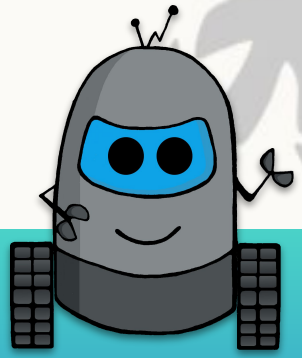


Play Code Learn

DINOSAUR STEPS

Lesson Three:

Debugging

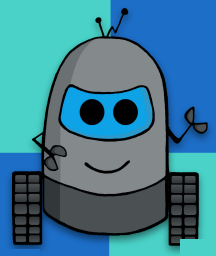
Lesson Three Learning Outcomes

Learning Intention:

...how to debug.

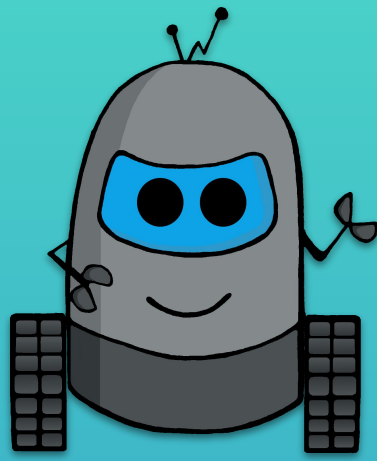
To learn how to spot errors in algorithms and patterns.

To learn how to solve problems and debug.



To understand why it is important to find bugs.

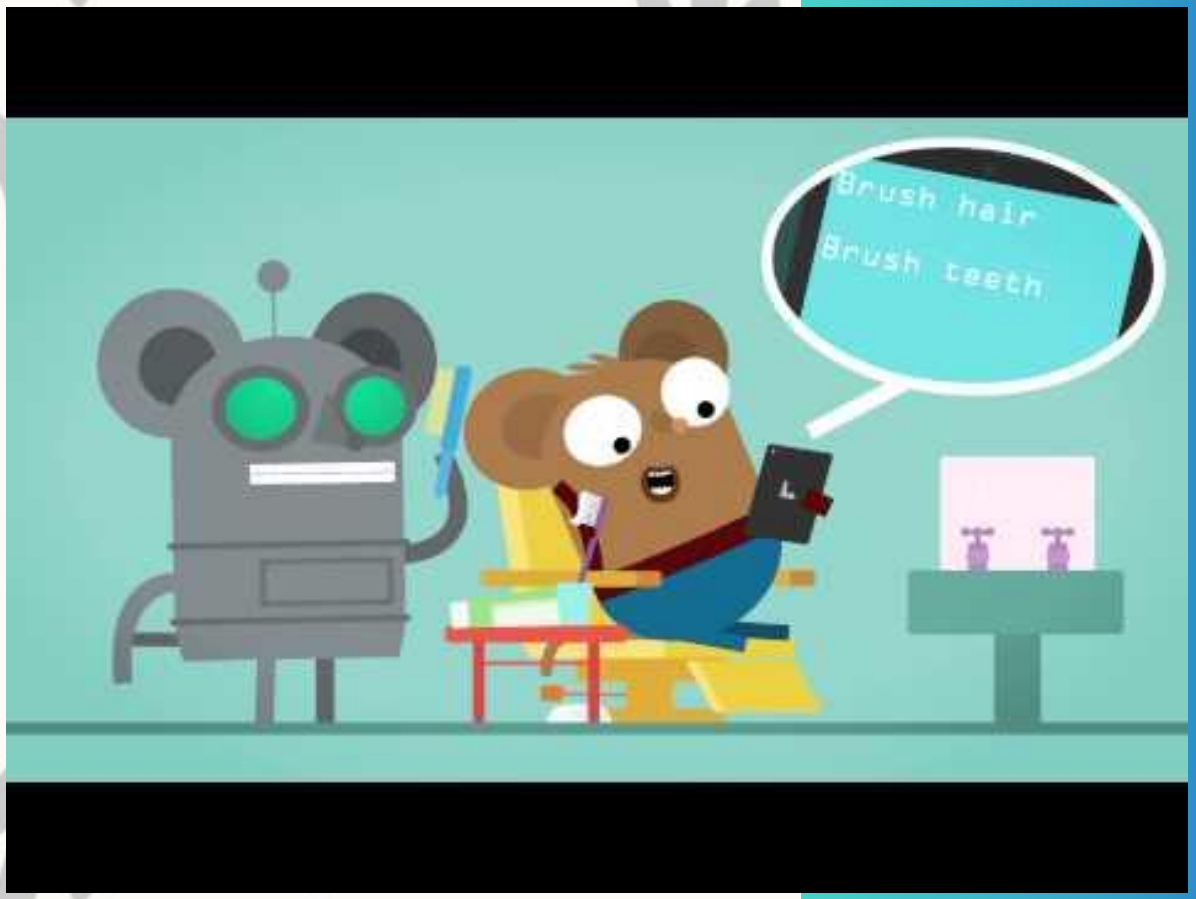
To develop a process to find bugs.



What are errors?

Discussion: What is an error or a bug?

Watch the video.



https://www.youtube.com/watch?v=CGPjraqX_ac

*What is a **mistake**?*

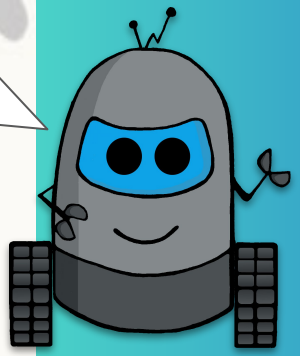
*What is a **bug**?*

Discussion: What is an error or a bug?

Think about the activity you did last lesson when you wrote an algorithm.

Did you have any '**bugs**' in the instructions?

What did you do when your found a mistake?



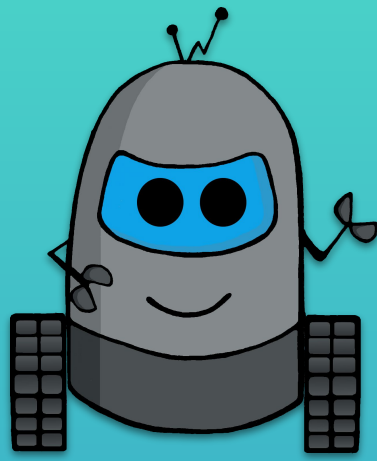
Discussion: Types of bugs

There are different types of mistakes that can be made when programming. Such as:

Syntax bugs - this is where there is an error in the writing of the instructions. A spelling mistake, or a missing full stop are examples syntax bugs. These can cause a program to 'crash' (stop working).

Logical bugs - This is a problem with the logic, like the sequence of the instructions. The program may still work but not how you wanted it to!

*Did you know that most programmers will spend more time **debugging** that writing their programs - this is where they correct any mistakes or errors!*



Debugging

Debugging

Discuss: Spot the *bug* in the algorithm

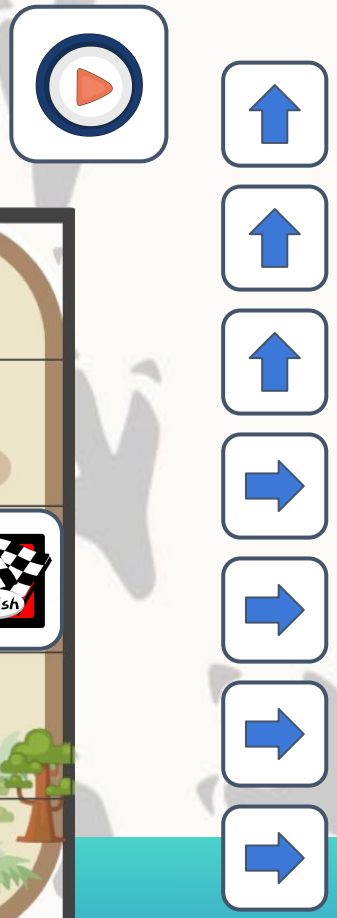
Look at the map.

Where should Explorer Ed **start** the journey?

Where should Explorer Ed **finish** the journey?

Does the algorithm get Explorer Ed from the start to the finish?

Why?



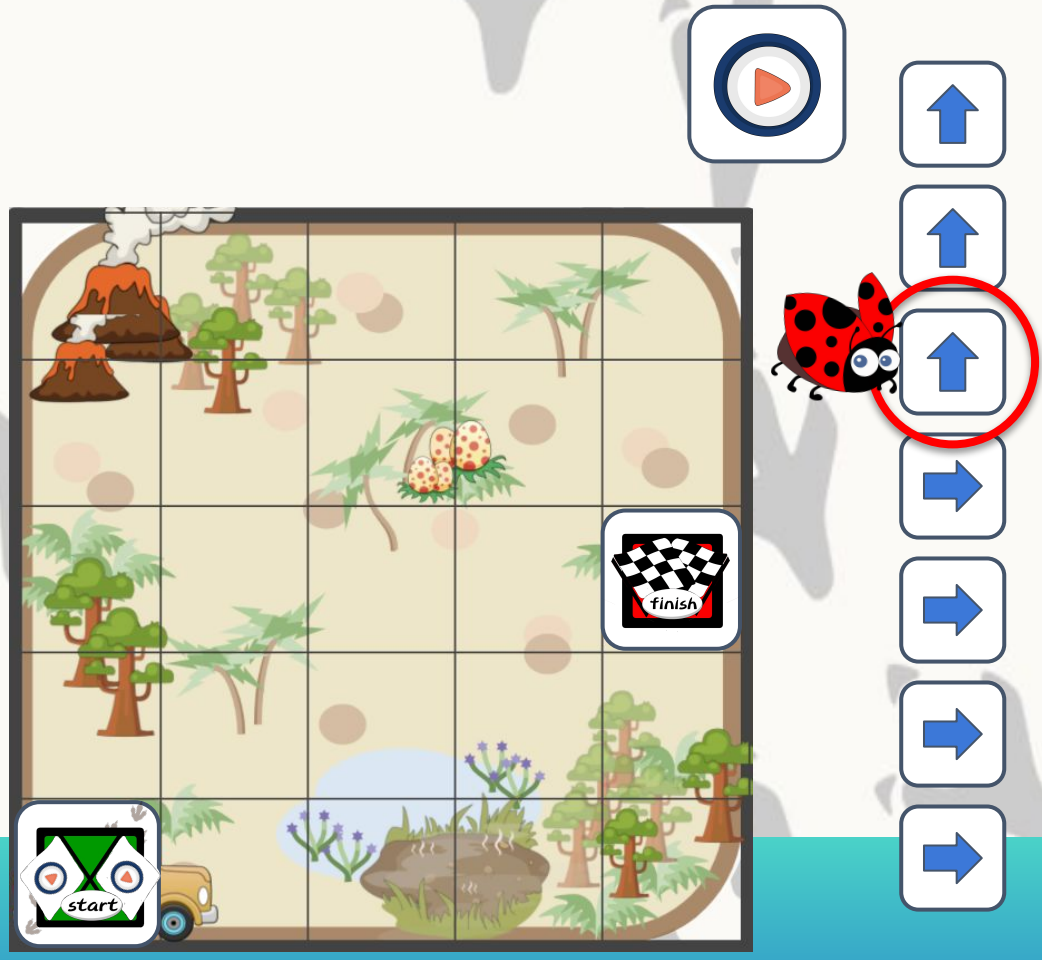
Debugging

Discuss: Spot the bug in the algorithm

Did you spot the **bug**?

Yes - there was an extra up arrow!

What should the correct algorithm be?



Debugging

Discuss: Spot the bug in the algorithm

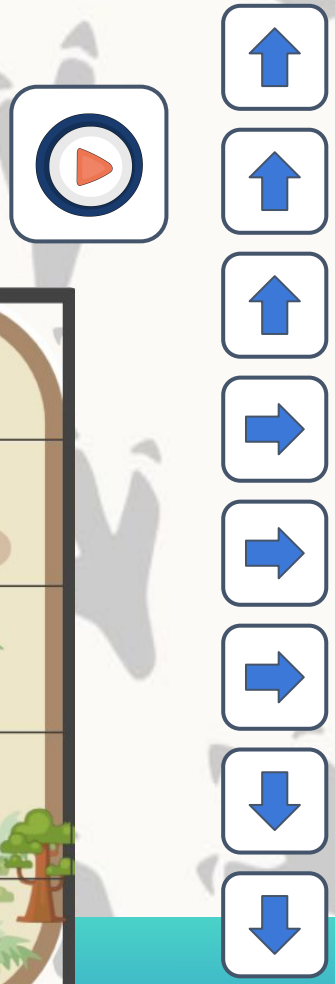
Let's have another go!

This time Explorer Ed needs to visit *Flappy* the Pterodactyl on the way.

*Is there a **bug**?*

If so, where is it?

What should the algorithm be?

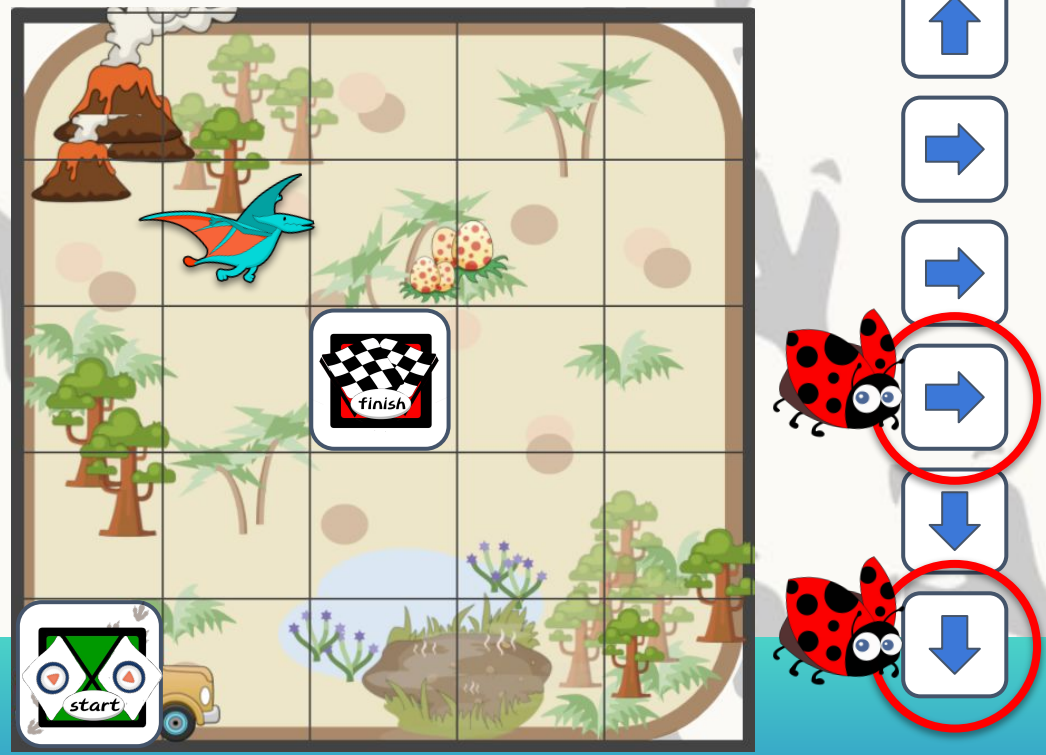


Debugging

Discuss: **Spot the bug in the algorithm**

What happened this time?

*Did you find two **bugs**?*



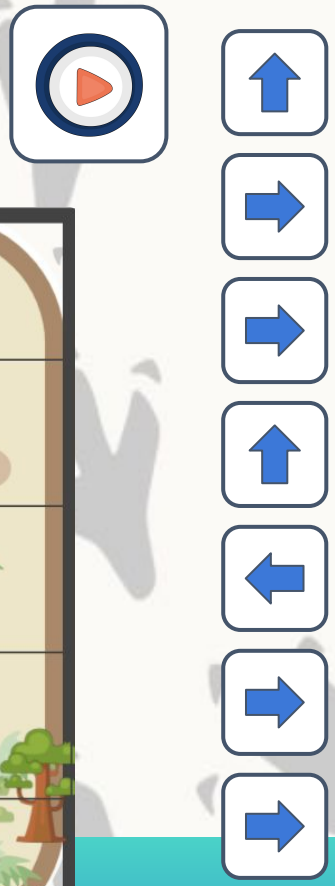
Debugging

Discuss: Spot the bug in the algorithm

Let's have another go!

This time Explorer Ed needs to visit *Flipper* the plesiosaur and *Flappy* the pterodactyl, on the way.

***Debug** the algorithm!*

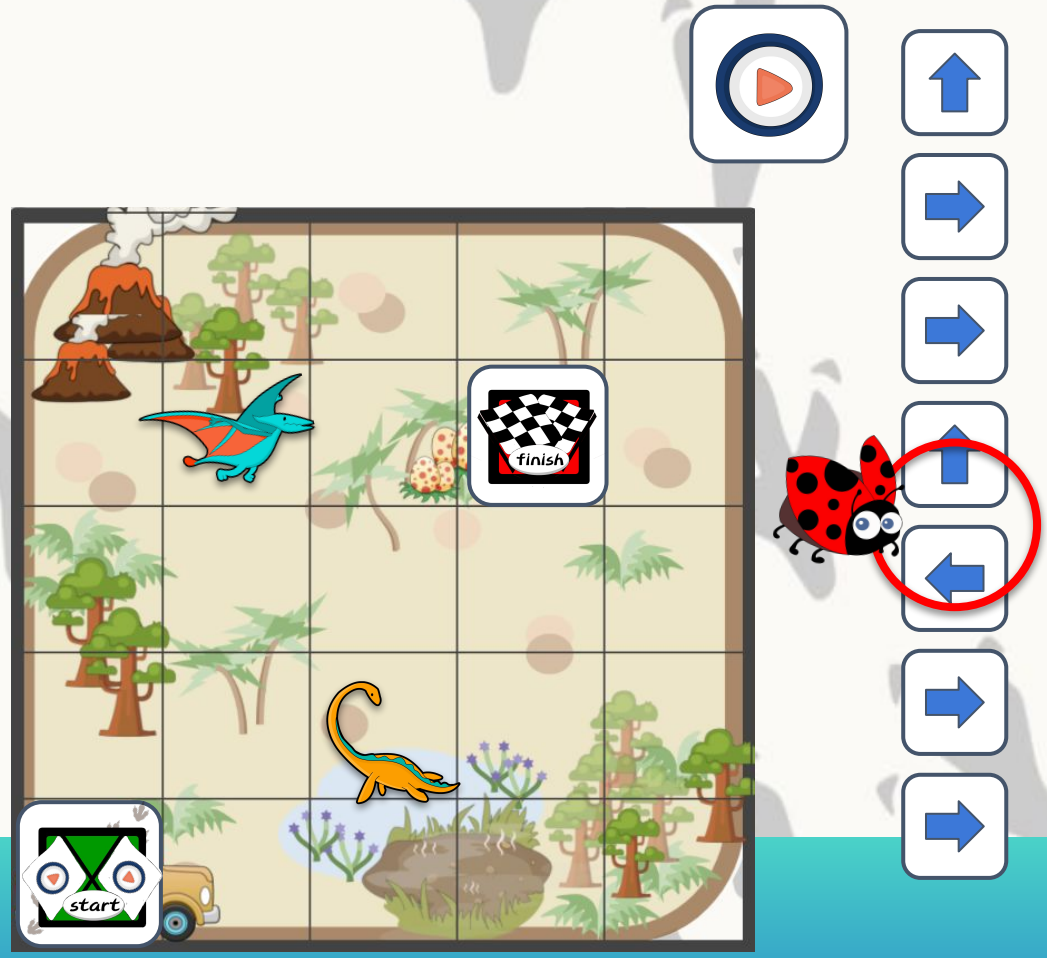


Debugging

Discuss: Spot the bug in the algorithm

Did you notice that this time a command was missing and not added to the algorithm?

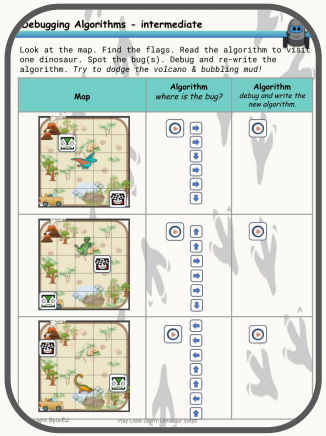
If so, well done!



Debugging

Activity: Play Code Learn: Dinosaur Steps

You need to use the Dinosaur Steps kit and the **debugging** activity handouts to have a go at reading an algorithm and spotting the **bugs**!



Once you have found an **error** or **bug** you need to **debug** the algorithm.

Test the new algorithm to make sure this one runs correctly - *be careful there may be more than one **bug**!*

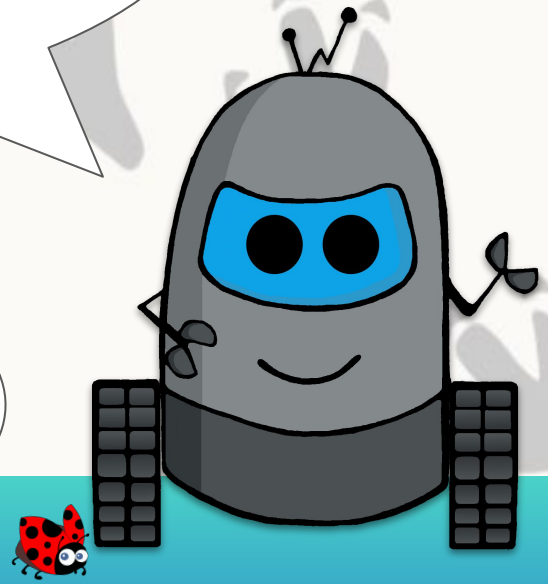
Debugging

Discussion: **The process of debugging**

*What process did you go through to find the **bug**?*

How did you spot the mistake?

I like to work through the algorithm step by step!



Debugging

Discussion: **The process of debugging**

https://www.youtube.com/watch?v=xrgbtXVVQFU

Watch the Debug IT movie.

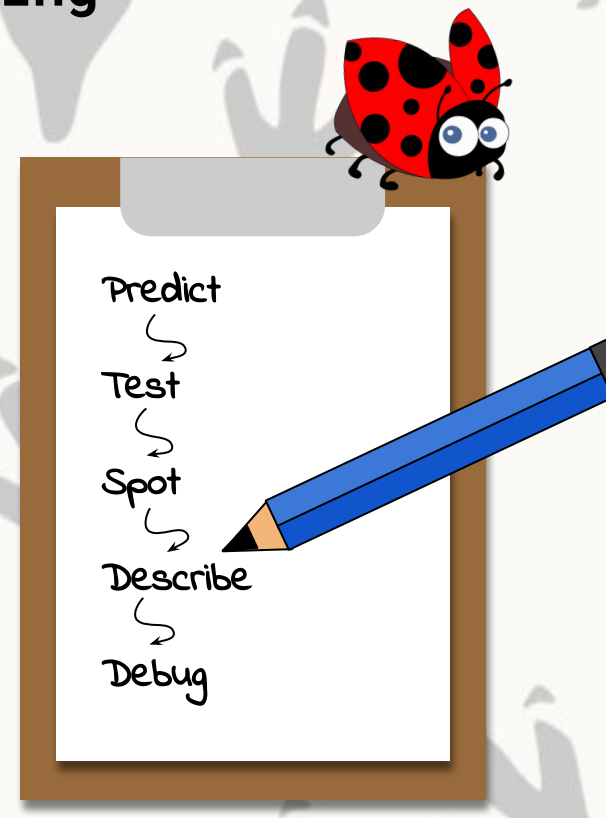
*What **debugging** process do these students use?*

Debug IT was written with primary school pupils after brainstorming ideas for an animation that would be connected with the computing curriculum.

Debugging

Discussion: **The process of debugging**

*Did you spot the stages of **debugging**?*



1. Predict the algorithm or program - *what do you think should happen?*
2. Test the algorithm or program - *what does happen?*
3. Spot the **bug** - *where has the code gone wrong?*
4. Describe the **bug** - *what has gone wrong?*
5. **Debug**
6. Try again - *work through 1 to 5 until SUCCESS!*

Play Code Learn

Above all stay calm and keep trying!

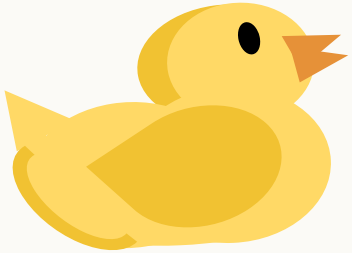
Debugging

Discussion: Spotting a bug

*But how do you spot a **bug**?*

The best way is to break down the task and read slowly through your algorithm or program step by step.

Rubber Duck debugging



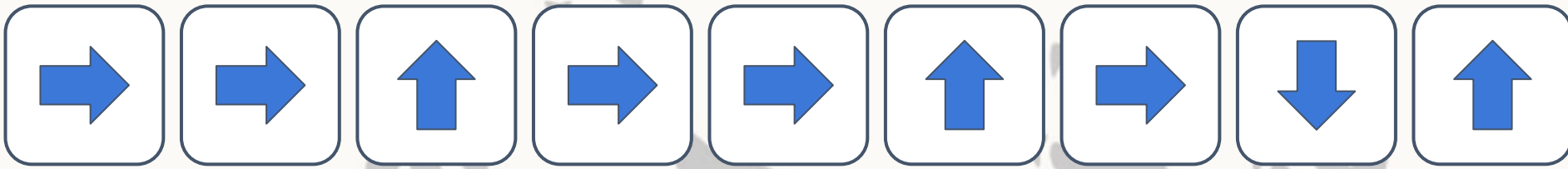
One way to find a **bug** is to use a technique called **Rubber Duck debugging**. This is where programmers talk to a rubber duck!

By reading out loud the program or algorithm slowly, programmers spot the **bugs** hidden inside their instructions. So if you need to **debug** - talk to a duck!

Debugging

Activity: Bugs in patterns

Look at the sequence of cards:



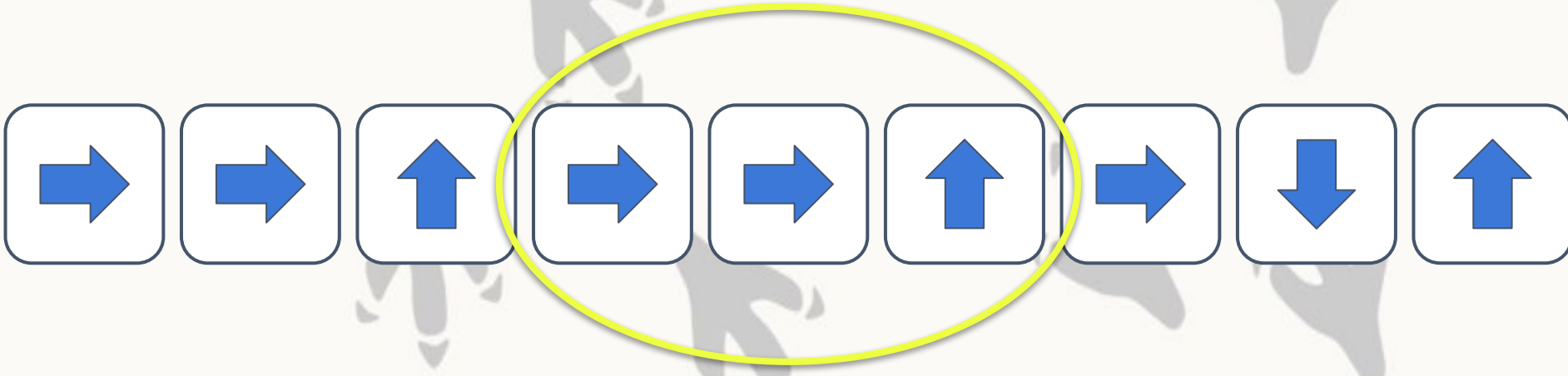
Can you find the pattern?

*Can you spot a **bug** in the pattern?*

Debugging

Activity: **Bugs in patterns**

Did you spot the pattern?



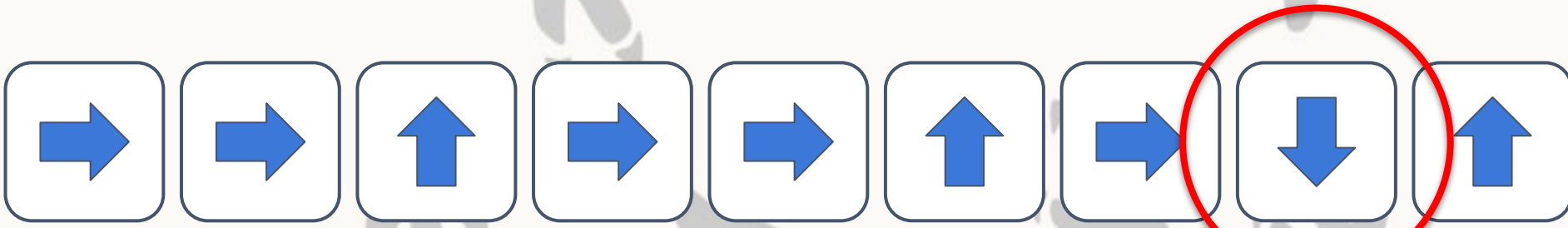
*Here is the pattern - **Right, Right, Up.***

How many times should this pattern repeat?

Debugging

Activity: **Bugs in patterns**

*Did you spot the **bug**?*



Here is is!

Debugging

Activity: Bugs in patterns

Let's try one more...

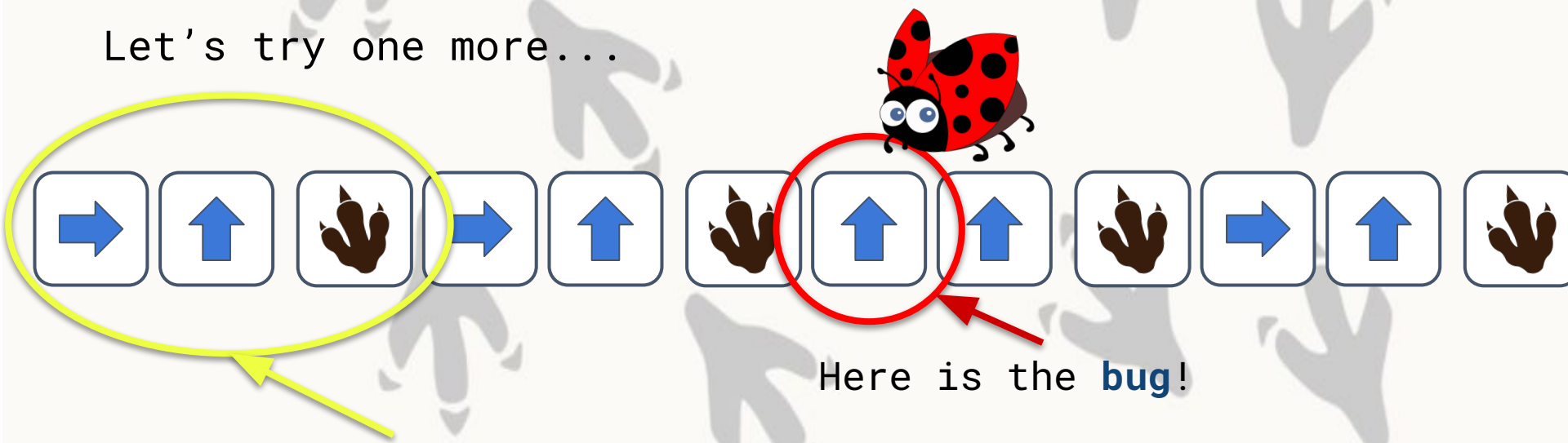
Can you find the pattern?

*Can you spot a **bug** in the pattern?*

Debugging

Activity: **Bugs in patterns**

Let's try one more...



Here is the **bug**!

The pattern is right, up, footprint.

Debugging

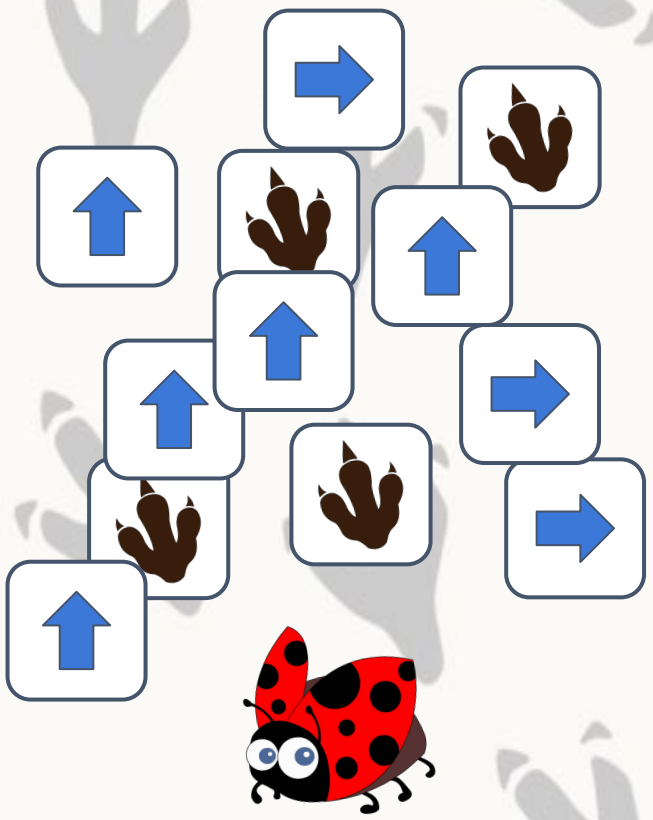
Activity: Bugs in patterns

It's your turn!

Find a partner.

One person make a pattern using the arrow and footprint cards. You may want to start by just using the arrows. Make sure there is a **bug** hidden in the pattern *(do not let your partner see the pattern or the **bug**!)*

The other person needs to find the **bug** and ***debug*** it!

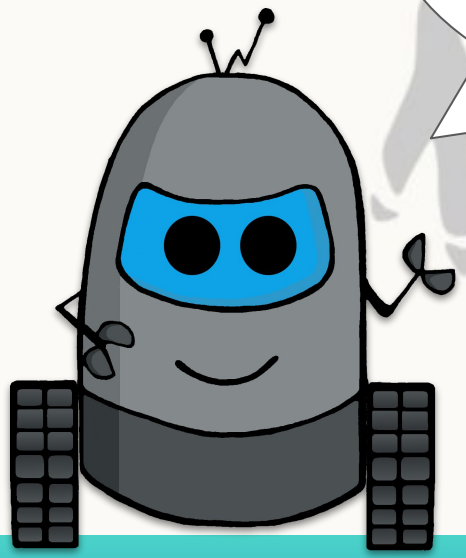


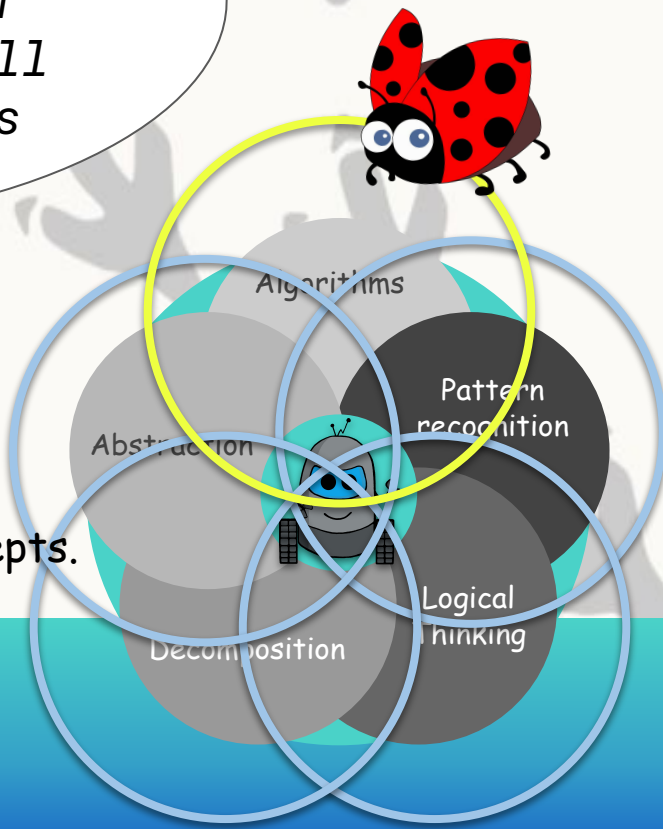
Swap and repeat as many times as you can!

Debugging

Discussion: **Debugging & Computational Thinking**

*In computational thinking, **debugging** is part of writing a successful algorithm but it can also use all of the other concepts too!*





Computational thinking concepts.

Play Code Learn

Debugging

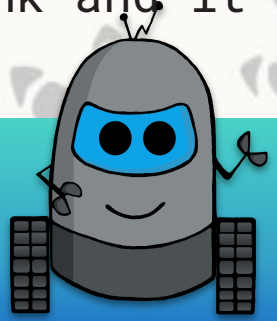
Discussion: **Why debug?**

Bugs can be annoying - they can stop a program from working properly and can be hard to find.

Why do we need to debug?

Bugs can be a challenge - but they are a good challenge!

They can help you to learn how to solve problems and how to persevere (keep trying). They help you to think and it feels great when you finally **debug**!

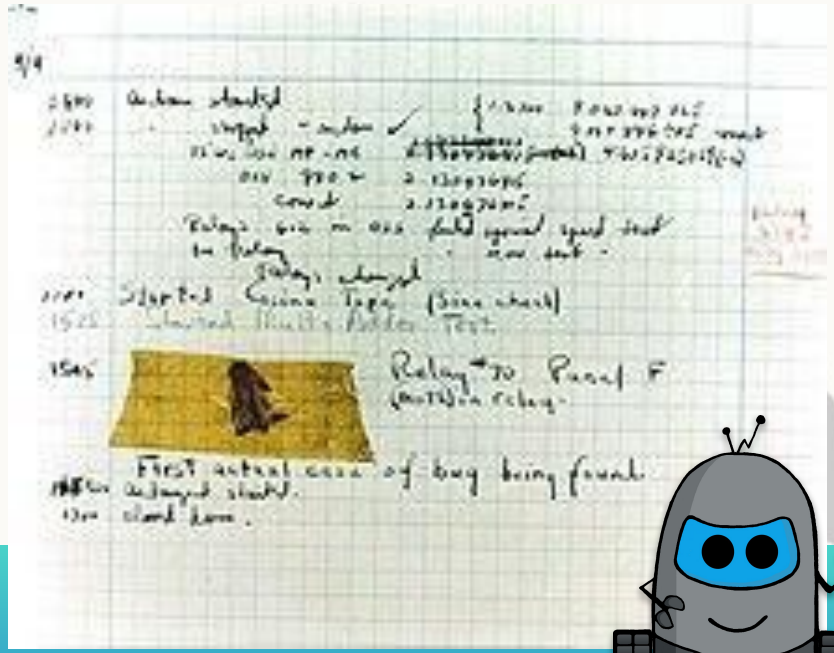


Bug facts!

Discussion: **Did you know?**

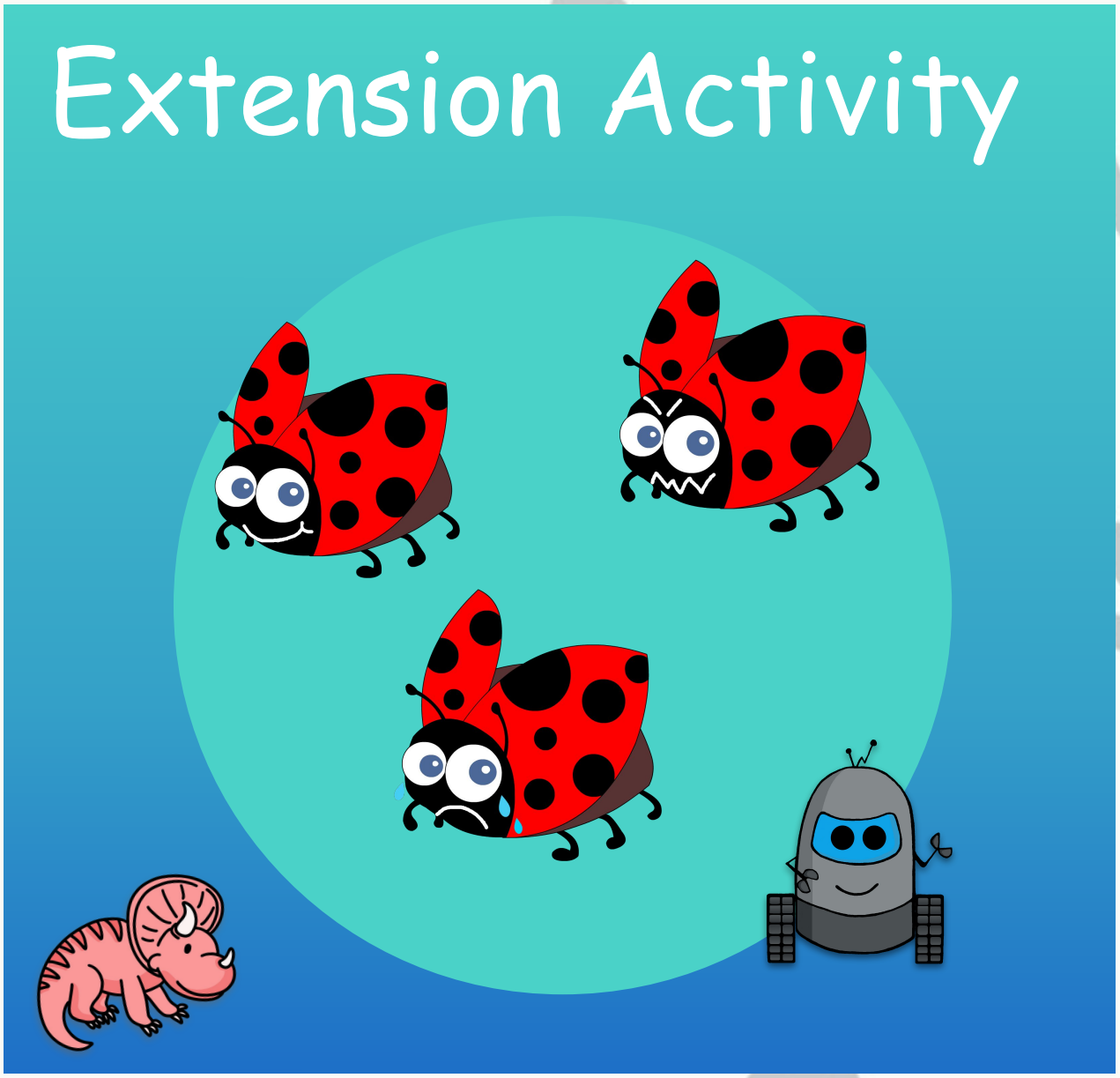
Interesting facts:

The term '**bug**' to describe defaults dates from the 1870's. In fact, Thomas Edison wrote about "**bugs**" when he was inventing.



The first ever 'real life' bug was documented as being found inside a computer in 1947!

Extension Activity



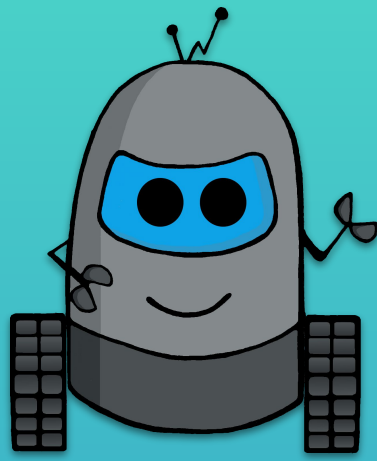
Emoji Bugs

Finding **bugs** can be a challenge.

What emotions did you go through when you:

a) Realised there was a **bug**?
b) Spotted the **bug**?
c) Found out what the **bug** did?
d) **Debugged** successfully?

Why is it important to learn about your emotions when learning?

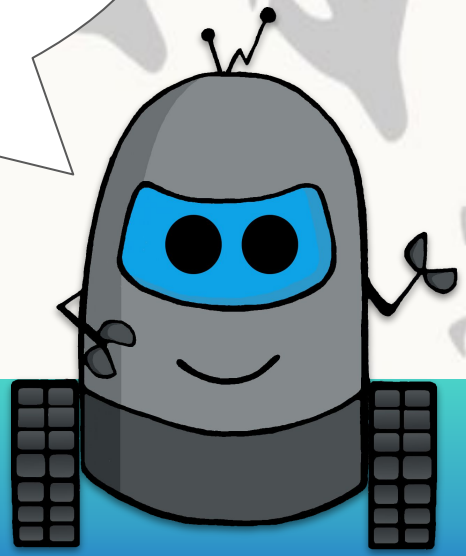


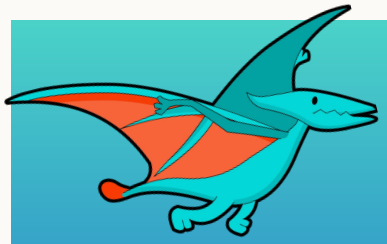
Reflection

Debugging

Discussion: **What do you think this song means?**

99 little bugs *in the code,*
99 little bugs*.*
Take one down.
Patch it around
127 little bugs *in the code...*

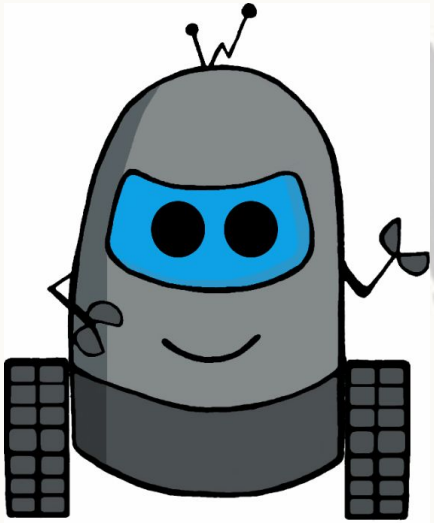




Reflection: Lesson Three

Learning Intention:

...how to debug.



How do you feel about today's lesson?



What were the key takeaways from the lesson today?

What would you like to learn more about?

Play Code Learn

Thank you!