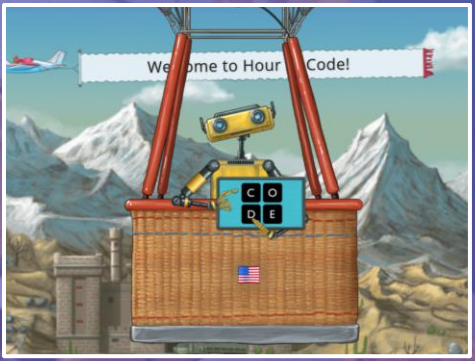
KAREL THE ROBOT: LESSON PLAN FOR HOUR OF CODE



OVERVIEW

A determined robot named Karel runs through jungles, swims in the sea and climbs icy mountains, collecting useful things like candies and spiders while trying not to crash into walls. Karel is a robot after all and needs a human to guide him, a human willing to write simple programs so Karel knows which way to go, what to pick up, and where to put it.

Karel the Robot teaches students how to write programs using the Karel programming language, which is based on Python. Starting with simple keystrokes in Game 1, students move on to typing lines of code to create and run programs. They learn to recognize patterns that can be written as repeat loops, test for conditions, and write defined commands for routine procedures.

The Karel the Robot activity consists of 15 mini-games. Most of the coding has already been done. Students fill in a few lines to complete each program, and then test the code by running Karel the Robot through a maze. Successful completion of each game unlocks the next one.

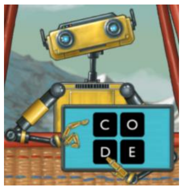
The games are designed for students ages 10 to 15 but can be enjoyed by anyone who wants to get a taste of programming.

OBJECTIVES

Students will learn to:

Control movement with `go`, `left`, and `right` commands. Pick up and place objects with `get` and `put` commands. Code a set of instructions with `go`, `left`, `right`, `get` and `put`.	Games 1 to 3
Create `repeat` loops based on repeated patterns. Repeat single and multiple commands.	Games 4 to 6
Test conditions using `if` statements.	Games 7 to 9
Create `while` loops to repeat commands until a continuing condition is false.	Games 10 to 12
Define a keyword (`def descriptive_name`) that calls a set of commands. Use the keyword in the main program.	Games 13 to 15

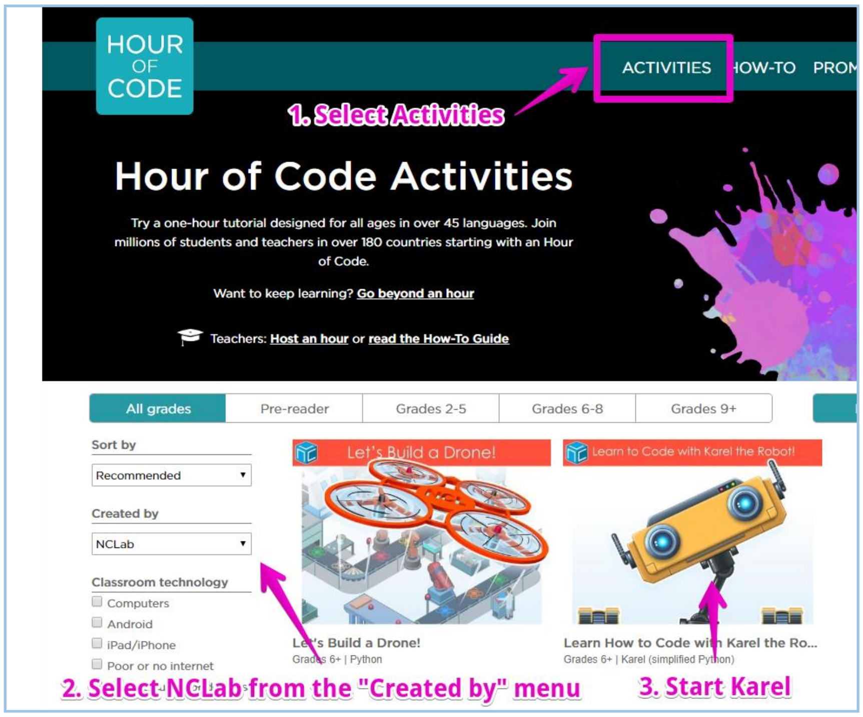
MATERIALS AND PREP



Personal computers or tablets, with Internet access: one per student or student pair.

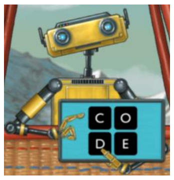
- Both PC and MAC platforms are supported.
- Use Chrome, Firefox, Safari, or Edge browsers. Internet Explorer is NOT supported and may cause problems with displaying graphics.
- Desktop computers, laptops and Chromebooks are preferred. Students find typing easier with a keyboard.
- iPad tablets are supported. Have students use their iPads in landscape mode for best results. Android tablets are not supported at this time.
- Phones are not supported.

Projector or Smartboard (optional) attached to a computer for demonstration or modeling.
Hour of Code Requirements: If you are doing this exercise as part of Hour of Code, please visit https://hourofcode.com for instructions on how to set up your class for a coding event. Students can easily find Karel on the Activities page by typing "NCLab" in the "Created By" menu.


HOUR OF CODE
ACTIVITIES
HOW-TO
PROM
1. Select Activities
Hour of Code Activities
Try a one-hour tutorial designed for all ages in over 45 languages. Join millions of students and teachers in over 180 countries starting with an Hour of Code.
Want to keep learning? Go beyond an hour
Teachers: Host an hour or read the How-To Guide
All grades
Pre-reader
Grades 2-5
Grades 6-8
Grades 9+
Sort by
Recommended
Created by
NCLab
Classroom technology
Computers
Android
iPad/iPhone
Poor or no internet
Let's Build a Drone!
Learn to Code with Karel the Robot!
Let's Build a Drone!
Grades 6+ | Python
Learn How to Code with Karel the Ro...
Grades 6+ | Karel (simplified Python)
2. Select NCLab from the "Created by" menu
3. Start Karel

YouTube videos: some schools block YouTube. The demonstration videos embedded in the games may need to be unblocked by an administrator to make them available to students.
Wrap up robotics video or demonstration: (optional) There are many cool videos that demonstrate robotics. Use search terms such as robots in surgery, agriculture, military, advanced manufacturing to find current examples to share during the wrap up. Demonstrate or have students work with programmable robots or toys. There are many inexpensive options on the market. Seeing robots in action will help your students connect Karel's behavior to real world robotic challenges.
Printables: (optional) Students can take notes as they make their way through the games and use the notes to complete an exit ticket. Printable materials are attached at the end of this lesson plan.
Certificates: Print certificates for students who successfully complete all 15 games.

TEACHING TIPS

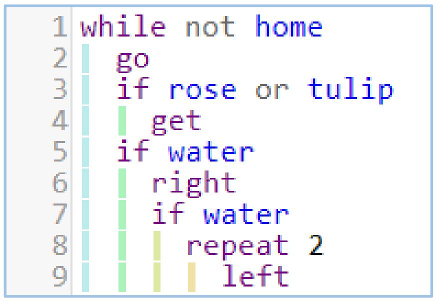


- Spend an hour or two ahead of time trying out the games yourself.
- Understand the basic moves:
    - `left` and `right` are always from Karel's point of view. These are 90 degree turns. Karel can turn around using two `left` or two `right` commands (180 degrees).
- Karel is equipped with sensors, much like a real robot. One is underneath the robot. The commands `get` and `put` use this sensor.
    - `get` picks up collectible objects in the same square as Karel.
    - `put` puts objects in a container in the same square as Karel.
- Karel is also equipped with a sensor that looks in front, much like we use our eyes. This sensor detects obstacles such as walls. For example, in the condition:

```
if wall
  left
```

  Karel will check the square in front for a wall, and if it is present, will turn left.

- Computers are the ultimate "Grammar Police". The Karel language simplifies grammar (syntax) but still has a few rules:
    - Indentation: the lines following a `repeat`, `if` or `while` statement must be indented two spaces (four spaces also works). Color-coded vertical lines make the indentations easier to see.
    - Spelling: must be perfect.
    - Capitalization: all words are lowercase.
    - Correctly spelled, lowercase keywords turn blue.
    - Punctuation: Karel does not use punctuation for the commands in these exercises.



```
while not home
  go
  if rose or tulip
    get
  if water
    right
    if water
      repeat 2
        left
```

- As much as possible, allow the students the freedom to figure out what to do on their own. Encourage students to watch what Karel does when they step through or run the programs.
- Decide ahead of time whether your students will be working alone, in pairs or as teams, or if certain students would benefit with being partnered with a mentor student. One effective pair method is "Navigator/Pilot". The Pilot is seated at the computer and controls the mouse and the keyboard. The Navigator stands next to the Pilot, reads all the instructions, and gives guidance. Students can switch roles for the next game or for the same game on the other student's computer, so that both students are awarded certificates.

LIST OF BASIC COMMANDS AND KEYWORDS

Command words: `go, left, right, get, put`

Directional commands `(go, left, right)` are always from the robot's point of view.

`go` advances the robot one step.

`left` turns the robot to its left.

`right` turns the robot to its right.

Retrieving and placing objects `(get, put)`

`get` picks up an object (Karel senses objects underneath, in the same square)

`put` places an object (Karel senses containers underneath it)

Loops

`repeat x`, where x = the number of times the body of the loop (indented commands) is repeated.

`while x`, where x = a defined continuing condition. The body of the loop will repeat until the continuing condition is false.

Conditions

`if x`, where x = a defined condition. If the condition is true, the indented commands will be executed. If it is false, the commands will be skipped.

Keywords that are Logical Operators

`not` the condition is that the sensor is `not` present.

Important Sensor Words

`north` Karel is facing `north`, or the top of the maze.

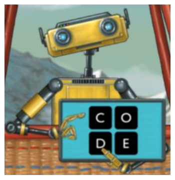
`wall` A `wall` is a type of obstacle. Karel senses obstacles one square ahead. Karel avoids obstacles by turning. Otherwise, Karel will crash.

`home` the `home` square. Karel sense it underneath. Arriving home is one way to end a program.

Defined commands

`def` `def` followed by a user-created keyword, initiates a defined command. The set of instructions in the indented body can be used elsewhere in the program simply by writing the keyword. The body may end with the word `return`, if the command returns values to the main program. Otherwise, `return` is optional.

VOCABULARY



Code is a set of instructions sent to the computer. For example, when you type the word `go`, the computer program knows to move Karel one square forward. Code is the way you translate instructions from "human" to "machine". In these maze exercises, Karel picks up objects, puts them in containers, and must make it to the home square without running into any obstacles. All these actions are controlled by code. Using the Step button, watch Karel's behavior as each line of code is executed.

A Computer Program is an algorithm or set of instructions written using a programming language. NCLab's Karel language is based on Python, a language used in math, engineering, science, backend programming, data analysis, and artificial intelligence.

Home is the destination square, marked by red diagonal stripes which change to green after Karel has met the goals of the program, such as collecting all objects. It is common to use `home` to end a program, and it used in conditions such as "`while not home`".

Max is the maximum number of steps, operations, or lines of code.

Steps are the number of squares that Karel moves. The shoe icon 10 counts the number of steps.

Operations are anything that Karel does: move, turn, pick up or put down objects. The computer icon 14 counts the number of operations.

Objects are items placed or collected in the maze. (The word "object" has other meanings in programming that are not used here).

Loop: A set of commands repeated a given number of times (`repeat`), or until a condition stops (`while`).

Body: the body contains the commands to be repeated. The commands are written on the lines following the `repeat`, `while`, `if`, or `def` statement. The body is indented two spaces.

Algorithm: a series of logical steps that leads to the solution of a task.

Logical error: a mistake in an algorithm. Planning helps reduce the number of errors.

Syntax: the way a command line is written.

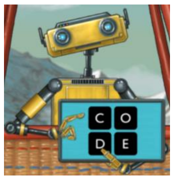
Syntax error: a mistake in spelling, operators, indentations, spaces.

Sensor words: keywords from the Karel library, which can include collectible items or objects (such as `orchid`), containers (such as `basket`), obstacles (such as `wall, plant`), or other conditions (`north, home`). A word that is both in the library and correctly spelled will be colored blue.

Condition: Conditions make decisions while the program is running, when we do not know exactly what will happen in advance. For example, Karel may need to collect all the coins it finds but may not know where the coins will be located. The `if` condition says: "If there is a coin, get it."

Comment: Comments are used to explain what the code is doing, written as # followed by the explanation. The hash symbol tells the program to ignore that line. You can also "comment out" a line of code when testing your programs, so that the program skips that line.

INTRODUCING THE LESSON (5 MINUTES)



Today, you will learn how to write code by playing games with Karel the Robot.

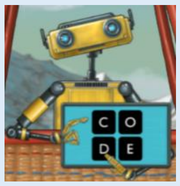
Code is a set of instructions sent to the computer. For example, when you type the word `go`, the computer program knows to move Karel one square forward. Code is the way you translate instructions from "human" to "machine".

Once you learn some programming techniques, you will be surprised to see how few lines of code are needed. In these maze games, Karel picks up objects, puts them in containers, and must make it to the home square without running into any obstacles. All these actions are controlled by code. Use the Step button to see Karel's behavior as each line of code is executed.

A Computer Program is an algorithm, or set of instructions, written using a programming language. NCLab's Karel language is based on Python, a language used in math, engineering, science, data analysis, and artificial intelligence.

There are fifteen games in all. With each game, you will be learning a new idea in coding. Most of the program is already written: you just have to fill in the blank lines with the missing code.

Note takers: If you have chosen to use the notetakers, hand them out at this time. Students can complete the notetakers as they go through the games or take notes after the whole session as a review.

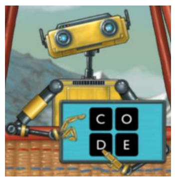


If you get stuck:

1. **Reread the instructions.**
2. **Run Karel through the maze mentally. Do you see any patterns?**
3. **Run the program line-by-line using the step icon.**
4. **Read the error message.**
5. **Check the list of words that are required in the program. Have you used them all?**
6. **Check for spacing, spelling and indentation.**
7. **Use the Help Menu at the top of the screen to view videos or read the reference manual.**
8. **Check with your partner.**

You must successfully complete each game to unlock the next one. Good luck, and let's get coding!

KAREL THE ROBOT ACTIVITY (40 MINUTES)



Students will play the games, progressing at their own pace. Students may be working as individuals, partners, or teams, as determined by the teacher.

Encourage and cheer problem-solving skills and progress; help as needed.
Let students know when they have about 5 minutes left.

WRAP UP (10 MINUTES)

Discuss how the games went. If you have robotics videos or "robots" available, show one now.

1. How do the real-world robot's actions compare to Karel?
2. Can you see Go, Left, Right, Get, and Put?
3. Do you see repeated patterns?
4. Do you see conditions (if-else, while)?
5. Do you see routines that could be written as functions?

Today, we participated in the Hour of Code, along with millions of other coders around the world. Computers and real robots need humans to write instructions for them, whether they are used for fun or for work. You can write code to create art, build a car, test medicines, or design a video game. Coding is used everywhere.

ASSESSMENT (5 MINUTES)

Give each student an Exit Ticket to write down a few sentences about Karel and coding: what was enjoyable, what was challenging, and what they could imagine doing with their coding skills.

DIFFERENTIATION

Support:

- Model Game 1 (demo) and Game 2 (requires coding), then have the students start on the games.
- If students have difficulty visualizing the movements and turns, try it "live" using a tiled floor, or a taped off grid on a carpet.
- Have students use the step button and talk through each step. For example:
    - "repeat 3" means Karel will repeat the following steps three times
    - "if key, get" means if there is a key underneath Karel, Karel will pick up the key.
- Use peer support, such as the Navigator/Pilot method.
- Remind students that they can restart the game and try again.
- ESL Students can run Karel in one of several languages.

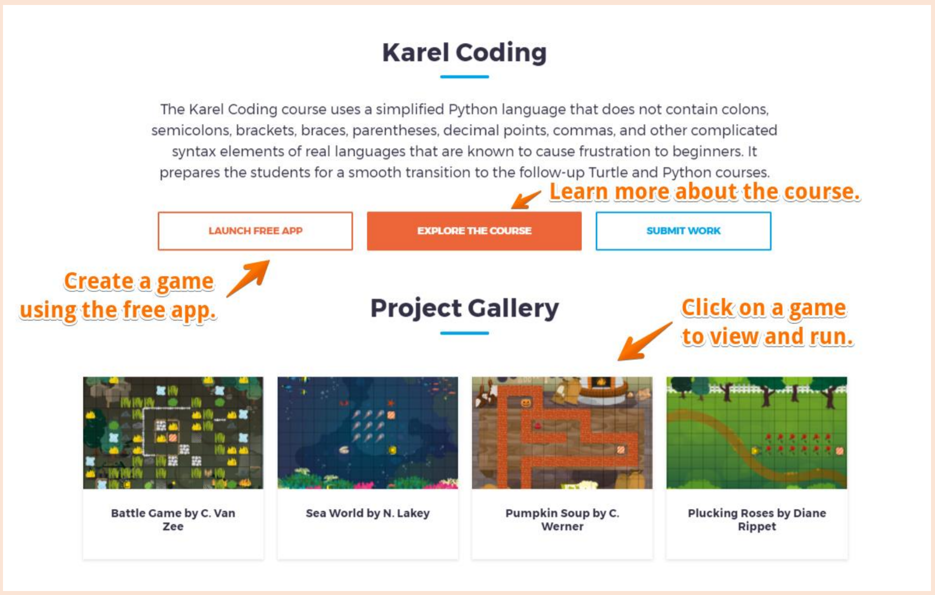
For students who finish the games early:

- Try one of the extended learning activities on the next page.
- Be a roving "Expert" to help other students.
- Explore other Hour of Code activities.

EXTENDED LEARNING

- Visit https://nclab.com/apps/ to view and run Karel games, 3D Models, and Python Turtle designs created by students.
- Create games using the Karel app.
- Learn about the full Karel course.
- Research programming careers on-line.
- Research robotics on-line.
- Explore other Hour of Code apps including NCLab's "Let's Build A Drone!"

For more Karel the Robot, visit https://nclab.com/apps/

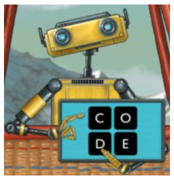


Karel Coding

The Karel Coding course uses a simplified Python language that does not contain colons, semicolons, brackets, braces, parentheses, decimal points, commas, and other complicated syntax elements of real languages that are known to cause frustration to beginners. It prepares the students for a smooth transition to the follow-up Turtle and Python courses.

Learn more about the course.
LAUNCH FREE APP
EXPLORE THE COURSE
SUBMIT WORK
Create a game using the free app.
Project Gallery
Click on a game to view and run.
Battle Game by C. Van Zee
Sea World by N. Lakey
Pumpkin Soup by C. Werner
Plucking Roses by Diane Rippet

STANDARDS



Common Core Standards for Mathematical Practices:

- SMP 1: Make sense of problems. Persevere in solving them.
- SMP 7: Look for and make use of structure.
- SMP 8: Look for and express regularity in repeated reasoning.

Common Core Standards for English Language Arts:

- L1, L2: Demonstrate command of the conventions of … grammar and usage, capitalization, punctuation, and spelling when writing. Coding conventions are "non-negotiable" and help shape these habits in students.
- SL.x.1: Students will engage in Speaking and Listening skills if they are using the Navigator/Pilot technique, peer assistance, or engaging in post-activity discussion.
- W.10: Students are writing responses as part of a routine of writing.

Next Generation Science Standards:

- SEP 2: Develop and use models. Students are learning algorithms that will be used to build their own programs.

Computer Science Standards (CSTA 2017):

- 1B-AP-10: Create programs that include sequences, events, loops, and conditionals. (P5.2)
- 1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. (P3.2)

Note: Hour of Code Karel has not been cross-walked with the organizations who oversee the use of standards. These standards are presented here for reference only.

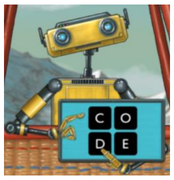
Citations:

Common Core: Authors: National Governors Association Center for Best Practices, Council of Chief State School Officers Title: Common Core State Standards (insert specific content area if you are using only one) Publisher: National Governors Association Center for Best Practices, Council of Chief State School Officers, Washington D.C. Copyright Date: 2010

Next Generation Science Standards: NGSS Lead States. 2013. Next Generation Science Standards: For States, By States. Washington, DC: The National Academies Press.

Computer Science Teachers Association (2017). CSTA K–12 Computer Science Standards, Revised 2017. Retrieved from http://www.csteachers.org/standards.

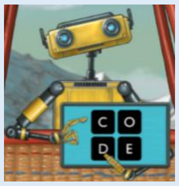
RESOURCES



The following pages included printable resources that you can share with your students.

1. Shareable slides with instructions
2. What is Karel Doing? (simplified explanation of commands)
3. Karel Notetaker
4. Karel Exit Ticket

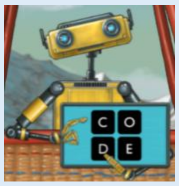
SLIDE 1



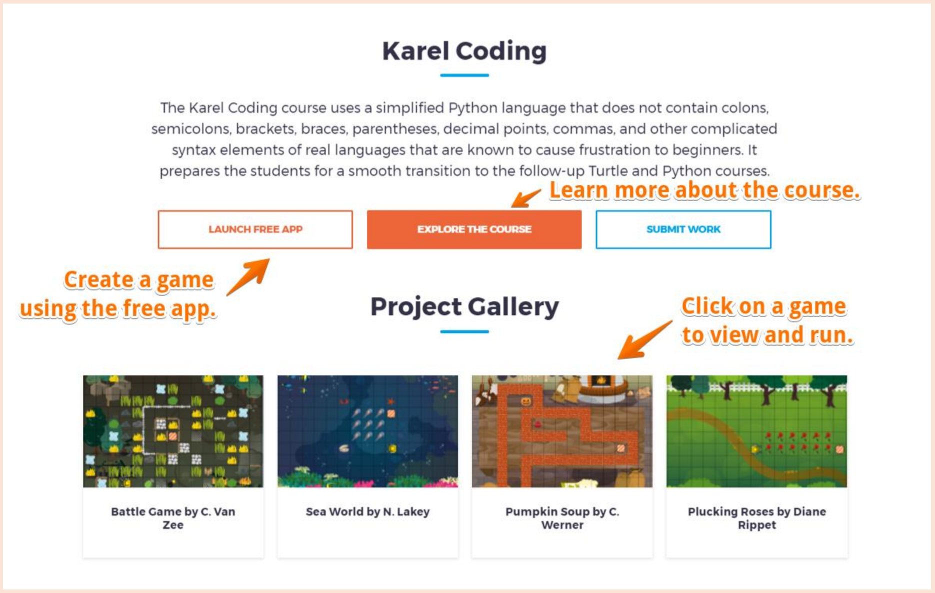
If you get stuck:

1. **Reread the instructions.**
2. **Run Karel through the maze mentally. Do you see any patterns?**
3. **Run the program line-by-line using the step icon.**
4. **Read the error message.**
5. **Check the list of words that are required in the program. Have you used them all?**
6. **Check for spacing, spelling and indentation.**
7. **Use the Help Menu at the top of the screen to view videos or read the reference manual.**
8. **Check with your partner.**

SLIDE 2



For more Karel the Robot, visit https://nclab.com/apps/



Karel Coding

The Karel Coding course uses a simplified Python language that does not contain colons, semicolons, brackets, braces, parentheses, decimal points, commas, and other complicated syntax elements of real languages that are known to cause frustration to beginners. It prepares the students for a smooth transition to the follow-up Turtle and Python courses.

Learn more about the course.

LAUNCH FREE APP | EXPLORE THE COURSE | SUBMIT WORK

Create a game using the free app.

Project Gallery

Click on a game to view and run.

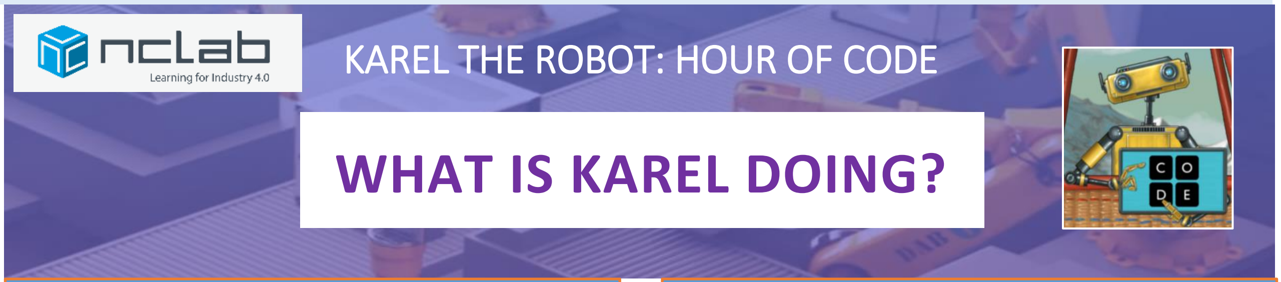
Battle Game by C. Van Zee

Sea World by N. Lakey

Pumpkin Soup by C. Werner

Plucking Roses by Diane Rippet

WHAT IS KAREL DOING?



GAMES 1 - 3

BASIC COMMANDS

`go`: Make one step forward.

`left`: Turn 90 degrees left.

`right`: Turn 90 degrees right.

`get`: Collect an object on Karel's square

`put`: Put an object in a container on Karel's square.

GAMES 4 – 6

REPEAT LOOPS

The `repeat` loop is used when the number of repetitions is known in advance. The body is indented two spaces.

```
repeat 5
  go
```

This loop tells Karel to move forward (go) five times.

Repeat loops can contain multiple commands, conditions, defined commands, and other loops.

GAMES 7 - 9

IF CONDITIONS

The `if` condition tells Karel to check for a condition. If the condition is true, Karel will execute the indented commands. If it is false, Karel will skip to the next part of the program. The body is indented two spaces.

```
if wall
  right
```

If Karel senses a wall, Karel will turn right.

GAMES 10 - 12

WHILE LOOPS

The `while` loop should be used when we do **not** know in advance how many repetitions will be needed. It ends when the condition is false. The body is indented two spaces.

```
while not wall
  go
```

This loop tells Karel to keep moving forward (go), as long as Karel does not sense a wall (not wall is true). If Karel senses a wall (not wall is false), then the loop ends.

While loops can contain multiple commands, conditions, defined commands, and other loops.

GAMES 13 -15

CUSTOM COMMANDS

Custom commands are used to define as set of instructions that are needed more than once in the program. The body is indented two spaces.

```
def turnback
  repeat 2
    left
```

Now you can use `turnback` in any part of the program just by writing the keyword `turnback`.

NOTETAKER

KAREL THE ROBOT: HOUR OF CODE

NAME: _________________________________

WRITE BRIEF NOTES ON EACH SET OF GAMES. WHAT DID YOU LEARN?

GAMES 1 (DEMO), 2, AND 3
GAMES 4 (DEMO), 5, AND 6
GAMES 7(DEMO), 8, AND 9
GAMES 10(DEMO), 11, AND 12
GAMES 13(DEMO), 14, AND 15

EXIT TICKET

KAREL THE ROBOT: HOUR OF CODE

NAME: ___________________________________

YOU CAN WRITE CODE TO CREATE ART, BUILD A CAR, TEST MEDICINES, OR DESIGN A VIDEO GAME. CODING IS USED EVERYWHERE!

WHAT PART OF KAREL THE ROBOT WAS THE MOST ENJOYABLE?

WHAT PART OF KAREL THE ROBOT WAS THE MOST CHALLENGING?

IF YOU WERE A CODER, WHAT WOULD YOU CODE? HERE ARE SOME IDEAS:

Games, Simulations, and Sports	Website Design
Robots and Drones	Engineering and Construction
Art and Music	Big Data and A.I. (Artificial Intelligence)

WHY DOES THAT AREA INTEREST YOU? WHAT DO YOU THINK YOU NEED TO LEARN NEXT?