

# nclab

More Than Coding

## How to Teach Karel Coding

<http://nclab.com>



# The purpose of this presentation

Why start with Karel?

What will my students learn?

- How will they learn it?
- How are they assessed?

How does Karel fit into my curriculum?

What else is there beside the course?



# Why start with Karel?

Before tackling complex languages, computations, and tasks, students need to develop **logical reasoning**.

Language and logic,  
using simplified script

Karel works with both  
sides of the brain:

Visual/spatial/kinetic/auditory  
instant feedback, using mazes

```
Lines: 20      Objects to collect: 1
Use: get, go, if, left, rand, right, rug

1 while empty
2   | if rand
3   | | if not wall
4   | | | go
5   | else
6   | | if wall
7   | | | right
8   | | else
9   | | | left
10  | if rug
11  | | get
```

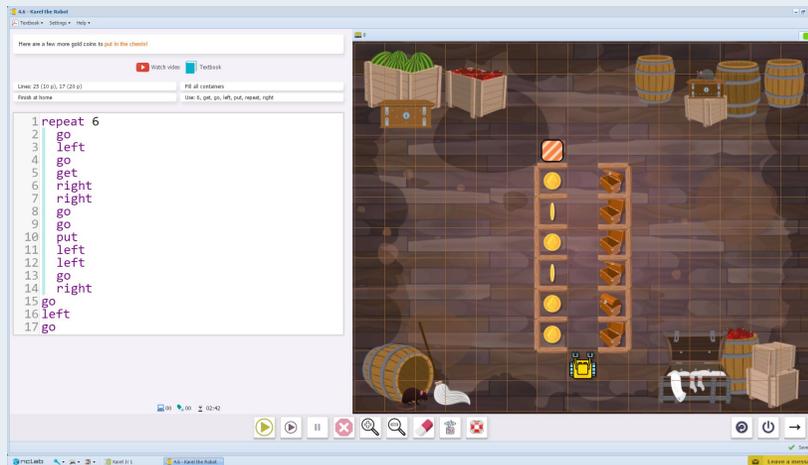
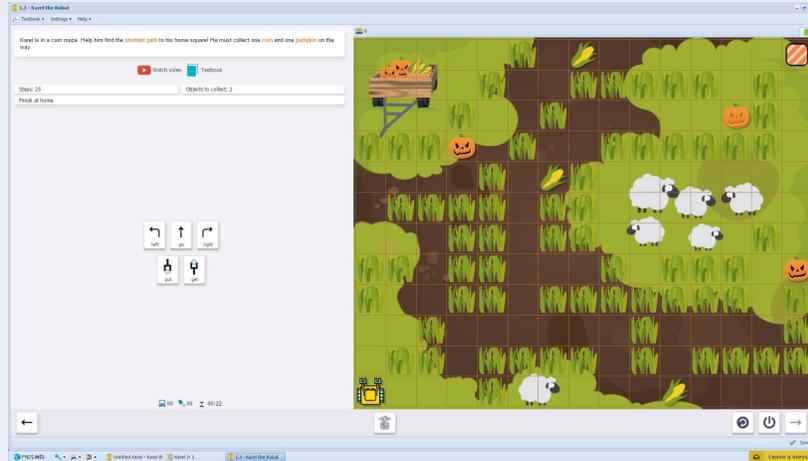


# What do my students learn?

Beginning level skills - quick to learn

**UNIT 1:** Students learn how to guide the robot, type simple programs, recognize repeated patterns, and use the repeat loop.

At the end of this unit, they should be able to create their own mazes with features such as nested loops.



# What will my students learn?

## Intermediate level skills

**UNIT 2:** Students will learn how to use **if-else conditions**, the **while loop**, and how to combine loops and conditions together.

**UNIT 3:** Students will learn how to use **custom commands**, local and global **variables**, and **functions** that return values.

The image displays two screenshots of the Scratch 'Karel the Robot' environment. The top screenshot shows a jungle scene with a toucan and a small robot on a grid. The code editor contains the following code:

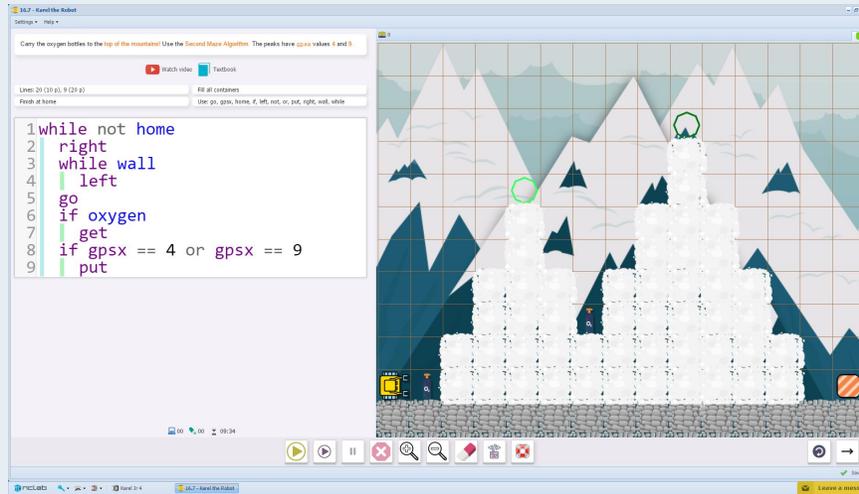
```
1 repeat 8
2   if plant
3     right
4     go
5     left
6     go
7     go
8     left
9     go
10    right
11    go
```

The bottom screenshot shows an underwater scene with a shark and pearls. The code editor contains the following code:

```
1 # Move along the wall, so that
2 # it is on your right. Collect
3 # a pearl if you find one:
4 def move
5   if pearl
6     get
7     right
8     while wall
9       left
10    go
11
12 # Main program:
13 while not home
14   move
```

# What will my students learn?

Advanced level skills



21.2 - Karel the Robot

Settings Help

Carry the oxygen bottles to the top of the mountain! Use the Second Maze Algorithm. The peaks have `gpsx` values 4 and 9.

Match video Feedback

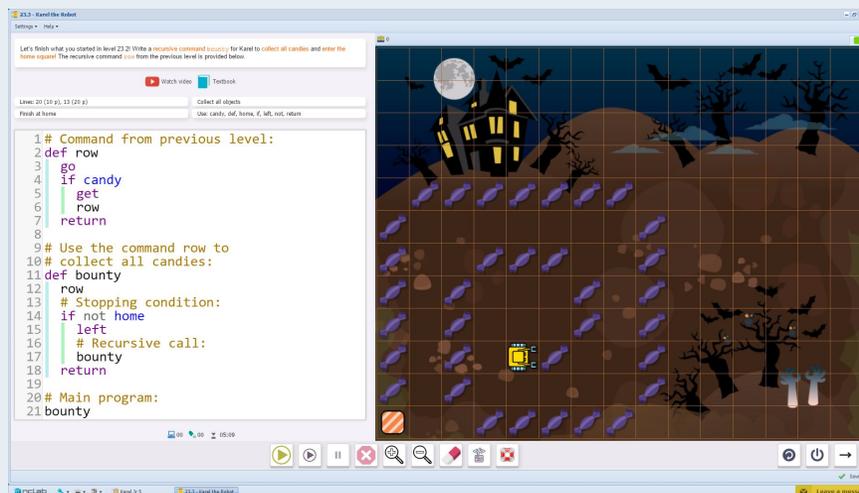
Level: 20 (20 p), 9 (20 p) 68 of 100 answers

Fetch all hints Use: `go`, `gpsx`, `home`, `f`, `left`, `not`, `or`, `put`, `right`, `wait`, `while`

```
1 while not home
2   right
3   while wall
4     left
5   go
6   if oxygen
7     get
8   if gpsx == 4 or gpsx == 9
9     put
```

00:00 09:34

**UNIT 4:** Students will learn how to use **GPS** coordinates, comparison symbols, Boolean values, and random variables.



21.3 - Karel the Robot

Settings Help

Let's finish what you started in level 22.2! Write a recursive command `bounty` for Karel to collect all candies and enter the home square! The recursive command `row` from the previous level is provided below.

Match video Feedback

Level: 20 (20 p), 22 (20 p) Collect all objects

Fetch all hints Use: `candy`, `def`, `home`, `f`, `left`, `not`, `return`

```
1 # Command from previous level:
2 def row
3   go
4   if candy
5     get
6   row
7   return
8
9 # Use the command row to
10 # collect all candies:
11 def bounty
12   row
13   # Stopping condition:
14   if not home
15     left
16   # Recursive call:
17   bounty
18   return
19
20 # Main program:
21 bounty
```

00:00 05:09

**UNIT 5:** Students will learn how to make random decisions, use recursion, and solve advanced programming challenges.

# How do my students learn?

In this Section you will learn how to guide Karel by **typing commands**. Before you go further, watch this short video:

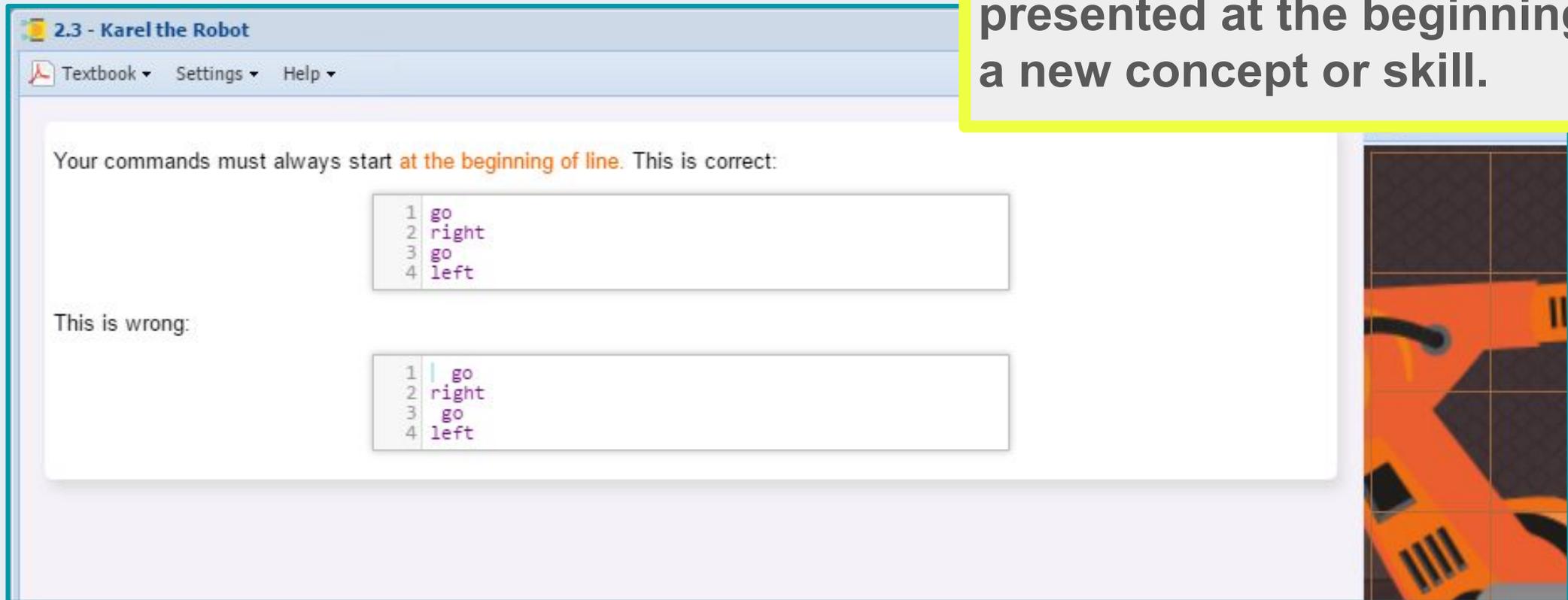


YouTube link: <http://youtu.be/s4Ewl1p2wX0>

**Instructional YouTube videos, presented at the beginning of a new concept and available any time.**

# How do my students learn?

Instructional screens,  
presented at the beginning of  
a new concept or skill.



The screenshot shows a window titled "2.3 - Karel the Robot" with a menu bar containing "Textbook", "Settings", and "Help". The main content area contains the following text and code examples:

Your commands must always start **at the beginning of line**. This is correct:

```
1 go
2 right
3 go
4 left
```

This is wrong:

```
1 | go
2 right
3 go
4 left
```

The code examples are displayed in a monospaced font within a light gray box. The first example shows commands starting at the beginning of each line, while the second example shows a vertical bar at the start of the first line, indicating an incorrect format.

# How do my students learn?

Step by step  
demonstration levels.

The screenshot displays the 'Karel the Robot' programming environment. The interface is divided into several sections:

- Instruction Panel (Top Left):** Contains a play button and text: "Your first task is just to step through the program below by clicking on the stepping button." Below this, it states: "The code cannot be changed. Click slowly and watch what every line does! Also watch the counters of steps and operations on bottom left. Can you figure out the difference between a step and an operation?" There are buttons for "Watch video" and "Textbook".
- Code Editor (Middle Left):** A list of seven commands:

```
1 right
2 go
3 left
4 go
5 go
6 right
7 go
```

The third line, "3 left", is highlighted in blue.
- Simulation Area (Right):** A grid-based environment with a robot (Karel) in the center. The environment includes a workbench with various tools (wrenches, screwdrivers, pliers, hammers), a toolbox, a power outlet, a laptop, and a green square object.
- Control Panel (Bottom):** Includes a "Finish at home" dropdown menu, a "Use: go, left, right" input field, and a set of navigation buttons (back, play, stop, zoom in, zoom out, refresh, and forward).
- Status Bar (Bottom):** Shows "02 01 04:15" and a "Leave a message" button.

# How do my students learn?

There are 5 units in Karel

UNIT

Each unit is made up of 5 sections

SECTION

A section represents a concept arc.

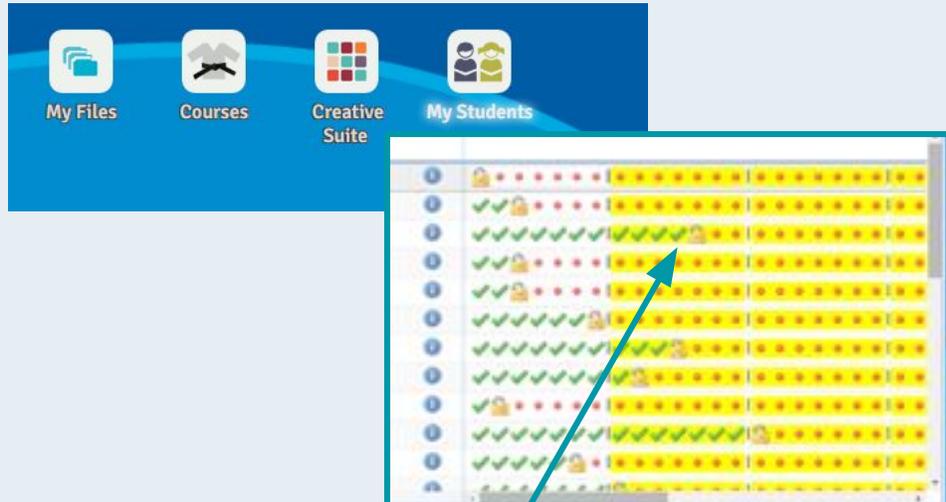
Each section is made up of 7 levels

LEVEL

A level represents one step of instruction or practice within the concept arc.

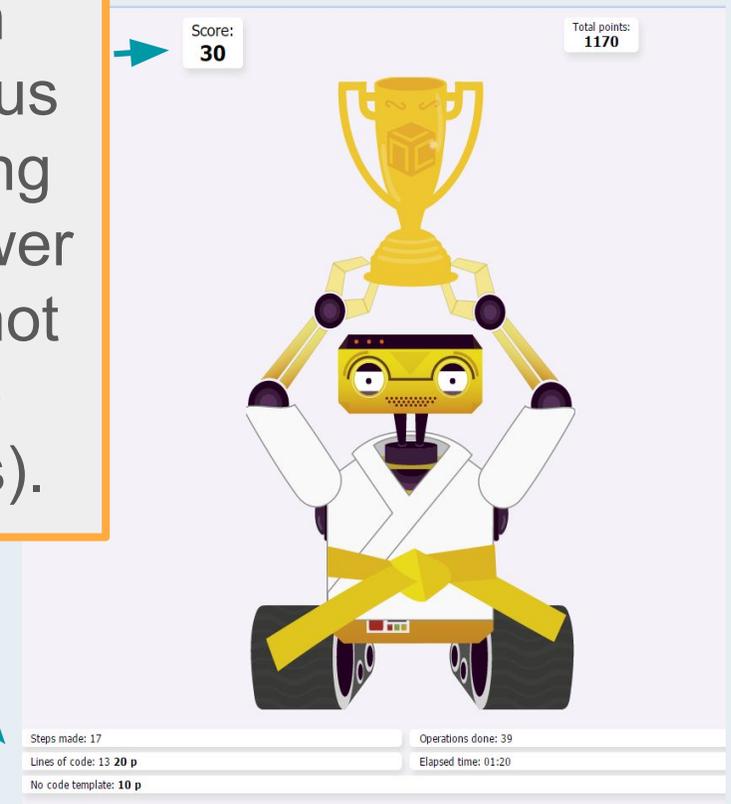
**Systematic instruction and practice with gradual release - 175 levels in all.**

# How do I monitor and assess my students?

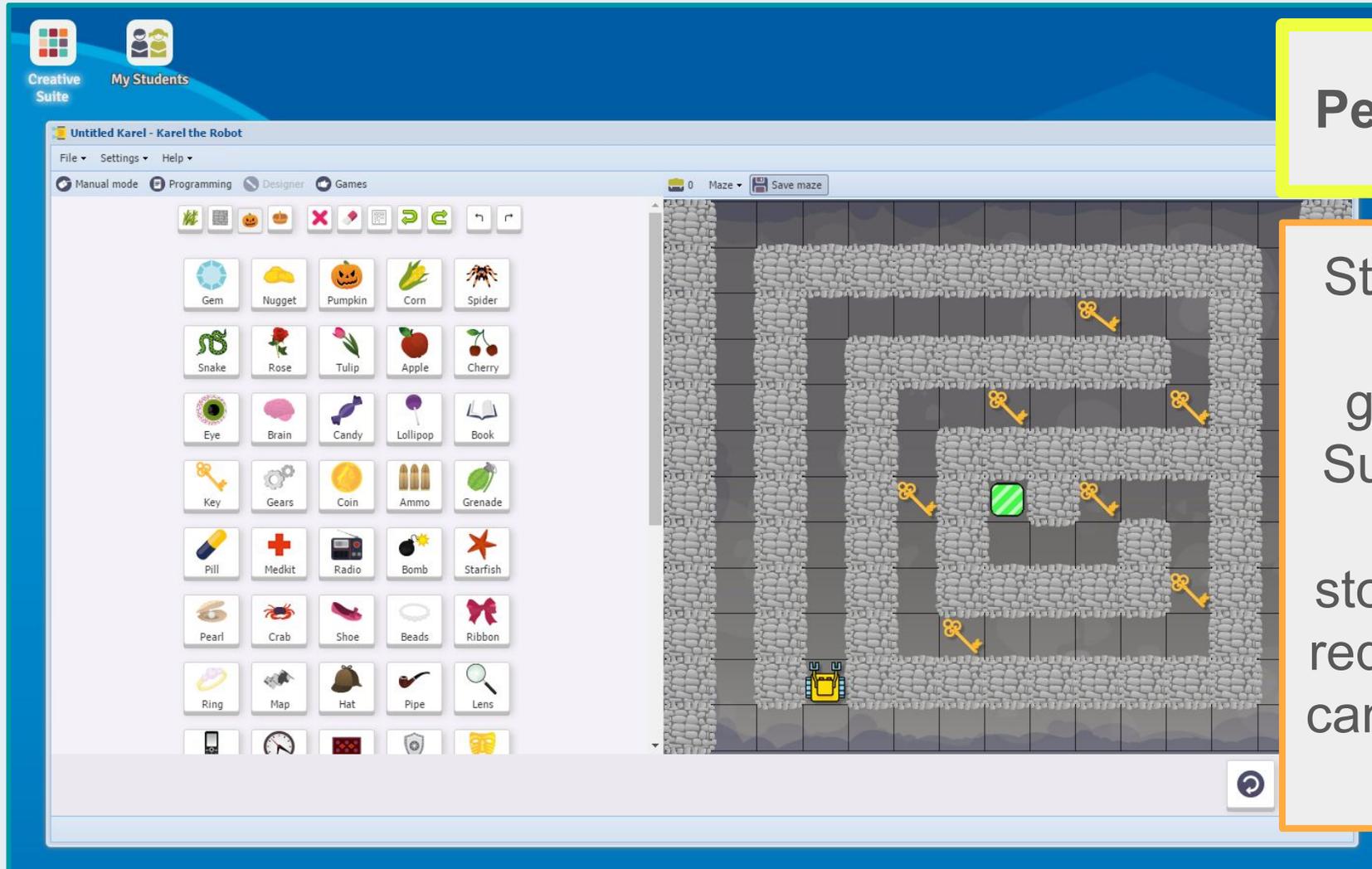


My Students shows how many points students have earned and what levels they are working on. You can get precise information by clicking on a completed level.

Students earn points for each level, with bonus points for writing the code in fewer lines, and for not using the code template (hints).



# How do I monitor and assess my students?



## Performance Tasks

Students can create their own Karel games in Creative Suite, complete with code solutions, storylines, and game requirements. These can be saved, shared and published.

# How do I monitor and assess my students?

## Student Journals

Each Section has 4 pages:

1. Recall questions
2. Notes/Application questions
3. Bulletin Board to post links, pictures, ideas
4. Design page for designing games

### SECTION 18: USING THE FUNCTION RANDINT()

In this section, you learn how to generate random integers using the function `randint()`, make Karel repeat something a random number of times, calculate the maximum and the minimum of a given set of numbers. You know that the function `randint(6)` can be used to simulate rolling dice.

`randint()` was used to simulate a game of chance in 18.1 and 18.2, and to build columns of random height in 18.3 and 18.4. In 18.5 to 18.7, you learned to write a function to determine maximum and minimum values of those columns.

Chance situations: How would you write a function for the following?

Conditions	Code
Rolling "snake eyes" 	
Rolling a 7 on a dodecahedral die 	

Explain the procedure for finding the maximum height of the columns in 18.6. What are the limitations? What minor change is needed to find the minimum?


### SECTION 18 NOTES

Use this space to write your own notes, questions, and problems.


### QUESTIONS

It's your worst nightmare: you start a test, and can't remember anything! You will have "go on" and hope for the best. This is a multiple choice test, with a, b, c, or d as answers. Write those on scraps of paper to be drawn at random for each answer.

Write which answer you drew in the spaces below. The answer key is at the end of this section. Check your answers. Did guessing (random drawing) pass the test? Compare your results with those of another student.

Question	Random Answer	Actual Answer	Correct? Y/N	Question	Random Answer	Actual Answer	Correct? Y/N
1.				6.			
2.				7.			
3.				8.			
4.				9.			
5.				10.			
Score (Correct/Total)				Did you pass?			

Is this a good application for randomness? Explain.

--

# How do I monitor and assess my students?



Home ▾ Teachers Librarians Parents Resources ▾ Blog Free Portal

## ASSESSMENT: ALGORITHMIC THINKING

Algorithmic thinking is a collection of skills that comprises critical thinking, identifying repeating patterns, and breaking complex problems into simpler ones that are easier to solve.

PRE-TEST

POST-TEST

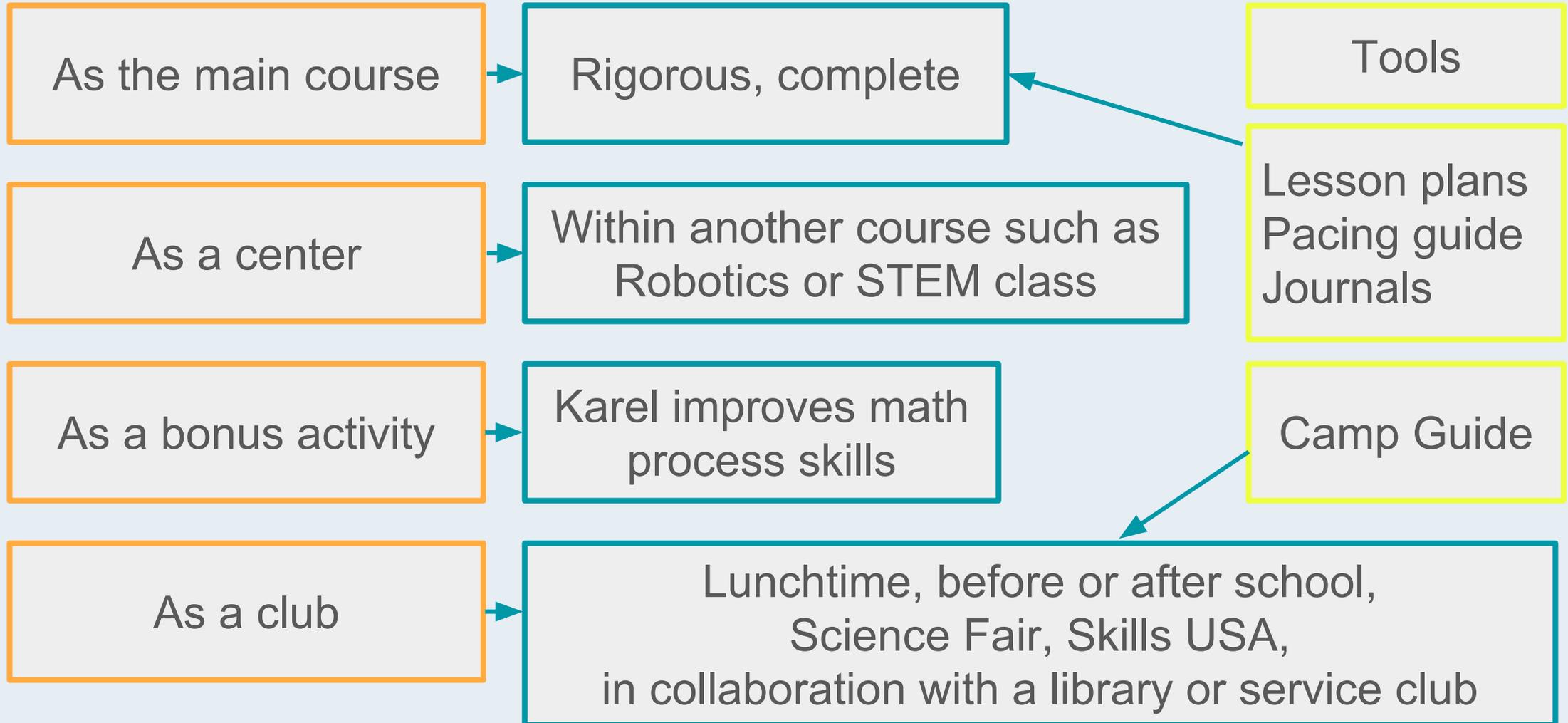
Have students take the pre-test on the NCLab website before starting Karel, then the post-test afterwards

Coming soon...

Built-in quizzes

End of unit assessments

# How does Karel fit in my curriculum?



# What else is there beside this course?

## Karel Hour of Code

<http://hoc.nclab.com>

for a quick demo

## More Courses

Turtle Coding  
Python Coding  
3D Modeling

## Creative Suite

Free programming,  
modeling, publishing,  
math tools and more

## Cloud Storage

Free cloud storage  
for files created in  
Creative Suite

## Online Gallery

Students can submit  
their games to be  
published

## Blog

Ideas, events, and  
contests on our blog  
(Newsletter coming soon)

# How Can We Help You?

**Email and phone support?**

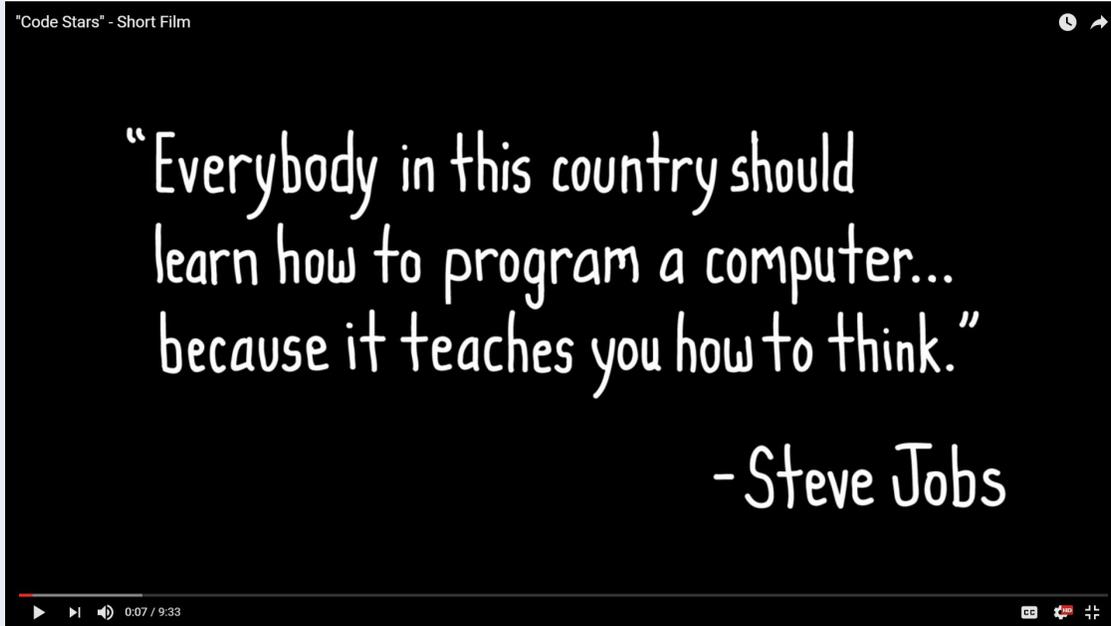
**Hangout Meeting?**

**Live training?**

**Webinar (your topic)?**

**Monthly newsletter?**

**What would you like to see?**



nclab

More Than Coding



We love to hear from you!

Email: [support@nclab.com](mailto:support@nclab.com)

Phone: (800) 666-2024 or (775) 303-6075