

Coding with the Calliope mini

Student material volume 2 from grade 4



CALLIOPE

Coding with the Calliope mini

Programming in primary school

Student material volume 2 from grade 4

Authors: Michael Abend (The Calliope mini as a stopwatch, The Calliope mini as a spelling trainer, The small coding lexicon)

Kirstin Gramowski (determine neighbouring numbers with the Calliope mini, The Calliope mini as a metronome)

Lars Pelz (The Calliope mini and the Nim game, tips for your own program ideas)

Bernd Poloczek (Introduction, The most important functions of the Calliope mini)

Consultants: Michael Abend, Kirstin Gramowski, Lars Pelz, Bernd Poloczek

Editor: Johanna Ahlrichs, Kirsten Pauli, Patrizia Schwarzer

Illustration: Benedikt Beck, Nuremberg
Calliope gGmbH, Berlin: Page 5, 44 and U1, U4
Open Roberta Lab: Page 3
zweiband.media GmbH, Berlin: all vector graphics

Cover design: COSAKitchen, Corinna Babylon, Berlin

Layout, graphics and technical implementation: zweiband.media GmbH, Berlin

www.cornelsen.de

The websites of third parties, the Internet addresses of which are given in this textbook, were carefully checked before going to press. The publisher assumes no liability for the topicality and content of these pages or those linked to them.

1st. 2nd edition Print 2019

All prints in this edition are unchanged in terms of content and can be used in parallel in lessons.

Translated by Calliope gGmbH based on the original title “Coden mit dem Calliope mini” 2020 Calliope gGmbH, Berlin

CC-BY-SA 4.0 Cornelsen Verlag GmbH, Berlin 2018

This document is under a Creative Commons license.
The terms of use can be found at the end of the title.

ISBN: 978-3-06-600013-9

Printing: AZ Druck und Datentechnik GmbH, Kempten

Advice: The NEPO Editor is constantly being improved and further developed. It may happen that some blocks look a little different. However, the functions remain the same, so that you can realise all the shown exercises.

Table of Contents

Introduction	2
Coding with the Calliope mini	2
Maths	
Determine neighbouring numbers with the Calliope mini	8
Subject teaching	
The Calliope mini as a metronome	14
The Calliope mini as a stopwatch	19
English	
The Calliope mini as a spelling trainer	24
Maths	
The Calliope mini and the Nim game	32
Tips for your own programme ideas	39
The little coding dictionary	43
The most important functions of the Calliope mini	44



Task in the notebook



Task on the computer



Task with the Calliope mini

Coding with the Calliope mini



Hello! I am Lio.
Have you ever coded?

Coding is giving orders to a computer to do exactly what you want it to do.
This is also called programming.

If the computers in our world weren't programmed properly, there would be a lot of confusion. Imagine if a calculator confused minus with plus.

The Calliope mini is also a small computer. You can learn to code with him. You programme it with the editor Open Roberta Lab.



Then you can put the Calliope mini into practical use.

Lio likes the mini piano.



Successfully code in 8 steps

1. Open Internet-Browser
2. Enter <https://lab.open-roberta.org> in the address line
3. Select "Calliope mini"



Select your system. Circle it in red.



4. Create or open a user account 
5. Give each new programme a name and save it  → 



Enter your username and password here.

Username

Password

6. Code and save at regular intervals 

7. Use simulation  → 

8. Transfer the code to the Calliope mini and run the programme ▶

The editor is regularly improved. That is why a block can sometimes look different than in the workbook. My tip: take a look and try it out.

Under "Edit" → "My Programmes" you can open and edit your saved codes.



Coding knowledge

Every programme on a computer is built on the same principle. This principle can be easily seen in a pocket calculator.



Input

You type numbers in a calculator.

Processing

The calculator processes the numbers.

Output

You get the result the arithmetic problem.



1. The principle is named after its first letter.

Fill in the

- Principle

The Calliope mini also works according to this principle.

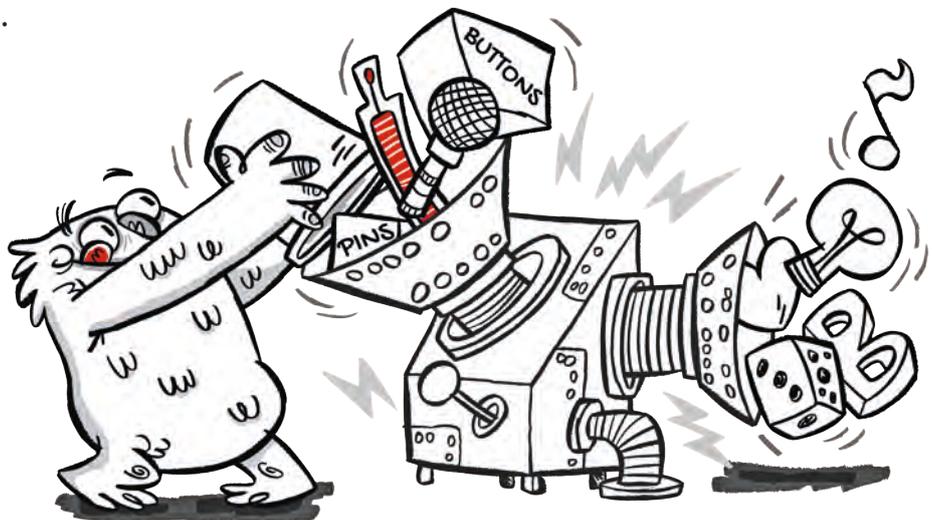


2. Colour in the picture.

Input: **green**

Processing: **yellow**

Output: **blue**

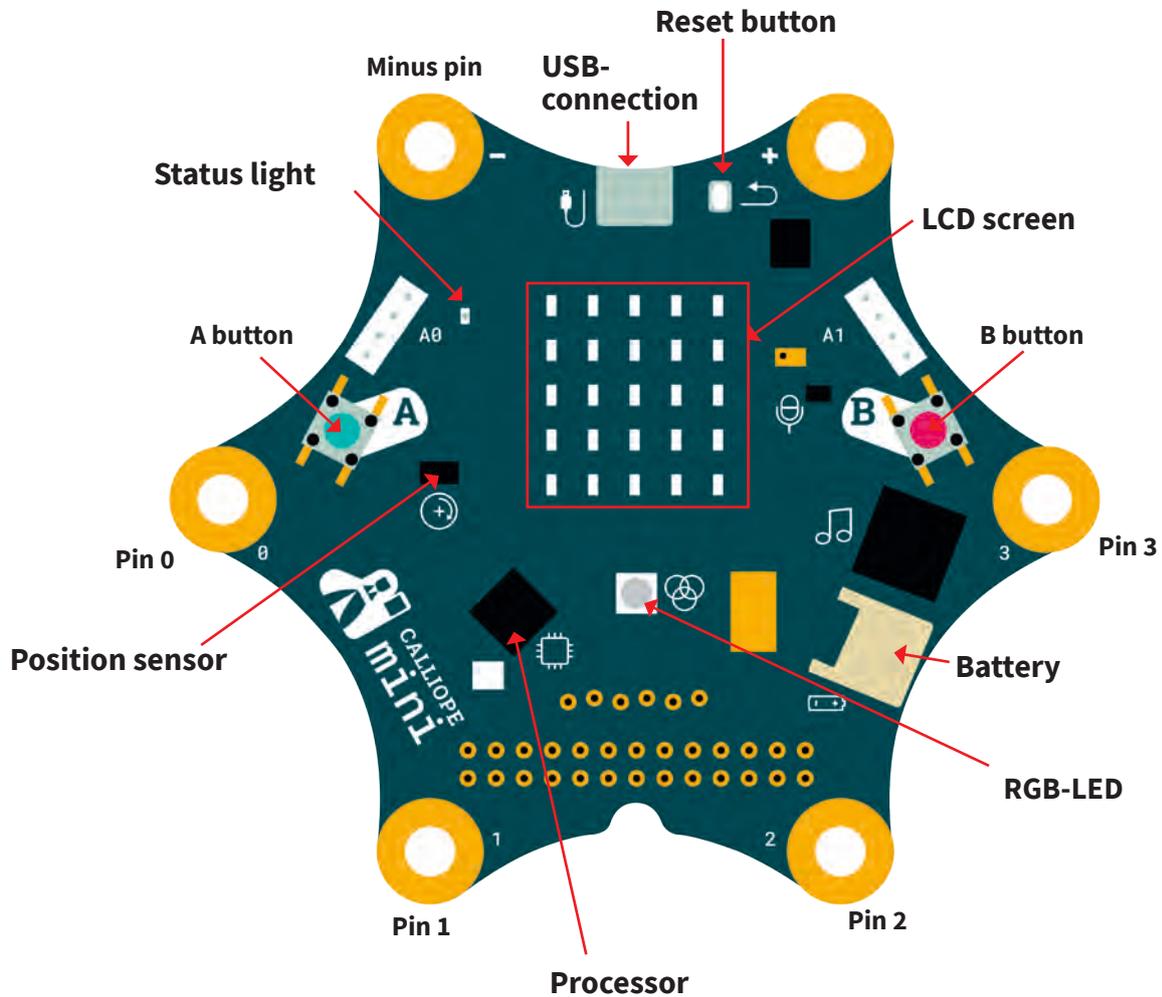


The individual parts of the Calliope mini have different tasks.



3. Circle the terms with the appropriate colours:

Input – **Processing** – **Output**.



Using the editor you can determine how the parts of the Calliope mini should be used.



4. Match the blocks with the appropriate terms. Connect.

- Action ●
- Sensors ●
- Control ●
- Logic ●
- Math ●
- Text ●
- Colours ●
- Images ●
- Variables ●

- Input
- Processing
- Output

The world of coding has its own jargon.

Here you will get to know some of the important **coding terms**.



5. Assign the technical terms to the images:

Infinite loop - statement - variable - condition

You clearly describe to the computer what it should do. That is called:



An instruction is executed again and again without stopping.

That is called:



S

An instruction is only carried out when the computer has received a certain input.

That is called:

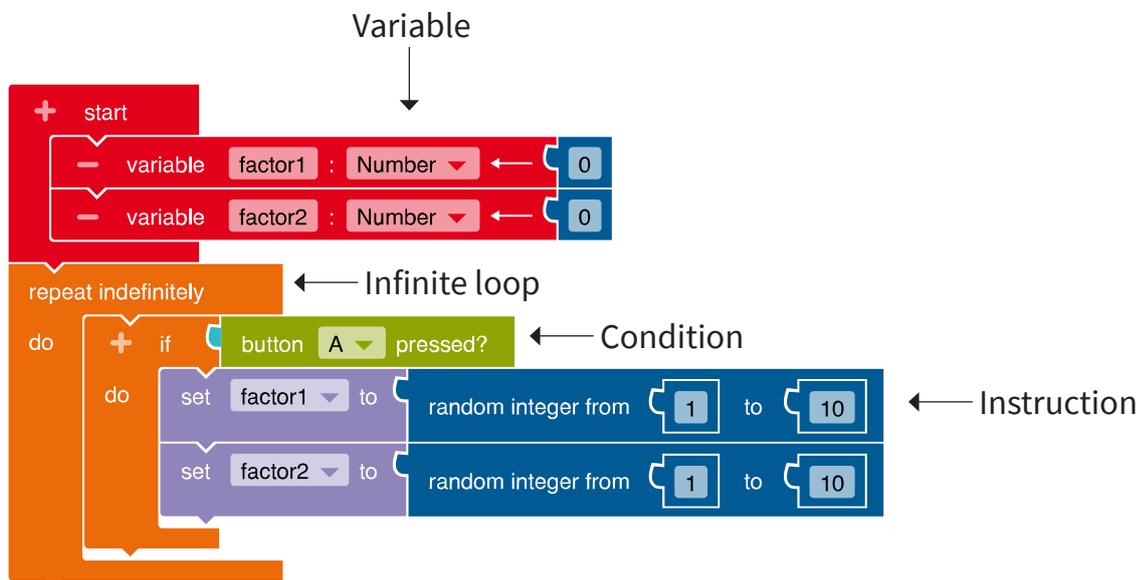


There are containers in which values (numbers, words ...) are stored that the computer will later work with.

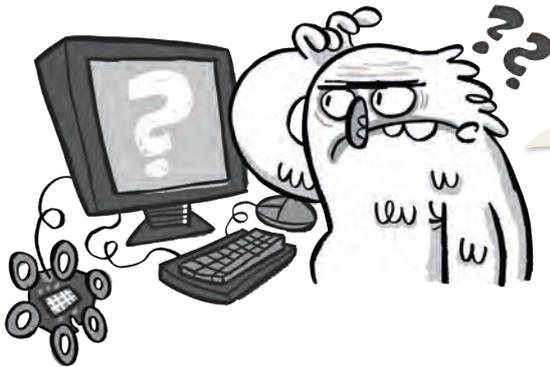
They are called:



In the editor it looks like this, for example:



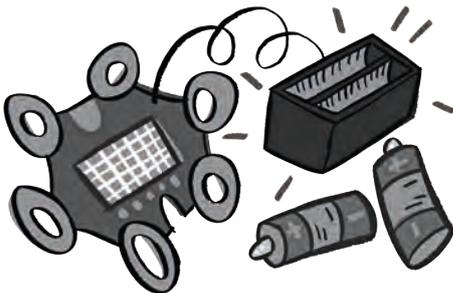
What to do in the event of mishaps and errors?



Have you ever had the situation where nothing seemed to work as it should?

There are mishaps and errors in coding.
Everyone who has already programmed knows that.

Sometimes the problem isn't your programme,
rather:



or



or...

My tip:
Be patient and try trouble-
shooting again and again.

In this booklet you will find five exciting
coding ideas.
You can reprogramme, change or expand
it.

Don't be afraid to invent your own codes!
You can find some tips on this on page 39.

Good luck!



Determine neighbouring numbers with the Calliope mini

Lio and the race

Lio wants to organise a race. Lio prints T-shirts with start numbers for it.

The number has to be re-entered into the computer for each T-shirt.

That is pretty tedious.

Perhaps Lio can write a programme that outputs neighbouring numbers very quickly.



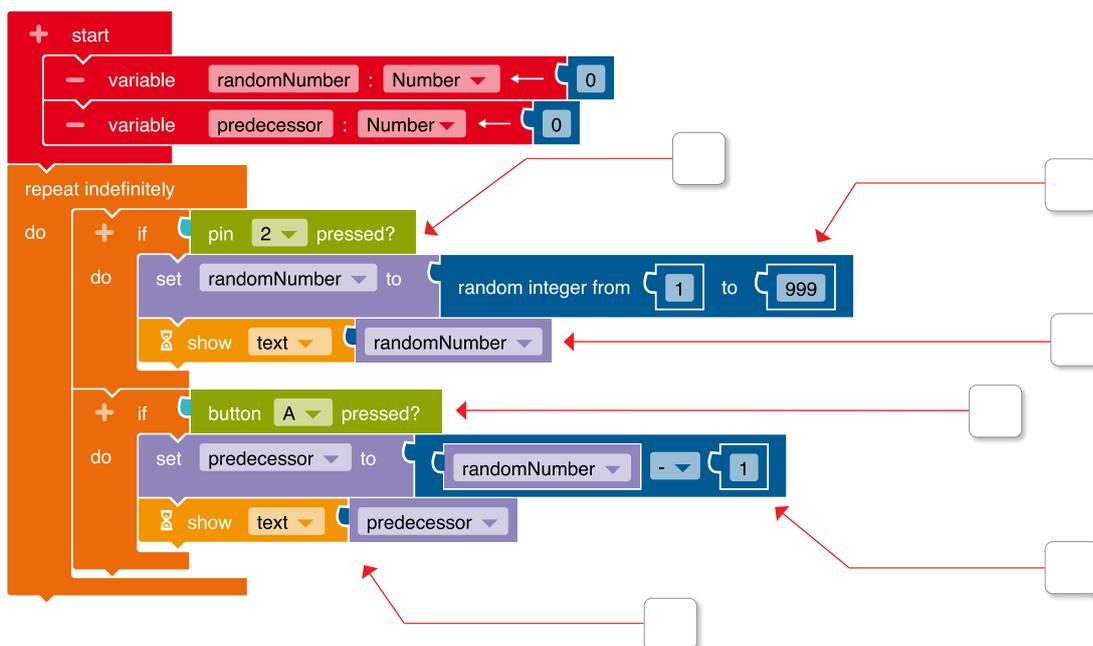
The calculating machine

A computer can solve maths problems at lightning speed. To do so, a person must have programmed in beforehand what it should calculate. To determine neighbouring numbers, the computer should calculate a number +1 or -1.

The code

If pin 2 of the Calliope mini is touched, a random number should be displayed on the LED screen.

Press the A button to output the predecessor.



1. Put the letters I (input), P (processing), and O (output) in the boxes. Each letter occurs twice.



2. a) Programme this code in the editor .

Do it step by step.

- Open "User"
- Click on "Login"
- Enter user name and password
- Open "Edit"
- click on "Save as"
- Assign a name for the programme
- Click on "OK"

A little tip: Keep saving changes !



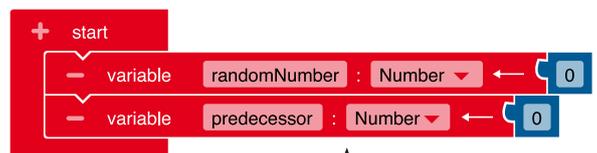
- You need two variables so that new random numbers are always output.

To do this, click on the "+" next to "Start".

Click on the word "Element" and type in the new variable name "randomNumber" with the keyboard.

Repeat the two steps.

Give the second variable the name "predecessor".



↑
variable

In the editor, German letters ä, ö and ü must be written as ae, oe and ue.



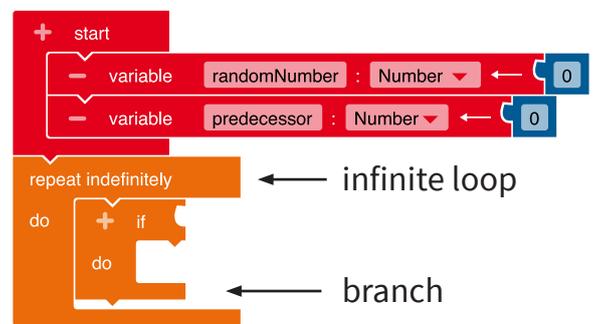
- So that the random numbers can be output infinitely often, you need an infinite loop.

Control → "Repeat /do" infinitely

- When an input is made (if), a random number should be output (do).

To do this, you need a branch.

Control → "if/do"



- If pin 2 is touched, a random number should be output.

Sensors → “Pin 1 pressed”

Add the block to the branch as a condition.

Click with the mouse on the "1".

A drop-down menu opens.

From it, select "2".

Variables → “Write randomNumber”

Insert the block into the branch.

Mathematics → "random integer between 1 and 100"

Add the block to the block "Write randomNumber".

Change the 100 to a 999.



```

+ start
- variable randomNumber : Number ← 0
- variable predecessor : Number ← 0
repeat indefinitely
do
+ if pin 2 pressed?
do
set randomNumber to random integer from 1 to 999
  
```

- The random number should now be displayed on the LED screen.

Action → "Show Text" Add the block to the branch. Remove the block "Hello".

Variables → “randomNumber”

Add the block to the "Show text" block.

```

+ start
- variable randomNumber : Number ← 0
- variable predecessor : Number ← 0
repeat indefinitely
do
+ if pin 2 pressed?
do
set randomNumber to random integer from 1 to 999
show text randomNumber
  
```

- If button A is pressed (if), the predecessor of the random number should be determined (do).

To do this, you need another branch with a condition.

Control → “if/do”

Sensors → “Button A pressed”

Variables → “Write predecessor”

Insert the block into the branch.

Mathematics → „“

Change the arithmetic sign to a subtraction sign by clicking the "+" and selecting the "-" in the drop-down menu.

Put in: **Variables** → “randomNumber”

Mathematics →  Change the number to a 1.

```

+ start
- variable randomNumber : Number ← 0
- variable predecessor : Number ← 0
repeat indefinitely
do
+ if pin 2 pressed?
do
set randomNumber to random integer from 1 to 999
show text randomNumber
+ if button A pressed?
do
set predecessor to randomNumber - 1
show text predecessor
    
```

- The Calliope mini should display the predecessor of the random number.

Action → “Show text”

Remove the block "Hello".

Variables → “Predecessor”

Add the block to the "Show text" block.



b) ▶ Transfer the code to the Calliope mini and run the programme.

Remember that you also have to touch the ⊖-Pin when determining the random number. Only then is the circuit closed.



3. In order to also display the successor of the random number, you need the following blocks.



a) Match the blocks to the correct places in the code.
Enter the numbers in the fields.

1

set successor to randomNumber + 1

2

button B pressed?

3

show text successor

4

variable successor : Number ← 0

+ start

- variable randomNumber : Number ← 0

- variable predecessor : Number ← 0

repeat indefinitely

do

+ if pin 2 pressed?

do

+ set randomNumber to random integer from 1 to 999

+ show text randomNumber

+ if button A pressed?

do

+ set predecessor to randomNumber - 1

+ show text predecessor

+ if

do



b) Programme the missing blocks in the editor.

c) ▶ Transfer the code to the Calliope mini and run the programme.

4. Change your code like this: If the Calliope mini is shaken, a random number in the range 100–500 should be output.

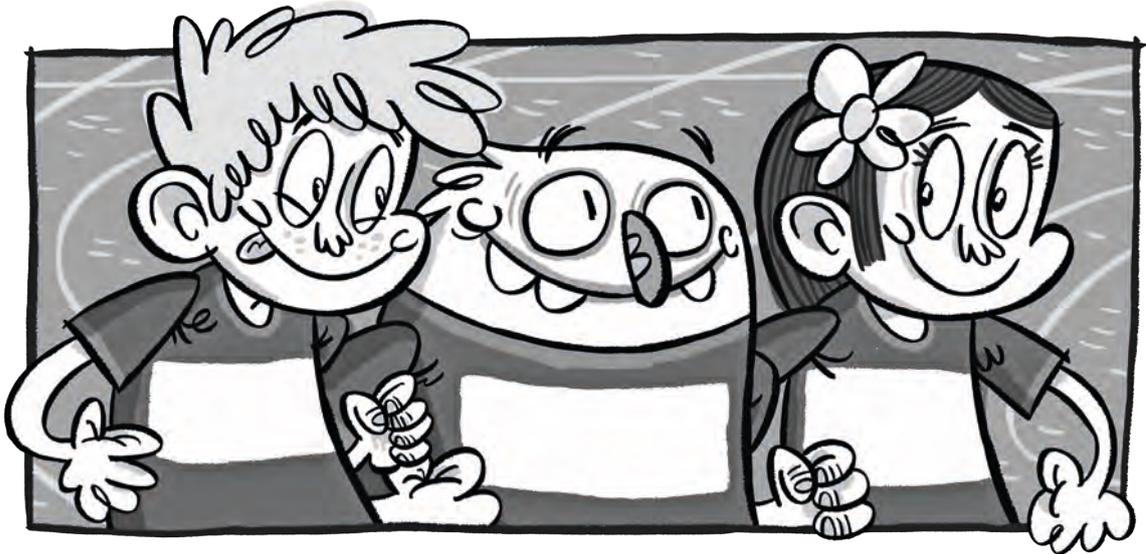
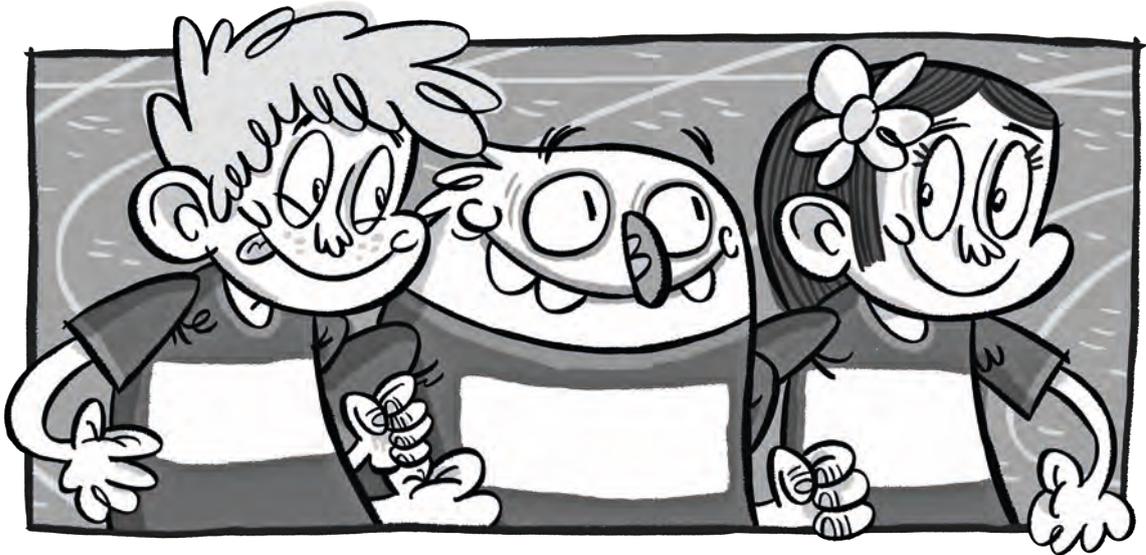


a) Circle the places in the programme where you need to change something.

b) Change your code in the editor.

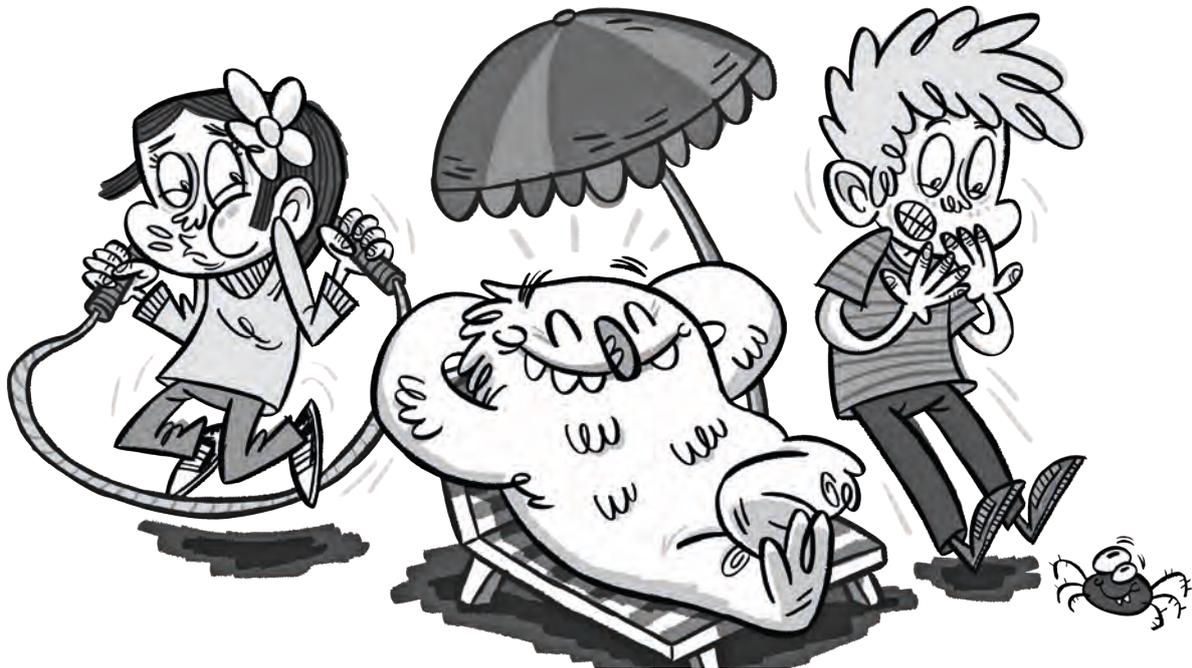
c) ▶ Transfer the code to the Calliope mini and run the programme.

5. a)  Generate a random number with the Calliope mini.
This is Lio's starting number.
- b)  Enter the random number on Lio's t-shirt.
Add the start numbers for Mats and Mia -
the preceding number and the successor number.
- c)  Check your result with the Calliope mini.
- d) Repeat the process.



6. You can also do other calculations, such as doubling a number. Change your programme and try it again.

The Calliope mini as a clock



Lio and the heartbeat

Lio, Mia and Mats observe that their hearts are sometimes fast and sometimes slow. Their heartbeat gets fast during exercise and when they are excited. When they sit quietly, their heart beats slowly.

Lio wants to show how fast a heart beats.

The metronome

The Calliope mini should become a metronome that regularly flashes several times in a minute or makes a tone.

You may know metronomes from music. They help to control the rhythm when making music. With a metronome you can also visualise and compare heartbeats of different people.

The code

The Calliope mini's RGB LED should flash once per second.

The flashes of light are created by switching the RGB LED on and off.

Waiting times must be defined so that the switching on and off of the light is visible.

The first waiting time determines how long the RGB LED lights up.

The second waiting time determines how long the RGB LED is switched off.



1. a) You need the following blocks for your programme.

Write the numbers of the descriptions to the appropriate blocks.

Block	Description
 <input type="checkbox"/>	<div style="border: 1px solid gray; border-radius: 10px; padding: 2px 10px; display: inline-block;">1</div> Programme start
 <input type="checkbox"/>	<div style="border: 1px solid gray; border-radius: 10px; padding: 2px 10px; display: inline-block;">2</div> The light signals should flash infinitely often.
 <input type="checkbox"/>	<div style="border: 1px solid gray; border-radius: 10px; padding: 2px 10px; display: inline-block;">3</div> The RGB LED is switched on.
 <input type="checkbox"/>	<div style="border: 1px solid gray; border-radius: 10px; padding: 2px 10px; display: inline-block;">4</div> First waiting time: 100 milliseconds (ms) (This is how long the RGB LED lights up.)
 <input type="checkbox"/>	<div style="border: 1px solid gray; border-radius: 10px; padding: 2px 10px; display: inline-block;">5</div> The RGB LED is switched off again.
 <input type="checkbox"/>	<div style="border: 1px solid gray; border-radius: 10px; padding: 2px 10px; display: inline-block;">6</div> Second waiting time: 900 milliseconds (ms) (pause between the light flashes)



b) Put the blocks together in the editor 1 in the correct order.

c)  Try out the code in the simulation.

Remember to always start the simulation with .

Is the RGB LED flashing regularly now?



2. What do you have to change in the code so that the RGB LED flashes more frequently in one minute? Put a cross.

I have to extend the second waiting period.

I have to shorten the second waiting period.

3. An elephant's heart beats slower than a human's heart.

A rabbit's heart, on the other hand, beats a lot more times in a minute than a human's.

Creature	Pulse values (heartbeats per minute)
Adult	approx.70
Baby	approx.120
Elephant	approx.24
Rabbit	approx.280



The number of heartbeats in a minute is called the pulse.

The heart of a mouse can beat up to 1000 times per minute. A whale's heart beats only about six times.



a) With the Calliope mini you can make the different pulses visible.

To do this, you have to change the second waiting time. Highlight the place in the code in green.

```

+ start
repeat indefinitely
do
  turn LED on Port internal colour
  wait ms 100
  turn LED off Port internal
  wait ms 900
  
```



b) Change your code so that the pulse of the elephant and the pulse of the rabbit are visible once. Use the values from the table to do so.

Creature	Second waiting time (in milliseconds)
Adult	approx.800 ms
Baby	approx.400 ms
Elephant	approx.2500 ms
Rabbit	approx.100 ms



c)  Try out the code in the simulation.

- Your pulse is not always the same either. If you do sport, it is faster, if you are relaxing it is slower.



- Change the programme so that you can change the beat with the buttons on the Calliope mini.

When button A is pressed, the metronome should flash 50 milliseconds (ms) faster.

When button B is pressed, the metronome should flash 50 ms slower.

This is what the code of the finished programme looks like:

A variable must be created so that the tempo of the metronome can be changed again and again. Call it "waitingTime".

The value of the variable "waitingTime" indicates how long the RGB LED should be switched off. 500 ms corresponds roughly to the pulse value of a child.

When you press button A, the waiting time is reduced by 50 ms. This causes the RGB LED to flash faster.

Each time you press button B, the waiting time is extended by 50 ms. As a result, the RGB LED flashes more slowly.



- Programme this code in the editor .



- Transfer the code to the Calliope mini and run the programme.



- Modify the program so that a heart shows the beat on the LED screen.



6. a) Adjust the Calliope mini so that it shows your heartbeat.
Do it like this:
- Sit down quietly.
 - Feel your pulse and observe the Calliope mini.
 - Adjust the blinking of the Calliope mini to your pulse. To do this, press button A or B.



- b) Find a partner who stops the time (15 seconds).



Count how often your Calliope mini flashes during this time. Enter the value in the table on the right.

- c) Calculate the heartbeat in one minute.

Resting heart rate

after 15 seconds	after one minute

Multiply your reading by four and you will get your heart rate per minute.



- d) Now do 20 squats. Immediately afterwards, feel your pulse again and adjust the Calliope mini to your pulse.
Then repeat the process from exercise 6b) and 6c).



Enter the value in the table on the right.

Pulse after movement

after 15 seconds	after one minute



- e) How has your pulse changed?



The Calliope mini as a stopwatch

Lio and the time

Lio is fascinated by time. Sometimes it goes very quickly and sometimes it seems to stand still, especially when you have to wait. Lio wants to measure the time and know how much time passes before the school bus arrives.



The clock - time display and time measurement

Clocks can tell the time. Lio's watch shows whatever time it is right now (time display).

But there are also clocks that measure the duration from one point in time to the next (time measurement).

Lio would like to build one of those clocks with the Calliope mini.

The code for measuring time

The Calliope mini should show how much time has passed between pressing button A and pressing button B.

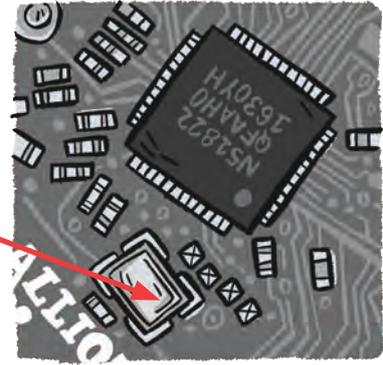


1. a) Clocks have different names. Look at the clocks in the table and write their names in the second column.
- b) Decide whether the clocks show a time or a length of time. Put a cross.

~~Wrist watch~~ - Stopwatch - Sun dial - Hourglass - Pendulum clock

Time	Name of the clock	Time display	Time measurement
	Wrist watch	X	
			
			
			
			

2. A quartz crystal is built into the Calliope mini. It swings like the pendulum of a pendulum clock, only thousands of times faster and is microscopic.



Find the quartz crystal on the Calliope mini.

3. The quartz crystal begins to vibrate when the programme is started on the Calliope mini. This enables the Calliope to measure mini milliseconds (ms).

With the following code you can visualise how much time has passed since the start of the programme.

1000 ms = 1 second



- a) Programme this code in the  editor. Make sure to save your code . If button A is pressed, the time since the programme start should be displayed.

```

+ start
repeat indefinitely
do
+ wait until
get pressed button A = true
show text get value ms timer 1
  
```



- b) ▶ Transfer the code to the Calliope mini and run the programme.

Whenever you press button A, a new time is displayed because the clock in the Calliope mini continues to run invisibly.



c) Press button A. Write down the number (start time).

Count to five in your head.

Now press button A again. Write down this number (end time).

Start time: _____ ms End time: _____ ms



d) With these two numbers you can calculate the length of time between the first and the second press of the button.

Add to this: **End time** – **Start time** = Duration

End time					ms
– Start time:					ms
= Duration:					ms



4. a) Round the duration from task 3d) up or down to the nearest thousand.

Example: 4587 ms → rounded: 5000 ms

ms → rounded:

 ms
 (Duration)



b) Now convert the milliseconds (ms) to seconds (s).

Divide the number by 1000.

Example: 5000 ms): 1000 = 5 s

ms : 1000 = s



Can't the Calliope mini do the calculation for me?



5. The Calliope mini is intended to be a stopwatch that calculates a length of time on its own. For this you need the code from page 20 and the following blocks.

You need variables so that you don't always have to write down the start time and end time individually.

```

+ start
- variable startTime : Number ← 0
- variable endTime : Number ← 0
  
```

The following block does the calculation for you. It rounds the time and converts the milliseconds into seconds.

```

show text round (endTime - startTime) / 1000
  
```

Here's the complete code for the stopwatch.

```

+ start
- variable startTime : Number ← 0
- variable endTime : Number ← 0

repeat indefinitely
do
+ wait until (get pressed button A) = true
set startTime to (get value ms timer 1)
turn LED on (Port internal) colour green
play (quarter) note c'
+ wait until (get pressed button B) = true
set endTime to (get value ms timer 1)
turn LED on (Port internal) colour red
show text round (endTime - startTime) / 1000
turn LED off (Port internal)
  
```

The stopwatch starts and ends with the push of a button. When you press the buttons, the RGB LED lights up and a sound is made.



a) Take a close look at the code and complete the text.

When _____ pressed,
the stopwatch starts.

The RGB LED lights up in colour _____,
and a sound is played.

When _____ pressed, the stopwatch stops.

The RGB LED lights up in colour _____,
and a sound is played. The measured time is then displayed in seconds.



b) Programme this code in the editor ☆2.

Check what you can use from the code from exercise 3a) and what you need to change.

Look for the division sign \div in the mathematics block.



c) ► Transfer the code to the Calliope mini and run the programme.



6. Think about activities that you can do in class. Write them in the table.

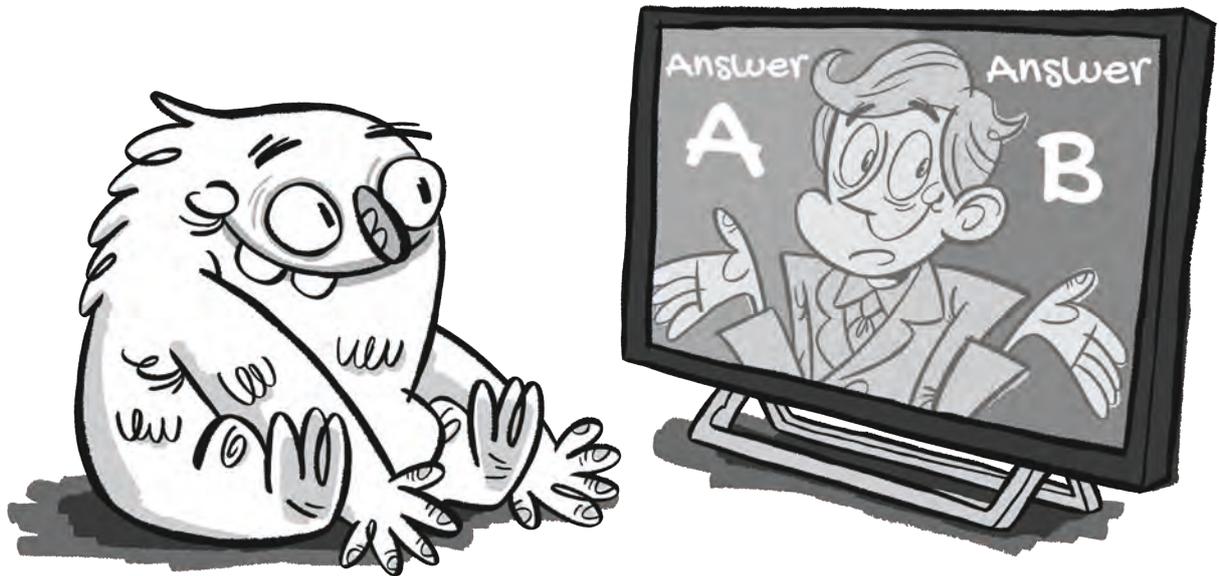
Carry out the activity and have another child stop the time with the Calliope mini.

Record your results in the table.

Activity	Stopped time in seconds
Stand on one leg without falling over.	10

The Calliope mini as a spelling trainer

Lio and the TV quiz



Lio is watching a quiz on TV. The moderator asks questions and gives two different solutions. The participants have to guess which solution is correct. Lio plays along and thinks about it at home.

It's really fun!

Lio has an idea: A quiz programme for the Calliope mini is a great way to practice your spelling.

The spelling quiz

Quizzes can help with learning.

A computer displays words with missing letters and offers two possible spellings to choose from.

The advantage is: The questions can be repeated as often as you like.

The code

The code for the spelling trainer consists of four parts.

The Calliope mini should:

- display a quiz question and offer two ways of spelling,
- check the player's answer,
- show the result (right or wrong),
- repeat the first three steps with further questions.



1. Think of three short words that are difficult to write. Check their correct spelling. Then formulate quiz questions about the words and write them down in the table. Proceed as in the example.

Make sure that sometimes A and sometimes B is the correct answer.

No.	Question	Solution A	Solution B	correct answer
0	PIN_ K or C?	K	C	A
1				
2				
3				



2. Programme the code for the spelling trainer in the ☆2 editor. Do it step by step.

- First you have to enter your questions and solutions and save them in the programme. To do this, create two variables and call them “questions” and “solutions”.
Select "String list" in the drop-down menu.



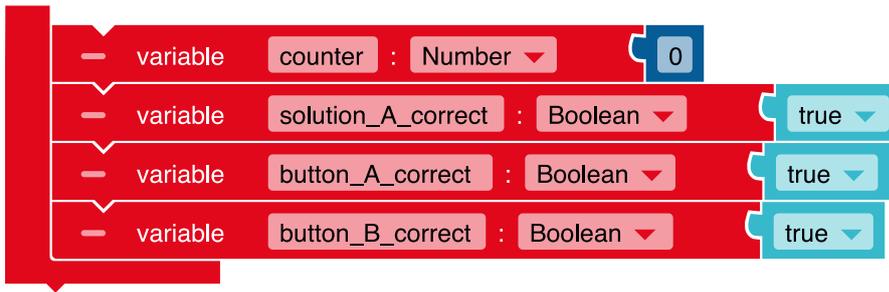
If you click on "+", a new line will appear.

```

+ start
- variable questions : List String
+ list : String
  " PIN_ K OR C? "
  " "
  " "
  " "
- variable solutions : List String
+ list : String
  " A "
  " "
  " "
  " "
    
```

- Complete the first list with your questions from exercise 1. In the second list, write which solution is the right one (A or B). Proceed as in the example (question: PIN_ K OR C,Solution: A).

- Add the following four variables.



View the first quiz question

- When the Calliope mini is turned upside down and straightened up again, the quiz should begin.



- a) Which blocks have to be added to the "Wait until" blocks?
Look in the category ☆2 under the **Sensors** category.



Add the "Wait until" blocks under the variables to the start block.

- The first question from the "questions" list on page 25 should be displayed. It is on list position 0.
To display question 0, you need the following blocks from the categories **Action**, **Lists** and **Variables**.

	list positions
“ PIN_K OR C? ”	0
“ ”	1
“ ”	2
“ ”	3



In the programming language, the number of a list position is also called an index.

Counter is = 0 at the beginning of the programme.

Check the answer

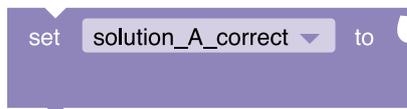
- The Calliope mini is supposed to wait for the player's answer. Button A or button B should be used to enter the answer. To do this, add the following blocks.



- Now it has to be determined which answer is correct. The Calliope mini should check what is in the "Solutions" list.

To do this, you need the following blocks from the categories:

Variables, Logic, Lists, and Text



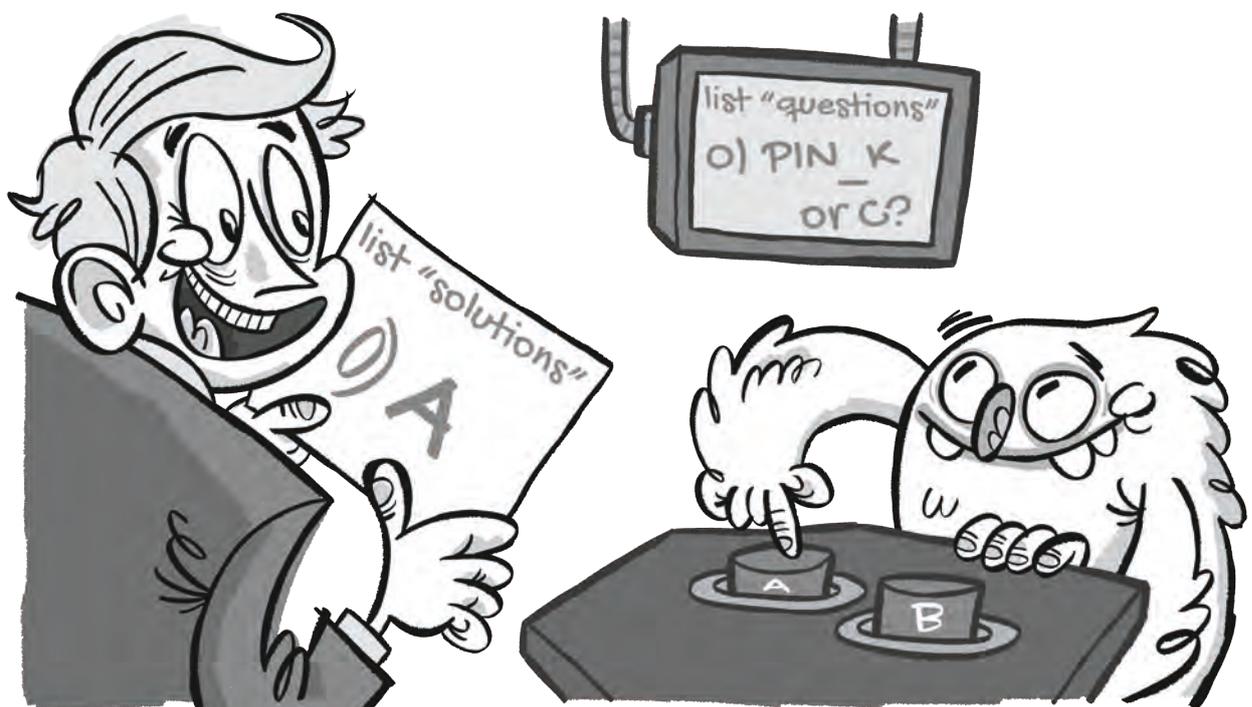
Check whether answer A is correct.

A comparison should be made ...

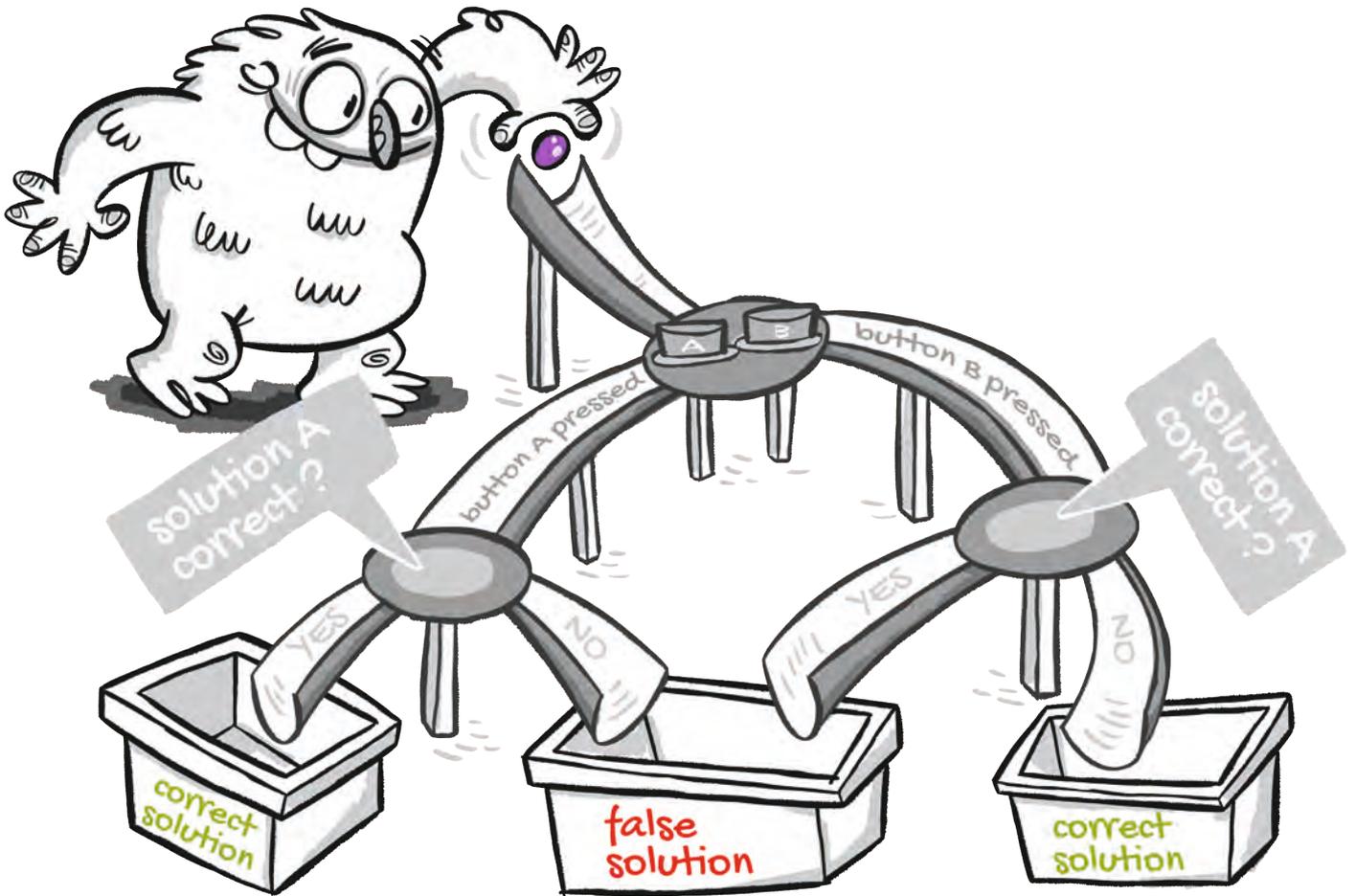
...whether there is an A in the correct list position (e.g. position 0)

in the "solutions" list.

Put the blocks together and add them to your code.



- Next, the Calliope mini has to evaluate the pressed button as a correct or incorrect answer. There are different options:



That is enough to tie you in knots!



b) Think about which letters are missing from the empty spaces in the following block. Take a close look at the marble run. Fill in:

```

set button  correct to  button A pressed? and  solution_A_correct
set button  correct to  button B pressed? and  not  solution_A_correct
  
```



c) Programme the blocks in the editor and save your code .

Show result

- When the correct button has been pressed (if) a check mark should appear on the LED screen (do). If the wrong key has been pressed, a cross should be displayed (else).

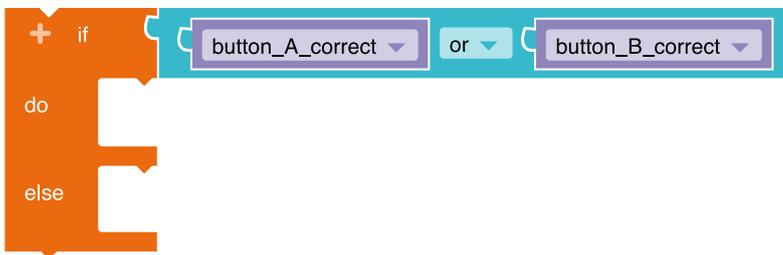


To do this, you need a branch.

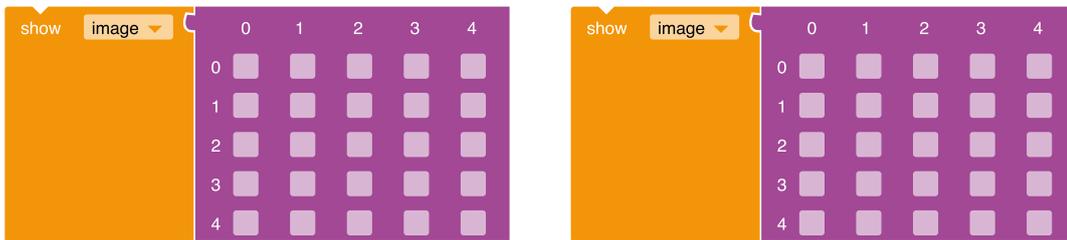
Control decision → "if/do/else"

- The correct button can be either button A or button B. From the **Logic** category, add the logical comparison with "and" and change it to "or".

Complete the variables.



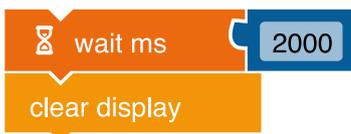
- You need the following blocks to display the tick or the cross.



d) Colour the boxes so that there are a tick and a cross.

Insert the blocks into the loop.

- The respective image should be visible for 2000 milliseconds (ms). Add the following blocks under the "if/do/else" block.



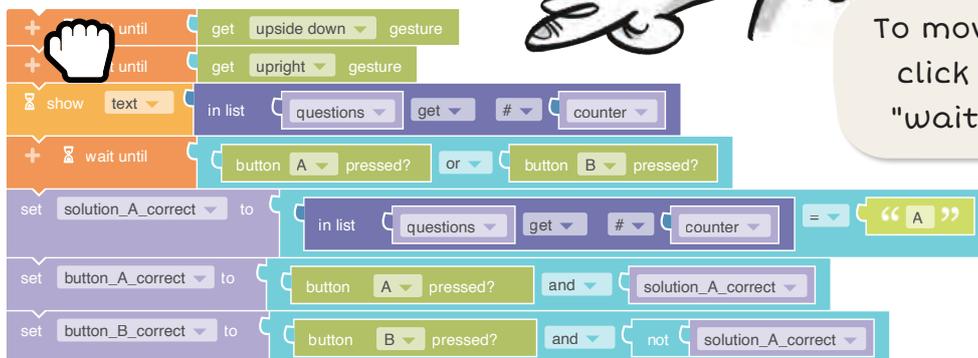
Show next question

- So far the Calliope mini can only display the first question. This is because the "counter" = 0. If the next question is to be displayed, the "counter" must be changed. So add the following blocks from the categories **Mathematics** and **Variables**.



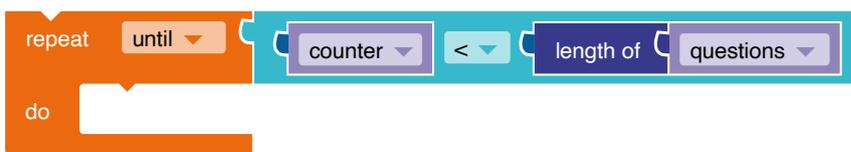
Now the counter = 1.

- To make sure that the programme does not end at this point, you need a loop. Add the loop "repeat to/do" at the top of the programme directly after the variables. Change the "until" to "so long as".



To move the blocks, click on the first "wait until" block.

- Now paste your remaining code into the loop.
- The loop should be repeated as long as the "counter" is less than the number of questions. For this you need the following blocks from the categories **Logic**, **Variables** and **Listen**.



e) When does the loop end for a quiz programme with four questions?
"Counter" = _____



3. ▶ Transfer the code to the Calliope mini and run the programme.

This is what the full code of the spelling trainer looks like. You have to fill the lists with your own questions and solutions.

```

+ start
- variable questions : List String
  + - list : String
    " PIN_K OR C? "
    " "
    " "
    " "
- variable solutions : List String
  + - list : String
    " A "
    " "
    " "
    " "
- variable counter : Number
  0
- variable solution_A_correct : Boolean
  true
- variable button_A_correct : Boolean
  true
- variable button_B_correct : Boolean
  true

repeat until counter < length of questions
do
+ wait until get upside down gesture
+ wait until get upright gesture
show text in list questions get # counter
+ wait until button A pressed? or button B pressed?
set solution_A_correct to in list questions get # counter = " A "
set button_A_correct to button A pressed? and solution_A_correct
set button_B_correct to button B pressed? and not solution_A_correct
+ if button_A_correct or button_B_correct
do
show image


|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| 0 |   |   |   |   |
| 1 |   |   |   | # |
| 2 |   |   |   |   |
| 3 | # |   | # |   |
| 4 |   | # |   |   |


else
show image


|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| 0 | # |   |   | # |
| 1 |   | # |   | # |
| 2 |   |   | # |   |
| 3 |   | # |   | # |
| 4 | # |   |   | # |


wait ms 2000
clear display
change counter by 1
  
```



4. Add other actions, such as a coloured status light or sounds.
5. Rebuild the code to create an English quiz or a science class quiz. To do this, change the lists of questions and solutions.

The Calliope mini and the Nim game

Lio is looking for a second player

Lio found a box with 12 small sticks with NIM on it. What does that mean?

Then it occurs to Lio: This is a nim game!

The rules are simple:

- There are two players and 12 sticks.
- The players can take turns taking 1, 2 or 3 sticks.
- Whoever takes the last stick wins the game.

Lio is still missing an opponent.

Maybe Lio can play against the Calliope mini?



A minicomputer as an opponent

The Calliope mini has neither eyes, arms nor a mouth.

So it can neither see nor take sticks, nor can it say how many sticks it wants to take.

But it can be programmed in such a way that it becomes an opponent anyway.

The code

The code needs to solve the following problems:

- A) How does the Calliope mini know how many sticks are involved?
- B) How does the Calliope mini know how many sticks Lio wants to take away?
- C) How does the Calliope mini perform its move?
- D) How does the Calliope mini check who has won?



1. Think in which category you will find blocks for solving questions A) and B). Put a cross.

Sensors

Variables

Action

Colours

The code for the Nim game consists of 4 parts:

1 Preparing for the game

2 It's Lio's turn

If an entry is made, 1, 2 or 3 sticks should be deducted from the starting value 12.

3 It is the Calliope mini's turn

If there are still sticks in play, a random number between 1 and 3 should be subtracted from the sticks.

4 The decision

If there are 1, 2 or 3 sticks left after the Calliope mini's last turn, Lio wins. A 😊 is displayed.

The image shows a Scratch script for a Nim game, organized into four distinct sections indicated by brackets on the right:

- Preparing for the game:** A 'start' block followed by four 'variable' blocks: 'Number_sticks' (Number, 12), 'Lio_takes' (Number, 0), 'Lios_turn' (Boolean, true), and 'Calliope_takes' (Number, 0).
- It's Lio's turn:** A 'repeat until' loop where 'Number_sticks' is less than or equal to 0. Inside, it shows 'Number_sticks', sets 'Lios_turn' to true and 'Lio_takes' to 0. A second 'repeat until' loop where 'Lio_takes' is not equal to 0 contains three 'if' blocks for 'pin 1', 'pin 2', and 'pin 3' pressed, each setting 'Lio_takes' to 1, 2, or 3 respectively.
- It is the Calliope mini's turn:** A 'set' block for 'Number_sticks' to 'Number_sticks - Lio_takes', followed by a 'show text' block for 'Number_sticks'. An 'if' block where 'Number_sticks' is greater than 0 contains a 'do' block that sets 'Lios_turn' to false, 'Calliope_takes' to a 'random integer from 1 to 3', and 'Number_sticks' to 'Number_sticks - Calliope_takes'.
- The decision:** An 'if' block where 'Lios_turn' is true, showing a 😊 image; otherwise, it shows a 😞 image.



2. a) Programme the code for the Nim game in the editor ☆2 . Do it step by step.

Preparation

- First create four variables.
Name the first variable "Number_sticks".
It saves the number of sticks present. Change the starting value to 12.

The variable "Lio_takes" stores how many sticks Lio takes.

The start value is 0.

The variable "Lios_turn" stores whether it is Lios turn ("true") or not ("false").

The "Calliope_takes" variable stores how many sticks the Calliope mini takes.



```

+ start
- variable Number_sticks : Number ← 12
- variable Lio_takes : Number ← 0
- variable Lios_turn : Boolean ← true
- variable Calliope_takes : Number ← 0
  
```

- The game should run until the number of sticks is less than or equal to 0 (≤ 0), i.e. no more sticks are in play.

Use the correct blocks from the categories **Control**, **Logic**, **Variables** and **Mathematics**.

```

repeat until
do
  
```

Logic block: Number_sticks ≤ 0

- During the game the number of sticks left should always be visible. Add the blocks "Show Text" and "Number_sticks" from the categories **Action** and **Variables**.

```

repeat until
do
  show text Number_sticks
  
```

It's Lio's turn

- First, it must be determined that it is Lio's turn. The value of "Lios_turn" becomes "true".

A Scratch 'set' block with 'Lios_turn' in the variable dropdown and 'true' in the value field.

- Lio hasn't removed any sticks yet. The variable "Lio_takes" is = 0. Add the blocks from the categories **Variables** and **Mathematics**.

A Scratch 'set' block with 'Lio_takes' in the variable dropdown and '0' in the value field.

- The following process should be repeated until Lio has selected a number of sticks. Use the correct blocks from the categories **Control**, **Logic**, **Variables** and **Mathematics**.

A Scratch 'repeat until' block with 'Lio_takes' in the variable dropdown, '≠' in the operator dropdown, and '0' in the value field.

- When pin 1, 2 or 3 is pressed, 1, 2 or 3 sticks should be removed from the existing sticks. Choose the right blocks from the categories **Control**, **Sensors**, **Variables** and **Mathematics**.

A complex Scratch code block starting with a 'repeat until' loop. The condition is 'Number_sticks ≤ 0'. Inside the 'do' loop, there is a 'show text' block for 'Number_sticks', followed by 'set Lios_turn to true' and 'set Lio_takes to 0'. This is followed by another 'repeat until' loop with condition 'Lio_takes ≠ 0'. Inside this loop, there are three 'if' blocks: 'if pin 1 pressed?' with 'do' block 'set Lio_takes to 1'; 'if pin 2 pressed?' with 'do' block 'set Lio_takes to 2'; and 'if pin 3 pressed?' with 'do' block 'set Lio_takes to 3'.

- The number of sticks that are removed must be subtracted from the number of sticks present. To do this, add the correct blocks from the categories **Variables** and **Mathematics** under the loop. Choose the correct mathematical symbol.

```

set Number_sticks to [Number_sticks] - [Lio_takes]
  
```

- After a pin has been pressed, the number of remaining sticks should be displayed. Add a "Show Text" block.

```

show text [Number_sticks]
  
```



b) Which numbers are displayed when Lio presses pin 2, then pin 3, then pin 1 and then pin 3 in sequence?

Add to this:

existing sticks	-	pressed Pin	=	displayed Number
12	-	2	=	10
10	-	3	=	<input type="text"/>
<input type="text"/>	-	1	=	<input type="text"/>
<input type="text"/>	-	3	=	<input type="text"/>



c) ▶ Transfer the code to the Calliope mini and check your calculation. Make sure to touch the minus pin too.

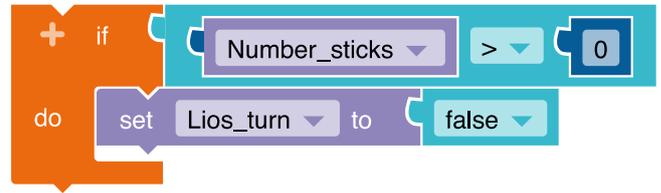
It is the Calliope mini's turn

- If there are still sticks left after Lio's move ($\text{Number_sticks} > 0$), it is the Calliope mini's turn.

The value of the "Lios_turn" tag becomes "false".

Add the correct blocks from the categories **Logic**, **Variables** and **Mathematics**.

Look for the $>$ (greater than) sign.



- A computer cannot decide how many sticks to use. But it can generate random numbers.

The Calliope mini should take a random number of sticks. Assign a random number to the variable "calliope_takes".

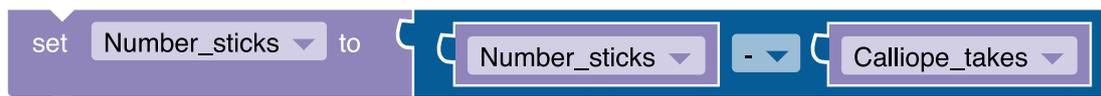


d) From which number range must the random number be created?

Write the numbers in the empty fields and then fill in the blocks in your code.



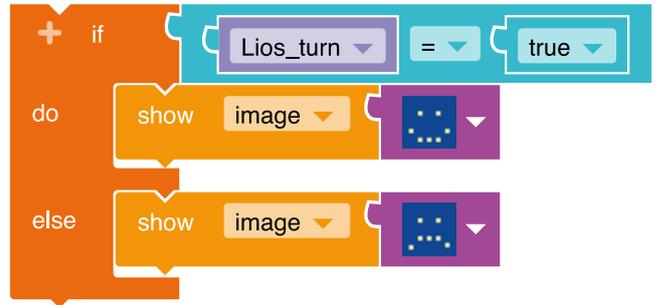
- The number of sticks that the Calliope mini takes must be subtracted from the number of sticks available. Add the correct blocks from the categories **Variables** and **Mathematics**.



The decision

When there are no more sticks, the loop ends and it's game over. Did Lio win?
To find out, it must be checked which player was last in line.

- After the end of the big loop, add a branch “if/do/else”.
If the last stick was taken by Lio (Lios_turn = true), Lio has won.
A 😊 is displayed.
Otherwise the Calliope mini won.
Then ☹ is displayed.



e) ▶ Transfer the code to the Calliope mini and play the Nim game.



3. Who do you think has the better chance of winning the Nim game: Lio or the Calliope mini? Give reasons for your answer.



4. Lio has a hard time remembering whose turn it is.
So expand your code.
When it is Lio's turn, the RGB LED should light up green.
When it is Calliope mini's turn, the RGB LED should light up red.



5. Lio's friend Mats wants to play the Nim game against Lio.
Change the programme so that two people can play against each other.

Solution: see page 42



My tip: You no longer need random numbers for this.

Tips for your own program ideas

1st Develop your own idea.

- What should the Calliope mini do?
- Check whether the Calliope mini has all the necessary functions for this.

2. Select the blocks for your programme.

- Determine which input options should be used (pins, buttons, position sensors, microphone).
- Determine what should be used for the output (screen, RGB-LED, speakers).
- If you want to use values several times in your programme, create a variable for each value.

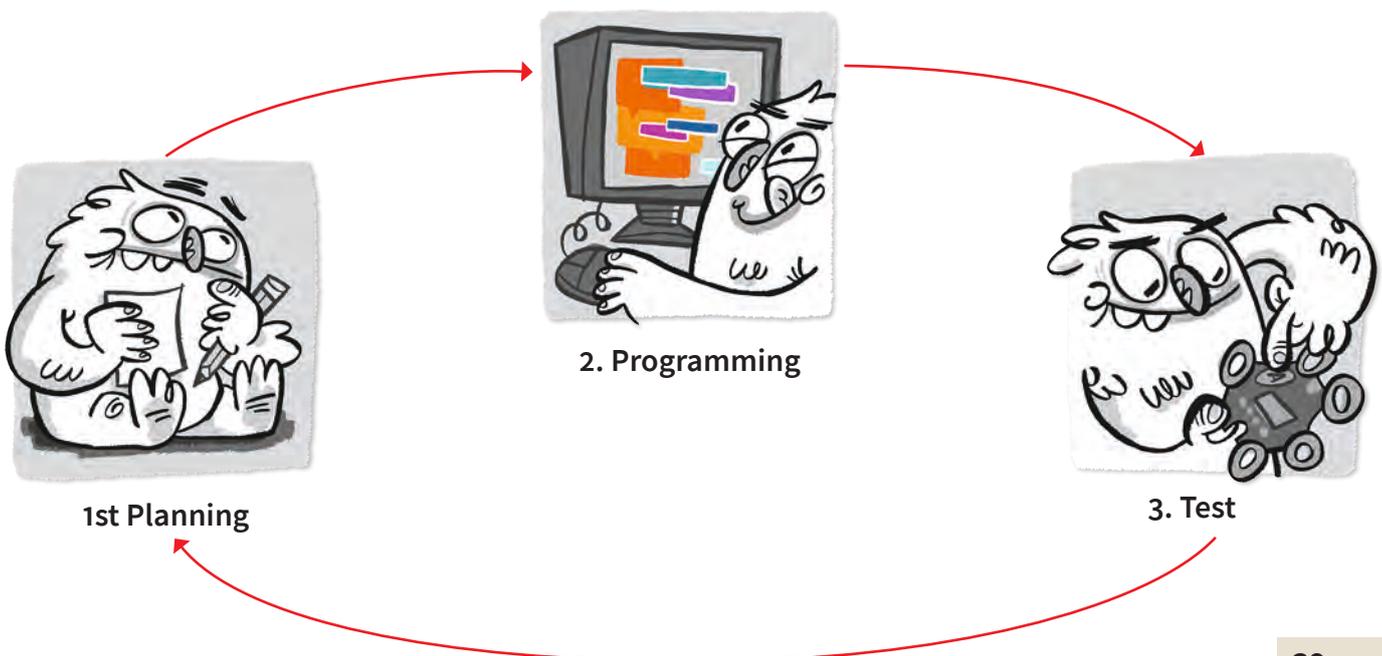
3. Programme in small steps.

- Start with a short programme.
- Expand the programme step by step.

4. Try out your programme again and again, even if it is not yet "finished".

- Use the simulation and the Calliope mini.
Tip: At the beginning, use sensors that are also used in the simulation (e.g. button or pin).
- If you find any mistakes, correct your programme.
Only change a little at a time.
After the change, try your programme again in the simulation.

5. Repeat the steps until the programme works properly.



Programme examples

Here you will find small programmes that you can incorporate into your own ideas.



1st Make outputs visible

If the Calliope mini is to display several different outputs one after the other, you have to use waiting blocks.

For example like this:

The 1 and the 2 appear respectively cannot be distinguished . That is enough to read.

That is too fast: The 1 and the 2 on the screen for half a second

2. Change the colour of the RGB LED with the push of a button

When you press the A button and release it again, the RGB LED should change colour. For this you need waiting blocks.

When exactly does the RGB LED change its colour? Make a cross.

When you press key A.

When you release button A.



3. Make a picture visible for a long or short time

When you press the A key, the “Smiley” image should appear on the screen. As soon as you release button A, the picture should disappear. What does your code have to look like for this to happen? Circle the correct answer.



a)

```

+ start
repeat indefinitely
do
+ if button A pressed?
do
show image [Smiley]
    
```

b)

```

+ start
repeat indefinitely
do
+ if button A pressed?
do
show image [Smiley]
else
clear display
    
```

4. Determine a fixed sequence of entries

Compare the following two programmes. Circle the differences.



a)

```

+ start
repeat indefinitely
do
+ wait until button A pressed?
show image [Smiley]
+ wait until button B pressed?
show image [Smiley]
+ wait until pin 0 pressed?
show image [Smiley]
    
```

b)

```

+ start
repeat indefinitely
do
+ if button A pressed?
do
show image [Smiley]
+ if button B pressed?
do
show image [Smiley]
+ if pin 0 pressed?
do
show image [Smiley]
    
```

One programme expects entries in a certain order: first button A, then button B and then pin 0. Only then do the images appear.

Which programme is meant? It is programme _____.

The other programme can react to the inputs in any order. It doesn't matter which button is pressed first.

Which programme is meant? It is programme _____.

The Nim game for two players

```
+ start
- variable Number_sticks : Number ← 12
- variable Player1_takes : Number ← 0
- variable Player1_turn : Boolean ← true
- variable Player2_takes : Number ← 0

repeat until Number_sticks ≤ 0
do
  show text Number_sticks
  set Player1_turn to true
  set Player1_takes to 0
  repeat until Player1_takes ≠ 0
  do
    + if pin 1 pressed?
    do set Player1_takes to 1
    + if pin 2 pressed?
    do set Player1_takes to 2
    + if pin 3 pressed?
    do set Player1_takes to 3
  set Number_sticks to Number_sticks - Player1_takes
  show text Number_sticks
  + if Number_sticks > 0
  do
    set Player1_turn to false
    set Player2_takes to 0
    repeat until Player2_takes ≠ 0
    do
      + if pin 1 pressed?
      do set Player2_takes to 1
      + if pin 2 pressed?
      do set Player2_takes to 2
      + if pin 3 pressed?
      do set Player2_takes to 3
    set Number_sticks to Number_sticks - Player2_takes
  + if Player1_turn = true
  do show text "Player1 wins"
  else show text "Player2 wins"
```

Tip: You can also enter your own name and the name of your fellow player at the bottom of the code under "Show text".

The little coding dictionary

- Instruction** (= command) When you receive an instruction, you can do it, for example:
 "Hang the wet socks to dry on the clothes horse."
 It is the same with a computer. It carries out instructions that clearly describe what it should do. A code / programme is composed of instructions.
- Loop with a condition** A loop allows a sequence of statements to be executed over and over again.
 For example: "Hang up socks as long as there is laundry in the basket."
 The **loop** is: "Hang (repeat) as long as ..."
 The **condition** of the loop is: "Is there still laundry in the basket?"
 The answer is: "Yes!" **two instructions** are executed one after the other in the loop:
 1. Take a wet piece of laundry
 2nd Hang the laundry on the drying rack
 If the answer is "No!", the programme continues after the loop: "Bring the basket into the bathroom."
- Infinite loop** An endless loop **has no condition** and is run through until the Calliope mini is switched off.
- Variable** A variable is a container for a certain value (number, word, picture or something else) that is specified at the beginning of the programme.
- Branch with a condition** Every branch in a programme needs a condition.
 The condition decides with which instruction the programme is continued.
 There are two ways to do this, for example:
 Condition: "Is the laundry on the drying rack still wet?"
 Branch
 If yes, then: "Wait an hour."
 If no, then: "Take the washing off."
- List** A list is a collection of similar values (numbers, words, or pictures). Each value has a fixed position in the list.
 This position is also called the index. The first value in the list gets the index '0', the second the index '1' and so on.
 Via the block "take #it from the list" you can access individual values in the list.

The most important functions of the Calliope mini

By pressing the **buttons A** and **B** buttons, you make entries so that the Calliope mini executes the programmed commands.

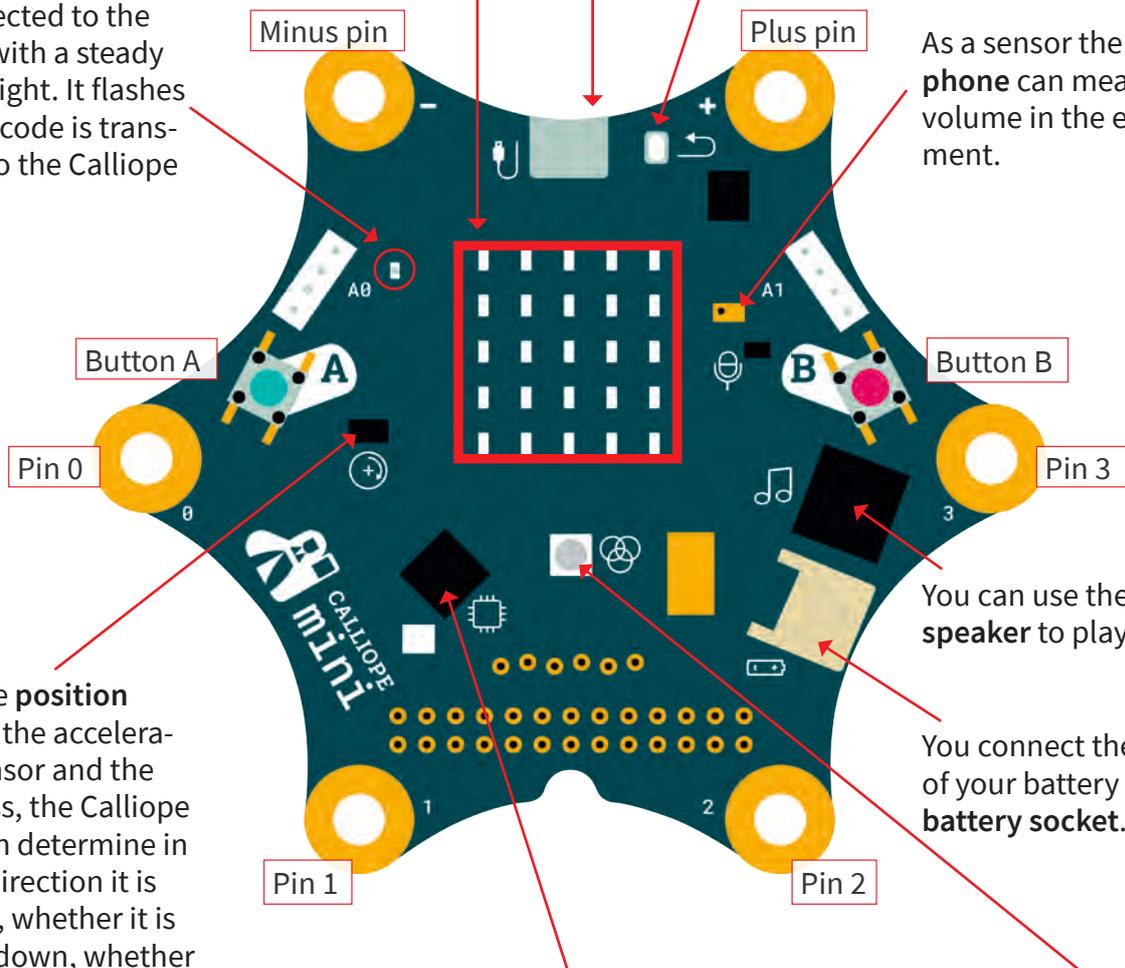
You plug a micro-USB cable into the **USB-port** to connect the Calliope mini to a computer.

The **LED-screen** is a 5 x 5 grid with red light emitting diodes.

With the **Reset-** button you restart the program on the Calliope mini.

The **status light** shows when the Calliope mini is connected to the power with a steady yellow light. It flashes when a code is transferred to the Calliope mini.

As a sensor the **microphone** can measure the volume in the environment.



With the **position sensor**, the acceleration sensor and the compass, the Calliope mini can determine in which direction it is moving, whether it is upside down, whether it is lying down, whether it is upright, in which direction it is turned or whether it is not moving.

The **processor** is the heart of the Calliope mini! The processor combines all functions and processes all information.

The **RGB-LED** can flash and shine brightly in different colours.

You can use the **loudspeaker** to play sounds.

You connect the cable of your battery to the **battery socket**.

By touching **pins 0, 1, 2, or 3** you can also make inputs so that the Calliope mini executes commands. You have to touch the minus pin (-) with the other hand at the same time.



CALLIOPE

Calliope mini is a product of Calliope gGmbH

The *Calliope mini* Microcontroller provides school children with a playful introduction and access to the digital world. Because it is only if we have digital knowledge that we can all actively participate in society and help shape it. For this purpose, experts from the IT and education sectors work together in an interdisciplinary manner as part of the Calliope team.

Find more information about the initiative at calliope.cc



The Open Roberta Lab is a freely available cloud-based programming platform on which children, youths and adults can programme microcontrollers and robots even without any prior knowledge. Schoolchildren intuitively bring the Calliope mini to life with the graphic programming language NEPO® and “drag and drop”. Open Roberta® is a technological open source development of the “Roberta® – Learning with Robots” initiative of Fraunhofer IAIS, which has been promoting digital education in Germany since 2002. The development of Open Roberta® has been supported by Google.org since 2014. Roberta®, Open Roberta® and NEPO® are registered trademarks of the Fraunhofer-Gesellschaft für angewandte Forschung e. V.

Click here for the Open Roberta Lab: lab.open-roberta.org

Terms and Conditions

This document is under the following Creative Commons license: <https://creativecommons.org/licenses/by-sa/4.0/deed.de> - You are allowed to reproduce, distribute and make publicly available the work or the content as well as modifications and adaptations of the work as long as you state the name of the author / rights holder in the manner specified by him and only pass on the newly created works or content under the use of license terms that are identical, comparable or compatible with those of this license agreement.

By using this document you accept the terms of use.

Terms of use

This document is published under following Creative Commons-License: <https://creativecommons.org/licenses/by-sa/4.0/deed.de> – You may copy, distribute and transmit, adapt or exhibit the work or its contents in public and alter, transform, or change this work as long as you attribute the work in the manner specified by the author or licensor. New resulting works or contents must be distributed pursuant to this license or an identical or comparable license. By using this particular document, you accept the above-stated conditions of use.



Jonathas Mello CC-BY 3.0 Unported

Coding with the Calliope mini



Using the workbook Coding with the Calliope mini – Programming in elementary school, you will get to know the subjects of science, mathematics and English from a different angle in five new programming examples:

- Programme a stopwatch
- Play the Nim game with the Calliope mini
- Create your own spelling quiz



You will deepen your coding knowledge, get to know new programmes, and receive tips for your own ideas.

So get started – good luck with your programming!



CALLIOPE.CC



CALLIOPE