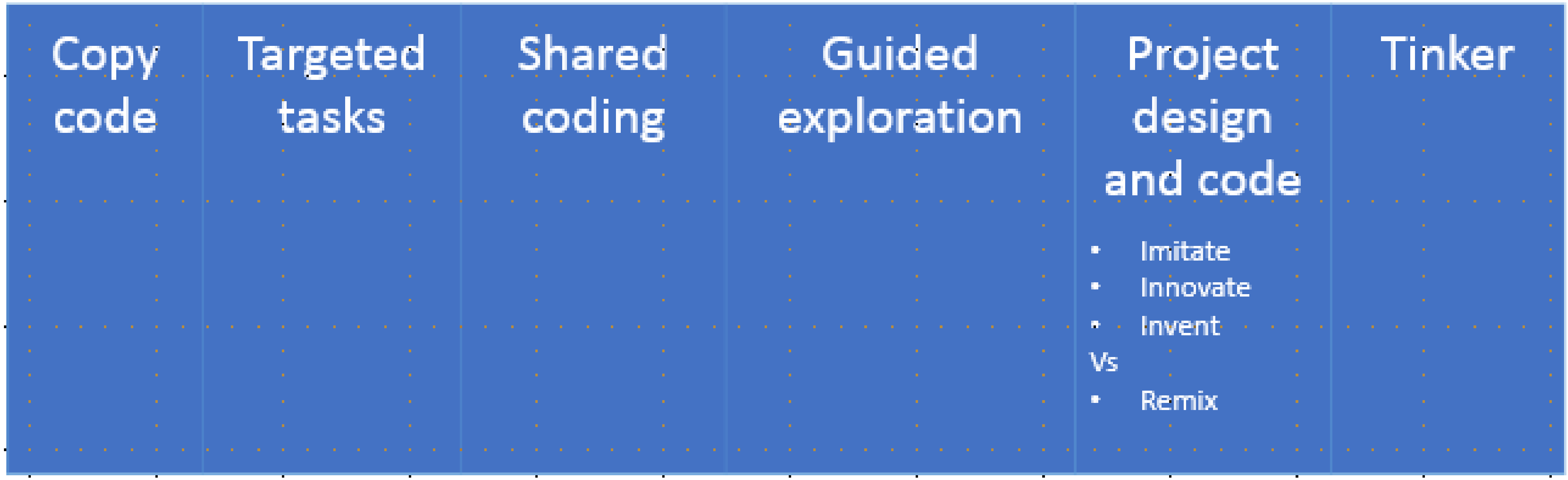
Reviewing planning

1. Review your planning for Scaffolding approaches (20 mins)
2. Review your planning for Design and LOA (20 mins)
3. Review your planning for objectives using the learning trajectories we created this morning(20 mins)

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON

Review planning

1. Annotate your planning with the approaches used e.g. copy code, targeted task etc
2. Work out what is your most common approach.
3. Work out your 'scaffolding path'.
4. Share with your team and discuss what might you change?



Copy code	Targeted tasks	Shared coding	Guided exploration	Project design and code	Tinker
				• Imitate • Innovate • Invent Vs • Remix	

Levels of abstraction

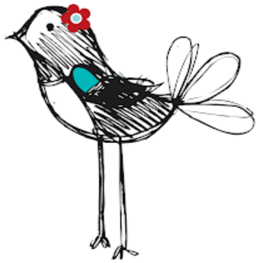
1. Task
2. Design – Algorithm
3. Code
4. Running the code

1. Annotate you planning with level of abstraction each activity works at.
2. Are you mostly working at one level?
3. Work out what is your most common approach.
4. Work out your level of abstraction path.
5. Share with your team and discuss what might you change?

Armoni, M. 2013. On Teaching Abstraction in Computer Science to Novices.
Based on Perrenet, J. *et al*. 2005. Exploring students' understanding of the concept of algorithm: levels of abstraction.
Design added and numbering changed to Cutts format by Waite et al 2016.

Compare to your SOW

1. USA - Massachusetts
2. Scotland
3. USA CSTA
4. Australian
5. Or the learning trajectories we created



1. Annotate your planning with objectives you might magpie
2. Annotate your planning with ideas you might magpie.
3. Make a list of things you have but are missing from the other curricula
4. Share with your team and discuss what might you change?

Queen Mary University of London
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL EDUCATE · ENGAGE · ENCOURAGE
KING'S College LONDON