



CodingPark



Teacher Manual

Table of Contents

- About1
- General Pedagogy1
- Context of Use2
 - The Lessons2
 - Types of Levels2
 - Pedagogical Guides3
- Setting Up a Workshop4
 - Procedure for the Teacher4
 - Step 1. Selecting Lessons4
 - Step 2. Student Identification5
 - Step 3. Workshop Launch6
 - Procedure for the Student7
 - Simplified Onboarding for Students9
- Automatic Assessment10
- Starting the Video Conference13
 - Prerequisites13
 - Joining a Session14
- Contact Us16
- Annex17
 - Quick Guide17
 - Pedagogical Guides18

About

Coding Park is a playful platform that helps children understand computational thinking through video games. Designed for both students and teachers, the platform enables students to gradually acquire the necessary skills within the game. It offers teachers ready-to-use interactive resources, as well as a workshop monitoring tool that allows real-time assessment of students' progress.

Pedagogy

In the world of Coding Park, Golden Quest is a treasure hunt journey set in a world of robots and pirates. Children create algorithms to solve logical problems with the goal of guiding Cody, the little pirate robot, to the hidden treasure in each exercise.

Orienting oneself and moving using a grid (coordinate system); making movements in space and coding them so that another student can reproduce them; creating representations of a restricted space and using them to communicate positions; programming the movements of a robot or a character on a screen.



Context of use

The platform is suitable for use in recurring workshops in the classroom and/or at home for self-study.

- During workshops and sessions conducted in schools, daycare centers, and other educational institutions, teachers have a tool that allows them to provide educational content and a means to measure and track student progress.
- In self-study mode, the platform offers contextual assistance within the game environment, with content that adapts based on the student's progress, as well as a comprehensive lexicon of actions that can be performed.

The lessons

The Golden Quest learning journey is a treasure hunt game that consists of 16 lessons or chapters. These lessons cover the fundamentals of programming through the adventures of Cody and his crew. Students can choose between Blockly, Pseudocode, or Python to complete the quests.



The types of levels

The types of levels in the course are as follows:

- **Green** levels: These are regular levels where students need to code Cody's actions.
- **Blue** levels: In these are inverted levels, the code is already provided to the student, and they need to draw the minimal path.
- **Red** levels: These are time-constrained levels where students must find a solution within a time limit.



Pedagogical sheets

To assist you in creating and implementing your workshops, we have prepared a pedagogical sheet for each lesson. On these sheets, you will find prerequisites, objectives, and the plan for each lesson.

The lesson plan will provide you with tips and solutions to support your students and to overcome each level. Additionally, on these same sheets, we also compile the most common errors we have observed during our workshops.

The complete list of pedagogical sheets is provided in the appendix.



Getting Started – Part 1

Objectives

- ✓ Get hands on the code editor and learn how to use the debugger
- ✓ Understand the syntactical rules of the pseudo language
- ✓ Understand how to move Cody on the grid

Prerequisites

- ✓ At least one computer for every two students and a good Internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must have joined the session with their session ID

Lesson Plan

1. Click Play: Click on Play and describe what Cody does
2. Golden Tooth: Write down() at line 7
3. The Jumpy Frog: Write jump(3) at line 10
4. The Bushes: Introduce the notion of "bug" and fix bug at line 8
5. The Skeleton: 2 bugs at line 7 and 9, use the appropriate parameters
6. The Snake: Briefly introduce the notion of loop, replace 3 by 8
7. The Dreadful Five: Add right() statement at line 5
8. The Teleporters: Fix the bug at line 4, replace 6 with 5
9. The Teleporter Direction: Explain the direction of entry/exit of a teleporter
10. End of Part I: Complete the given code starting from line 7

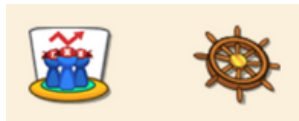
Setting up a workshop

As a teacher or educator, you have the option to create your own workshops in the classroom or during extracurricular hours, using a supervision tool also known as the "Dashboard." The Dashboard allows you to select the lessons to include in the workshop, identify students by their first name or email, and track the progress of each of your students in real-time.

This document provides detailed guidance on how to create your own workshops and take advantage of the features designed specifically for teachers. Workshop Setup Guide

Steps to follow by the teacher

If you have created a teacher account on Coding Park, you should have two additional icons in the horizontal menu.



The first icon provides access to the "Lesson Composer." This tool allows you to configure your workshop in three steps, namely:

1. Choose the lessons to include in the workshop.
2. Identify the students who can join the workshop.
3. Launch the workshop.





Once the workshop is launched, the second icon provides access to the Dashboard. It will be initialized with the workshop's data and will automatically refresh as students progress in the workshop.

Step 1 - Selecting Lessons

Once you click on the "Lesson Composer" button, you will see a split view with two parts. On the left, you have the list of lessons available in the catalog. On the right, you have the configuration tool.

To select the workshop's lessons, simply click on the lessons on the left; they will instantly appear in the configuration tool.

1. Click on the lessons you want to include in the session (maximum 2).

	Getting started: Part 1	
	Getting started: Part 2	


You cannot select more than two lessons per workshop.















Please note that the order of lessons is important: in this example, students will start with the exercises from the "Getting started: Part 1" lesson, followed by "Getting started: Part 2". You can change the lessons by clicking on the (x) button to the right of each lesson in the table.

Step 2 - Student identification

The next step is to identify the students participating in the workshop. The recommended method is to assign an identifier in the form of an alphanumeric string, but generally, a first name is sufficient. In case of homonyms, they should be separated by a number, for example: Alex1, and Alex2.

2. Identify the students you want to invite by their names or e-mails.

Name or e-mail: 

	sofia	Id: sofia@GoldenQuest-20181124142648	
	vlad	Id: vlad@GoldenQuest-20181124142648	
	nathan	Id: nathan@GoldenQuest-20181124142648	
	amine	Id: amine@GoldenQuest-20181124142648	
	cristina	Id: cristina@GoldenQuest-20181124142648	
	jabier	Id: jabier@GoldenQuest-20181124142648	
	adrien	Id: adrien@GoldenQuest-20181124142648	

For each of your students, enter an identifier and click the Add (+) button.

Please note that the maximum number of students per workshop is determined by your subscription. For example, with a Coding Park Club subscription, the maximum number of students is 20. With a Coding Park School subscription, you can set this number according to your needs.

Step 3 - Starting the workshop

It's almost done; just click the Start button to launch the workshop.

3. Click on the button to start the session.

Start

As soon as the workshop is launched, you will see an entry in the "Sessions" table at the top right of the screen. All other fields become grayed out, and it's not possible to modify the content or the list of students once the session has started.



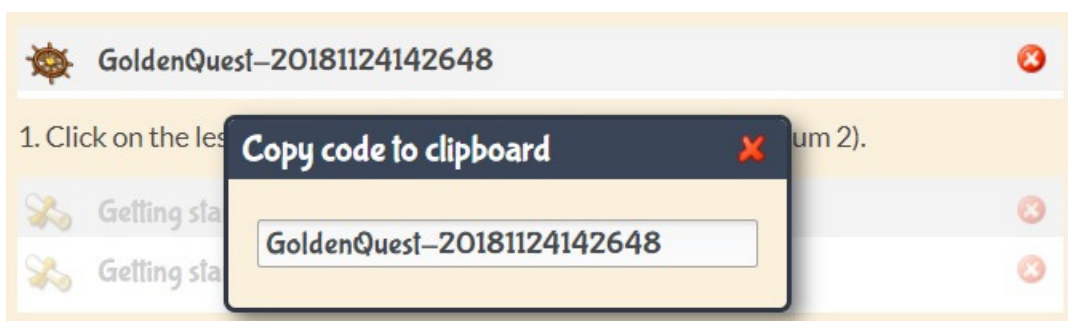
GoldenQuest-20181124142648



If you want to modify the content or the list of students after starting a session, simply delete the session by clicking on the (x) button to the right of the session. Once the session is deleted, the other fields in the configuration tool become accessible again, and you can make the changes you want to implement.

The final step is to retrieve the session code and provide it to each of your students. This code will allow them to join the session from the application without having to create an account beforehand.

To copy the code, click on the session code, then select the text.

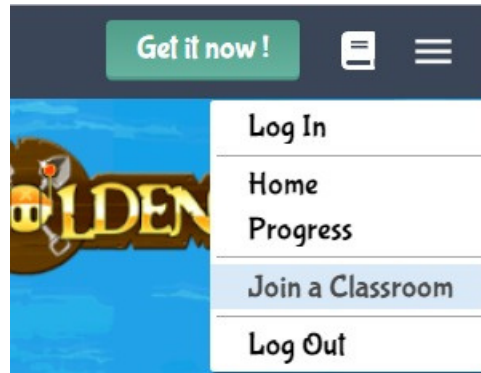


Once the session is launched, the Dashboard initializes with the workshop's data. By clicking the "Dashboard" button, you can already see your students on the starting blocks.

Steps to follow by the students

With the unique identifier you've defined for each student and the session code, students can log in to the session from the public interface. To do this, they simply need to go to <https://app.codingpark.io/goldenquest>.

Then, each student clicks on the menu in the upper right corner (hamburger button) and selects the "Join a Session" option.



An authentication window will open, and the students simply need to enter their identifier and the session code that the teacher provided earlier.

A screenshot of a "Join Group Session" dialog box. The dialog has a dark blue title bar with the text "Join Group Session" and a red close button (X) on the right. The main area has a light beige background. At the top, there is a paragraph of text: "Enter the identifier of the group session that has been sent to you by email. Please check your inbox or contact your inviting teacher." Below this text is a circular logo featuring a cartoon robot character with a red head and yellow body, waving. The robot is surrounded by yellow stars, and the words "Coding Park" are written in a curved path around the robot. Below the logo are two text input fields. The first field contains the text "luna". The second field contains the text "GoldenQuest-20181124142648". At the bottom of the dialog are two green buttons: "Join" and "Cancel".

And there you go! In this example, the student identified as Luna has already embarked on the adventure. She will be provided with a catalog of the lessons selected for the current session. Simply click on the lesson (the treasure chest) to be automatically redirected to the first exercise.

As students progress through the session, their scores are converted into medals.

Simplified Onboarding for Students

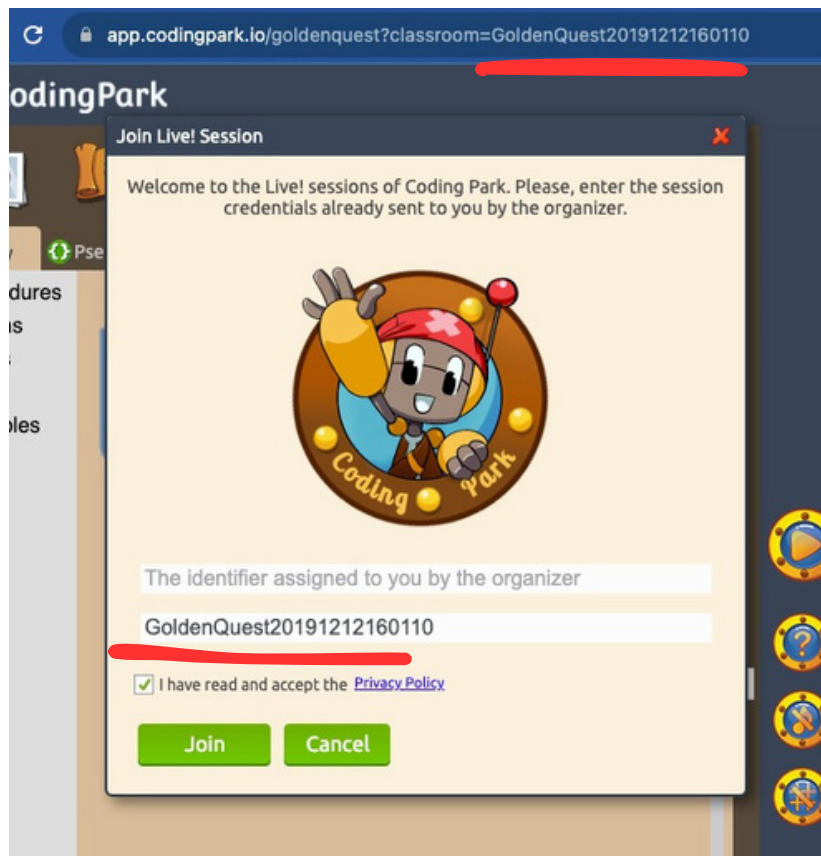
We understand that entering the temporary identifiers and the session ID (GoldenQuestXXXX) can be a bit tricky, especially due to the long character sequence of the session ID. To make the integration process smoother for your students, follow these steps:

- In the browser's address bar, type the following URL:

<https://app.codingpark.io/goldenquest?classroom=GoldenQuestXXXX>

Make sure to replace "GoldenQuestXXXX" with the session ID.

- Press Enter.
- The application will load with the "Join Group Session" dialog box already open and the session ID pre-filled.
- Students only need to enter their temporary identifier and click the "Join" button to get started.
- For your convenience, save this URL in your browser's bookmarks. This way, you can easily access it for future coding sessions.



Automatic Evaluation

As you may have noticed, students' solutions are evaluated in comparison to an optimal solution known in advance. Several parameters come into play in calculating the score, with the most important ones being:

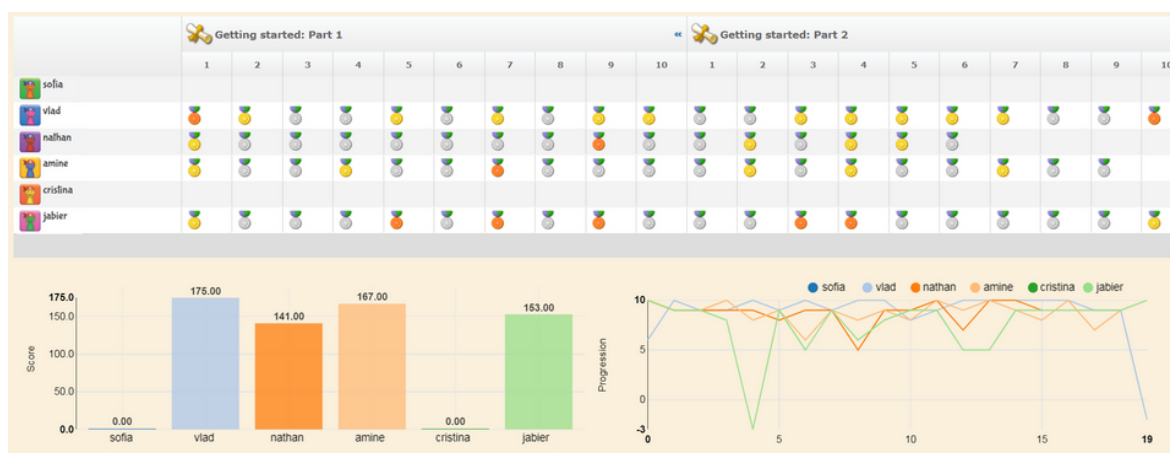
- **Number of Trials/Errors:** The score is weighted by the number of times the student tests their algorithm before arriving at a solution. The first attempt is not counted.
- **Solution Length:** This evaluates the solution in terms of algorithmic complexity and execution performance. For example, writing `jump(3)` is more relevant than writing `jump(1)` three times.

A score out of 10 is assigned to the student's solution, which is then converted into stars (up to 5 stars, with each star representing 2 points). Additionally, students always receive a medal according to the following rules:


- **Gold Medal:** Score of 10 out of 10
- **Silver Medal:** Score between 7 and 9 out of 10
- **Bronze Medal:** Score below 6 out of 10


















On the **Dashboard**, only the medals appear in the grid. By hovering the mouse cursor over a medal, the student's score (out of 10) is displayed.



Lastly, by hovering the mouse cursor over the exercise number, the solution is displayed.

 **Getting started: Part 1**

1	2	3	4	5	6	7	8
							
							
							

```
procedure Cody() {  
  down()  
  right(2)  
  jump()  
  right(4)  
  fight()  
  right(2)  
  dig()  
}
```

Starting the Video Conference

Prerequisites

Before joining a live session with video conferencing, please ensure that your computer does not block access to the camera and microphone.

- Prefer a recent browser: Chrome, Firefox, Edge, Safari.
- Make sure you do not have Skype, Zoom, or Hangout running.
- Check that there is no parental control blocking access to the camera/microphone.

Ensure that the firewall does not block access to the camera/microphone.

If you are using Chrome:

- Type `chrome://settings` in the address bar.
- Type camera in the search bar.
- Go to Site settings > Camera.
- Make sure the "Ask before accessing" option is selected.
- Type microphone in the search bar.
- Go to Site settings > Microphone.
- Make sure the "Ask before accessing" option is selected.

If you are using Firefox:

- Type `about:preferences`.
- Type camera in the search bar.
- Go to Permissions, next to Camera click on Settings...
- Make sure the "Block new requests asking to access your camera" option is unchecked.
- Click Save Changes.
- Type microphone in the search bar.
- Go to Permissions, next to Microphone click on Settings...
- Make sure the "Block new requests asking to access your microphone" option is unchecked.
- Click Save Changes.

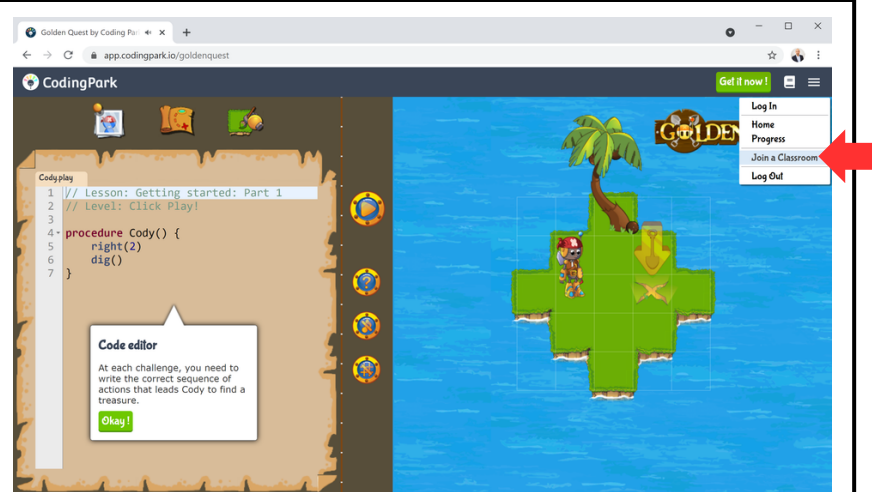
If you are using a Mac:

- Choose the Apple menu > System Preferences.
- Click on Security & Privacy, then Privacy.
- Select Camera.
- Check the box next to the browser to allow it to access the camera.
- Select Microphone.
- Check the box next to the browser to allow it to access the microphone.

Join a Session

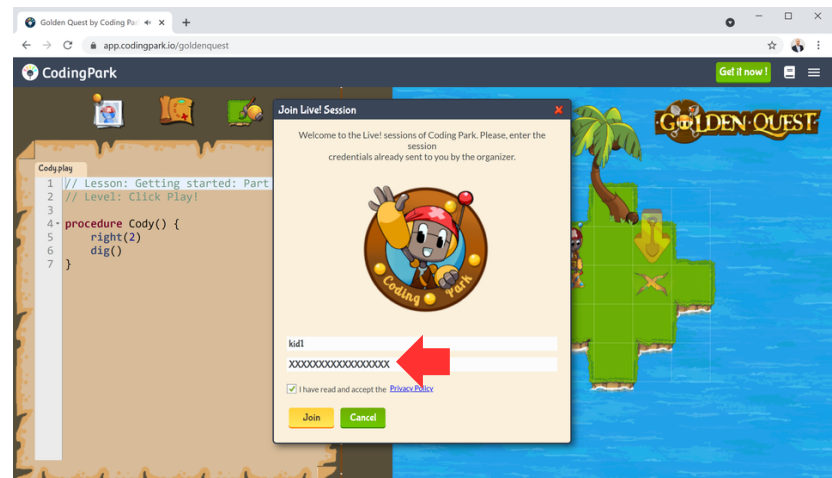
Step 1

Click on the menu at the top right and select "Join a session."



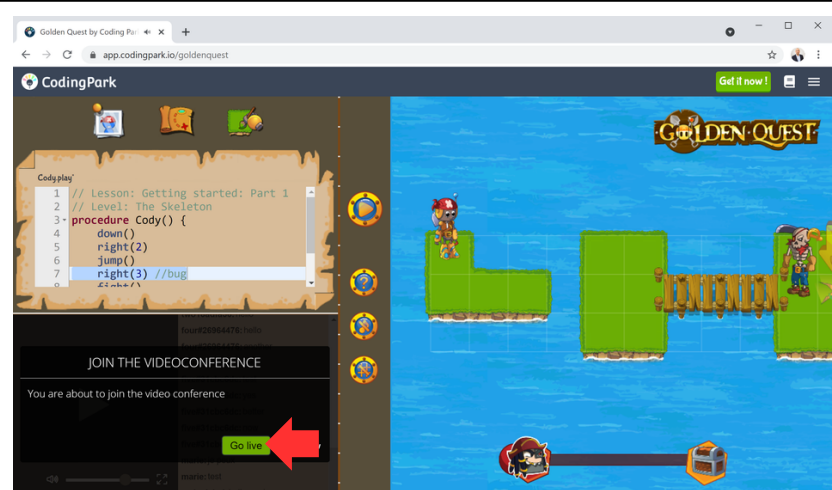
Step 2

Enter the student's username and the session code, accept the terms and conditions, and then click the "Let's go!" button.



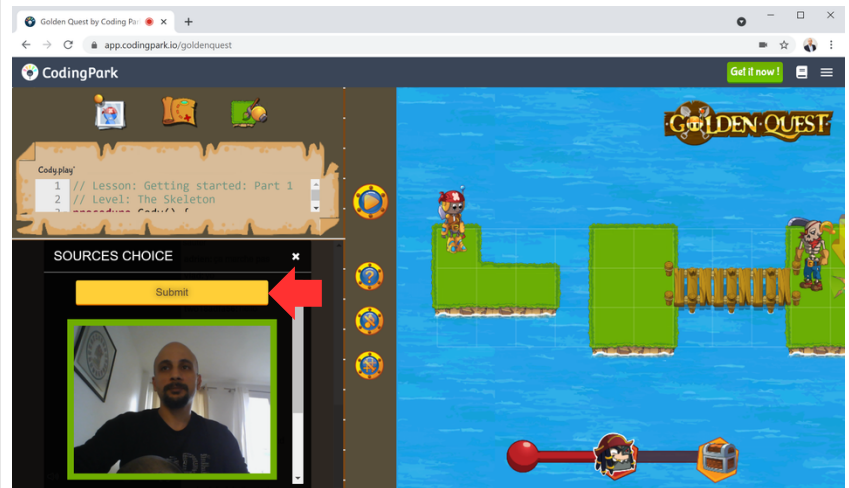
Step 3

Once authenticated, click "Join Now" in the video conference area. This area can be resized by dragging the edges from bottom to top and from left to right.



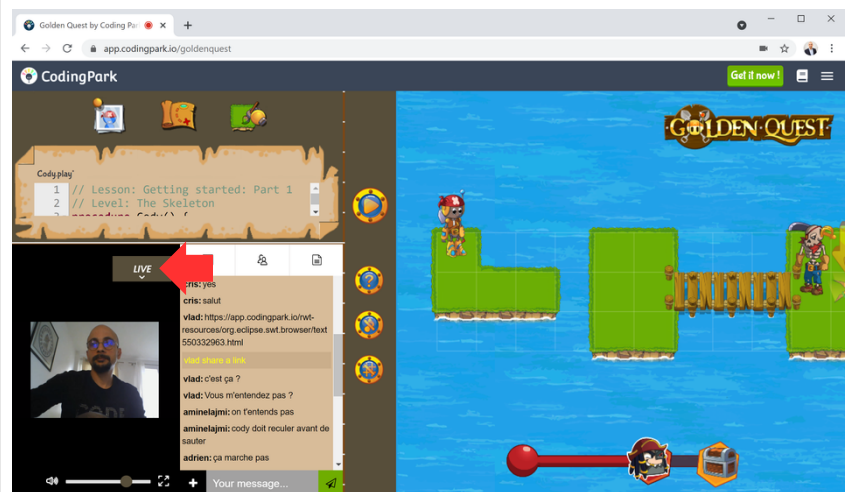
Step 4

Check that the camera and microphone are working, click the "Submit" button, and you're good to go!



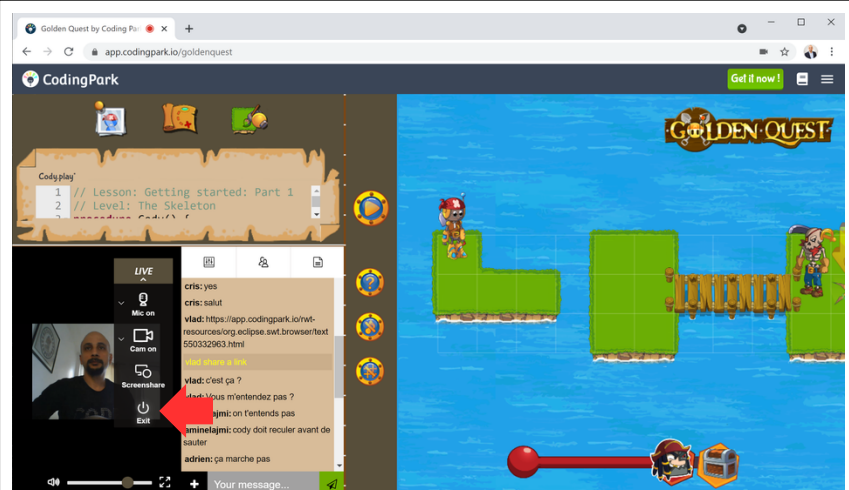
Step 5

During the live session, click the "Live" button to enable or disable the microphone or camera or to request screen sharing.



Step 6

Once the live session is over, click the "Exit" button.



Contact Us

We are here to assist you, so please feel free to send us an email at support@codingpark.io for any questions regarding the platform's operation. Additionally, we welcome any suggestions that could enhance the user experience, automatic assessment, or any other related areas that can make the platform better from an educational perspective.

Contact Us

Annex

Pedagogical Guides



Getting Started - Part 1

Objectives

- ✓ Get hands on the code editor and learn how to use the debugger
- ✓ Understand the syntactical rules of the pseudo language
- ✓ Understand how to move Cody on the grid

Prerequisites

- ✓ At least one computer for every two students and a good Internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must have joined the session with their session ID

Lesson Plan

1. Click Play: Click on Play and describe what Cody does
2. Golden Tooth: Write down() at line 7
3. The Jumpy Frog: Write jump(3) at line 10
4. The Bushes: Introduce the notion of "bug" and fix bug at line 8
5. The Skeleton: 2 bugs at line 7 and 9, use the appropriate parameters
6. The Snake: Briefly introduce the notion of loop, replace 3 by 8
7. The Dreadful Five: Add right() statement at line 5
8. The Teleporters: Fix the bug at line 4, replace 6 with 5
9. The Teleporter Direction: Explain the direction of entry/exit of a teleporter
10. End of Part I: Complete the given code starting from line 7

Our Advices

- Start by explaining what computer programming is in simple terms
- Introduce the objective of the game: lead Cody, the pirate robot, to the treasure
- Ask the students to always (1) test the initial code given to them, then (2) observe what is wrong, and finally (3) correct or complete the given code
- Emphasize the pronunciation and spelling of the basic actions
- Let students progress at their own pace

Tips

- To jump in a given direction, Cody has to gain momentum
- The exit direction of a teleporter is the same as the entry direction

The most common student mistakes

- Forget the parentheses after an action keyword or a function name
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Getting Started - Part 2

Objectives

- ✓ Get hands on the code editor and learn how to use the debugger
- ✓ Understand the syntactical rules of the pseudo language
- ✓ Understand the base actions to move Cody on the grid

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID

Lesson Plan

1. The Hat: Fight one pirate at a time, starting with the one on top
2. The Bandits: Dodge the pirate group by using the teleporter
3. Take a little run: Take momentum before the jump by going to the left
4. Jumping: Go around the island in order to have the run-up in the right direction
5. The Button: Use `pushButton()` function to activate a teleporter
6. Several Buttons: Deactivate the teleporters, jump, then activate them again
7. Deactivate and Activate: First fight the pirate to create the path
8. Calling Procedures: Call procedure `L()` at line 8
9. Calling with Parameter: Write `fightLine(...)` at line 7 to beat the last row
10. End of Part 2: Turn on the colored teleporters to reach the treasure

Our Advices

- Remind the students the notions learned in the previous session
- Recall how to insert a line break in the code editor using the "Return" key
- Recall the base keywords seen so far (up, down, left, right, jump, fight, etc.)
- Let the students' elaborate their solutions and encourage them to help each other

Tips

- To jump into one direction, Cody must gain momentum, like in real-life
- To activate/deactivate a teleporter, press the button of the same color

The most common student mistakes

- Forget the parentheses after an action keyword or a function name
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Getting Started - Part 3

Objectives

- ✓ Learn how to combine actions to solve more elaborated problems
- ✓ Know how to solve an “inverse exercise”, i.e., read code and draw island
- ✓ Write more than 15 lines of code in complete autonomy

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session identifier

Lesson Plan

1. Double Teleportation: Take the yellow teleporter in the appropriate direction
2. Little Antarctica: Go the yellow teleporter and take it back in the other direction
3. The Not Complicated: Go around over the small rocks on the left-hand side
4. Happy Winter: Take the yellow teleporter to get to the treasure
5. Between The Trees: Take the purple teleporter then the orange one
6. The White Snail: A blue teleporter is hidden behind the first tree
7. The Tetris Island: Take the right path by going straight down
8. The Good Choice: Go to the blue teleporter at the top of the island
9. The Four Corners (inverted): Use the Level Editor to draw the minimal path
10. The Special One: Take the blue teleporter and turn it off to get to the yellow one

Our Advices

- Remind the students how to insert a new line in the code editor using Return key
- Let the students discuss how they approach the problem before writing any code
- These levels are more sophisticated, reassure the students about their progress
- Let students progress at their own pace, whether in pairs or individually

Tips

- When two teleporters are side by side, it is possible to go directly from one to the other: when leaving the first one, Cody jumps directly into the second one
- In a reverse exercise, e.g. The Four Corner, you have to draw the minimal path leading Cody to the treasure. The more approximate the path, the lower the score

The most common student mistakes

- Forget the parentheses after an action keyword or a function name
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Procedures

Objectives

- ✓ Start addressing the basics of computer programming
- ✓ Understand what a procedure is and how to call it
- ✓ Understand the mapping between visual patterns and code blocks

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID

Lesson Plan

1. The Shapes: Call `otherShape()` procedure at line 7
2. The Tree: Reverse procedure calls at lines 5 and 6
3. The Big Fight: Call `bigFight(...)` procedure at line 6
4. Two Palmtrees: Replace `down()` at line 11 by `up()`
5. The Button: Complete the procedure at line 12
6. Broken Bridges: Exchange procedure calls at lines 6 and 7
7. Black Box Island: Call `blackBox(...)` procedure at line 6
8. Islands: Call procedures at lines 7/8 and complete line 25
9. Zigzag (inverted): Draw the minimum path using the Level editor
10. Procedures: Put the correct procedure calls at lines 7 and 8

Our Advices

- Explain the concept of a procedure (or function) at the beginning of the lesson
- "To call a procedure" means to write that procedure name at a particular place
- Provide guidance to students especially on the first two levels, let them ask Luna
- It is sometimes necessary to change several things, bugs are indicated in the code editor as comments `// bug`
- In many islands, there is a mapping between the code and the visual patterns, reason by analogy with what is already written

Tips

- Cody can't fight and jump simultaneously; he has to either fight the enemies upstream or find another way

The most common student mistakes

- Forget the parentheses after an action keyword or a function name
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Parameters

Objectives

- ✓ Understand what a procedure parameter is and why it's used
- ✓ Understand the notation of parameters in a procedure definition
- ✓ Know how to call a procedure and affect values to parameters

Prerequisites

- ✓ At least one computer for two students and a good internet connection
- ✓ The creation of a session beforehand by the teacher / facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have previously explained the concept of procedures

Lesson Plan

1. Parameter: Put a 4 instead of 3 for the snake() function at line 6
2. Snakes: Add the function snake() at line 5 and 6 with the correct parameters
3. Lines: Add line() function at line 6 and 7 with the correct parameters
4. The Ls (inverted): Draw the minimal path using the Level editor
5. The Ls 2: Add the shape() function to line 7 with the correct parameters
6. The Nested Snakes: Complete lines 13, 14 and 15 with the correct parameters
7. The Cs: Complete the procedure that matches the islands on the right
8. Teleporters: Add island() to line 5, 6, 7 with the correct parameters
9. The Intestine: Add the function p() to line 7 and 8 with the correct parameters
10. The Lines: Fix the bug at line 15 by replacing 2 by x

Our Advices

- Explain the abstract notation of parameters x, y, z, ..., this notion acts as a placeholder that takes on the numerical value given to it
- Take time to test the initial code, most of the time you just need to make the appropriate procedure call
- Show the mapping between the names of procedures and the visual patterns

Tips

- Take into consideration the input/output direction of the teleporters

The most common student mistakes

- Forget the parentheses after an action keyword or a function name
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Loops - Repeat

Objectives

- ✓ Approach repetitions loops more in depth
- ✓ Understand how code works inside a repeat loop
- ✓ Know how to associate a visual pattern to an iteration of a loop

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a session by the teacher/facilitator
- ✓ All students must be logged in with their session ID

Lesson Plan

1. Basic Repeat: Fix the bug at line 7 by setting the appropriate value
2. Bushes: Write the appropriate code to browse the first Island
3. Repeat: Each repeat(...) loop is equivalent to one half of the Island
4. Skeletons: Each repeat loop(...) corresponds to a row of pirates
5. Nested Repeat: The 1st repeat(...) stands for the number of rows of pirates
6. Stairways (inverted): Draw the minimum path with the Level editor
7. Skeletons: The 1st repeat(...) corresponds to the number of rows of pirates
8. The W: Modify the repeat(...) loops at lines 5 and 11
9. Two Loops: Complete the second repeat loop at line 15
10. Battle: Write fight() at line 7 and 13, and right() at line 8 and 14

Our Advices

- Recall the first lesson where repetition was quickly introduced (The Snake)
- There are two types of loops: the repeat(...) loop and the while(...) loop. The repeat(...) loop is used when the number of repetitions is known in advance. The while loops will be covered in the next lesson
- Remember the modification to be made is often simple and indicated by `//bug`

Tips

- Cody always attacks a pirate located in one square next to him
- Cody can attack one pirate at a time

The most common student mistakes

- Write the solution in the wrong repeat(...) loop
- Forget the parentheses after the repeat keyword
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Variables - Part 1

Objectives

- ✓ Discover variables and their role in a computer program
- ✓ Understand how to declare a variable
- ✓ Understand how to assign a value to a variable
- ✓ Understand how to use variables through simple puzzles

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the Parameters lesson

Lesson Plan

1. The Increment Island: Put $x+2$ between parentheses at line 10
2. The Variable Bandits: Increment x by 1 ($x=x+1$) at line 15
3. The Three Arms (inverted): Draw the minimal path using the Level editor
4. The Three Arms Repeated: Increment x by 1 ($x=x+1$) at line 15
5. Take the Stairs: Add step(...) at line 11 and 12 with the appropriate parameter

Our Advices

- Recall the Parameters lesson, in particular how parameters are represented by abstract names (x, y, z, \dots). The same applies to a variable which acts as a placeholder that takes the value, we assign to it at a given time, should it be a numerical value or an arithmetic expression
- At the beginning of the lesson, let students focus on the visual patterns and the symmetrical aspect of the islands
- The change to be made is often simple and almost systematically indicated as a comment (in green) and most often, a simple bug to fix
- Explain the notion of increment, e.g., $x=x+1$ is read "x takes the current value of x increased by 1"

The most common student mistakes

- A variable declaration without the keyword var
- Forget the parentheses after the repeat keyword
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Variables - Part 2

Objectives

- ✓ Discover variables and their role in a computer program
- ✓ Understand variable scope (global vs local)
- ✓ Understand how to use variables inside a repeat/while loop
- ✓ Understand how to use variables through simple puzzles

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the Variables- Part 1 lesson.

Lesson Plan

1. Variable: Assign the value 4 to the variable step at line 14 (step = 4)
2. Addition: The island is here split into 3 parts, add down() on line 16
3. The Spiral: Between two down(), the snake decreases of 4 squares, put $x=x-4$
4. Global Variable: In this level Cody executes each function 3 times
5. Same Name: This level is explanatory, explain what is going on
6. Parameter Name: This level is explanatory, explain what is going on
7. Limited Scope: This level is explanatory, explain what is going on

Our Advices

- Remind the students "Variables -Part 1" lesson, practiced earlier, especially the two important concepts: (1) how to declare a variable, and (2) how to assign a value to a variable
- In Same Name level (variable scope), ask students how many steps Cody goes down before hitting play button. Try to change the value of steps (1) globally at line 5 and (2) locally at line 10 and make students guess the execution result
- In Parameter Name, ask students to guess what would be the number of squares Cody goes to the right. It's the 5 in the procedure call bridge(5) at line 8, so the parameter variable has more priority than the global variable steps at line 5
- The modification to be made is simple and indicated as comments `// bug`

The most common student mistakes

- Thinking the = sign means equality, while it's an assignment, e.g., $x=x+1$
- A variable declaration without the keyword var
- Forget the parentheses after the repeat keyword
- Mis-spell an action keyword or a function name
- Delete by mistake a bracket "{" or "}"



Loops: While

Objectives

- ✓ Understand the operation of a while loop(...)
- ✓ Understand the stop condition of a while(...) loop
- ✓ Know how to use the while(...) loop to solve a problem

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the "Loops: Repeat" and "Variables - Part 1 and 2" lessons

Lesson Plan

1. The While: Replace the value 4 by 5 at line 5
2. Minus One: The variable x will be used only for the right(...) action
3. Plus One: Write $x=x+1$ to increment the value of x by 1
4. The Boa Island: The snake decreases by 2 squares, decrease x by 2: $x=x-2$
5. Teleporters: The variable x will only be used for the jump(...) function
6. Skeletons: The variable x will only be used for the right(...) function
7. The Race: The variable x will be used for jump(...), increase x by 1: $x=x+1$
8. Multiplication (inverted): Draw the minimal path using the Level Editor
9. Spaceships: The variable x is used only for the functions right(...) and left(...)
10. The last While (inverted): Draw the minimal path using the Level Editor

Our Advices

- Explain the difference between repeat(...) and while(...): in the former, the number of repetitions is known in advance (exit from the loop after n iterations), while in the latter, a Boolean exit condition is evaluated at each iteration to determine whether or not the program exits the loop
- The change to be made on code is often simple and almost always indicated with a `//bug` comment; it is most often a simple bug to correct

Tips

To activate/deactivate a teleporter, press the button with the same color

The most common student mistakes

- Unintentional deletion of brackets or parentheses after a procedure name or inside a repeat/while loop
- Forgot the line that increments a variable (ex: $x = x + 1$) inside a loop



Conditionals

Objectives

- ✓ Discover conditional expressions in computer programming
- ✓ Understand the logical structure of a block: `if (condition) { ... } else { ... }`
- ✓ Know how to formulate conditional expressions to solve a problem

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the "Variables - Part 1" and "Variables - Part 2" lessons

Lesson Plan

1. Conditional: Put the value 4 inside the if condition, to fight at the right time
2. The Button: Put the right value where the bug is indicated, then press the button
3. Skeletons: Put the right value at the place of the bug, to jump at the right time
4. Not Equal: It is not necessary to increase by 1 at line 9
5. Greater or Equal: Put the value 3 in the if condition to be able to fight first
6. Else if: Replace the value that caused the bug
7. Gold: The buttons are placed on the 3rd and 5th columns
8. Teleporters: The `jump(...)` function is used except for the value 3
9. And: It is necessary to fight between the values 2 and 6
10. End of Conditionals: Add an if condition to be able to press the button

Our Advices

- Explain the structure of an if block: `if (condition) { ... } else { ... }`
- At the beginning of the lesson, highlight similarities with previous lessons
- Explain how to write conditions using Boolean expressions with operators `>`, `<`, `==`

Tips

- To jump into one direction, Cody must gain momentum

The most common student mistakes

- Forget parentheses/parameters after a procedure call
- Delete by mistake the condition parenthesis in the if block
- Remove brackets `{` or `}` after the if condition or after the else block
- Wrong writing of a symbol, ex: `<>`, `==`, etc. inside the condition of the if block



Puzzles

Objectives

- ✓ Master the concepts learned so far (procedures, repeat, while, if)
- ✓ Exploit these notions to solve levels with advanced algorithmics
- ✓ Know how to write more or less complex pieces of code in complete autonomy

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the "Loops: While" and "Loops: Repeat" lessons

Lesson Plan

1. Chess: Once you find the right path, just jump to get to the treasure
2. The Pyramid: You have to fight against the rows while moving diagonally
3. The Bridges: Call Go function at lines 5, 6 and 7 with the appropriate parameters
4. Vertical and C's: The names of the functions correspond to the path taken by Cody
5. The Squares: You have to reach the buttons and then take the right paths
6. The Horde: You have to beat the skulls by columns using the repeat(...) function
7. Snake Twins: a while block must be created, the two islands are built in mirror

Our Advices

- Explain to children that most of the problems are logic problems
- Be patient, these levels require more logical thinking than the previous lessons
- For more complex levels (level 2 and 3), show that the required solution is often easy to write/correct

Tips

- To jump into one direction, Cody must gain momentum
- To activate/deactivate a teleporter, press the button with the same color
- The exit direction of a teleporter is important: the direction taken when entering a teleporter is the same as the direction taken when leaving the teleporter

The most common student mistakes

- Delete by mistake some blocks, such as variable declarations or assignments
- Unintentional deletion of brackets or parentheses in a procedure call, or inside a block of code (while, repeat, if, ...)
- Since geometric shapes are simple, children tend to work around the problem by deleting the initial code given in the statement



Loops: Nested

Objectives

- ✓ Master the notions learned about repeat(...) and while(...) loops
- ✓ Know how to nest a repetition loop inside another one
- ✓ Know how to declare and increment variables inside loops

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the lessons “Loops: Repeat” and “Loops: While”
- ✓ Have already practiced the lessons “Variables - Part 1 & 2” and “Procedures”

Lesson Plan

1. Nested Whiles: Move x spaces to the right and y spaces down
2. Nested Whiles 2: Same principle, but the moves are different
3. Nested Whiles 3: Here, the islands decrease in size instead of increasing
4. Nested Whiles 4: This is a mix of the three exercises seen previously
5. Skeleton: Two repeat(...) loops: one for vertical fight, one for horizontal fight
6. The Arrow: Two repeat(...) loop: one for ascent, one for descent

Our Advices

- Show the students the slight evolution of the island's geometry at each level
- Recall the notion of variables, and explain that x and y often represent the move along vertical and horizontal axis of each island

Tips

- To activate or deactivate a teleporter, press the button of the appropriate color
- The exit direction of a teleporter is important: the direction taken when entering a teleporter is the same as the direction taken when leaving the teleporter

The most common student mistakes

- Unintentional deletion of brackets or parentheses in front of a procedure name, or inside a code block (while, repeat, if, ...)
- Confusion between the horizontal and vertical axis of an island
- The visual patterns being simple, students tend to work around the problem by deleting the initial code given to them in the statement



Maths

Objectives

- ✓ Make a synthesis of all the notions acquired so far
- ✓ Use mathematical expressions in a program
- ✓ Know how to identify visual patterns and associate them with functions

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced the "Procedures" and "Variables - Part 1 & 2" lessons

Lesson Plan

1. Multiplication: To complete the zeta(...) function, observe the ratio "number of steps to the right" / "number of steps down", deduce that the number of steps down is $2 \times x$
2. Division (inverted): Draw the minimum path with the Level Editor
3. The Third (inverted): Draw the minimum path with the Level Editor

Our Advices

- Remind students the "Variables -Part 1" lesson practiced earlier, especially (1) how to declare a variable, and (2) how to assign a value to a variable
- At the beginning of the lesson, observe the repeated visual pattern and explain that each pattern is traversed by the function zeta(...)
- When calling zeta($2 \times x$) function, the $2 \times x$ multiplication is evaluated before the actual function call. As a general rule, the precedence of operators/functions is always performed from right to left

Tips

- The jump() action must necessarily appear in the function that applies to the visual patterns (for example in the zeta(...) function of the first level) because this instruction allows to move from one pattern to the next one

The most common student mistakes

- Unintentional deletion of brackets or parentheses in front of a procedure name, or inside a code block (while, repeat, if, ...)
- The visual patterns being simple, students tend to work around the problem by deleting the initial code given to them in the statement



Recursion

Objectives

- ✓ Review all the notions and concepts learned so far
- ✓ Understand the concept of recursion in a program
- ✓ Have a visual intuition of recursion
- ✓ Illustrate recursion through a Fibonacci sequence

Prerequisites

- ✓ At least one computer for every two students and a good internet connection
- ✓ The creation in advance of a coding session by the teacher/facilitator
- ✓ All students must be logged in with their session ID
- ✓ Have already practiced all the lessons in the catalog before this one

Lesson Plan

1. Recursion: Just call the function `recursion(...)` with the parameter $x+1$
2. This level is explanatory, no modification is necessary, it serves as an introduction to the Fibonacci sequence. Here we stop at the 7th element of the sequence

Our Advices

- Take the time to introduce the concept of recursion with simple examples
- Explain that recursion can be seen as a kind of abyss
- To illustrate recursion, use the "Laughing Cow" logo, whose earrings contain the image of the Laughing Cow, whose earrings contain the image of the Laughing Cow, whose earrings contain the image of the "Laughing Cow", etc.
- A computer program cannot support an infinite number of recursive calls, it gets stuck (bug) because it cannot chain function calls indefinitely. For this reason, we have to write an exit condition that should be checked systematically by the function to decide whether to continue or not
- Fibonacci sequences are a good example of recursion, use examples from nature, show in nature, show pictures of plants, shells, etc.
- Get inspiration from the paintings of Maurits Cornelis Escher
- In this Fibonacci sequence, an element is equal to the sum of the two previous elements. Unfold on a white board what happens starting from two elements $\text{fibonacci}(0) = 1$ and $\text{fibonacci}(1) = 1$. The next element $\text{fibonacci}(2) = 1 + 1 = 2$. The next element $\text{fibonacci}(3) = 2 + 1 = 3$, etc.

The most common student mistakes

- The visual pattern being simple, the children tend to work around the problem by deleting the code initially given to them in the statement