nclab
More Than Coding

KAREL JR PROGRAMMING COURSE

PACING GUIDE

AUGUST 17, 2016

PREFACE

This pacing guide is just that – a guide. Karel can be used by itself, or with supporting activities which enrich the learning experience. Younger and less experienced students will take longer to master each level, while those with more experience using the course to solidify their foundation will finish the course quickly. Either way, Karel is a fun and visual way to enter the world of programming.

Karel uses a simplified version of Python, so syntax is easy. Students are scripting their code from Section 2 onward. Running the maze is an immediate visual check on how the code works. Our experience has shown that students stay focused on learning for the entire period. They aren't afraid to make mistakes, and that is a fundamental key to success when crafting programs in the real world.

NCLab has an array of materials to support your classroom. The lesson plans contain ideas to help open and close each section of learning, and provide you with notes on each level that can help guide students if they get stuck. There are also Student Journals for each of the five Units of Study. The journals provide students with a place to answer questions, take notes, consider how programming can solve problems in the real world, and design games. Journaling expands the reach of the program across the curriculum.

As a teacher or program administrator, you can decide how Karel fits within your own computer science curriculum. Will it be taught as an intensive course over a short period of time? Will it be woven into your curriculum throughout the year? If you aren't sure, try running the course for about a week with your students. How quickly does your particular class progress?

If only limited time is available, the course can be run without some or all of the additional performance tasks and journals. This might be a preferred option if Karel is being used as part of the full suite of NCLab courses in less than one school year, as a section in a broader curriculum, or as a quick introduction to programming before studying more advanced languages.

In this guide, we are assuming 50 to 60 minutes per day of class time. Estimates are given for a full activity sequence (first number), and only the self-paced course (second number).

Remember, as with all self-paced courses, some students will complete the course faster than others. Consider having the advanced students coach others, create more challenging projects, or move on to another self-paced course such as NCLab's Tina, 3D Modeling, or Python.

Standards

In the left column, you will see a reference to standards. Standards vary from state to state and are always being revised. This guide uses computer standards based on ISTE/CTE and CSTA models. Wherever possible, the levels are correlated to Common Core Math and English Language Arts standards, and to Next Generation Science Standards. References to these standards are included at the end of this document.

Have fun teaching this class! Please don't hesitate to contact NCLab at our website (nclab.com) if you have any questions or require access to materials.

KAREL JR 1-5				
Course length is based on 50 to 60 minutes per day of class time. Estimates for the units and whole course are given for a full activity sequence (first number), and only the self-paced course (second number). Standards Listed: Common Core Math Content, Common Core ELA, CSTA Computer Science, NGSS Science. Standards are listed in sections that are closely matched or where they first appear, but may be present in other sections. Common Core Math Process standards are exercised throughout the course.				
Standards	**Unit 1**	**Section 1 and 2:Basics Sections 3-5: Repeat loops**	**Concepts and Skills**	**5.3/2.4**
Introduction to Course: Log onto NCLab Desktop; view course overview video. Optional pre-test				1/0.5
OA.C Patterns and relationships	1.1-1.7	Manual controls	Control Karel with manual (keystroke or icon) commands: go, left, right, get, put Familiarize with game features. Learn about steps and operations.	0.5
W.x.1, 2,3 Narrative (story lines), informational (game instructions, journals) **W.x.6** Use technology to publish **W.x.10:** Write routinely **ETS1, 2, 3** Engineering design **CT.L2-10** Evaluate what kinds of problems can be solved using modeling and simulation. **CT.L2-11** Analyze the degree to which a computer model accurately represents the real world.		**Creative Suite** Performance Task/Journal	Learn how to create a game in Creative Suite. Create game in manual mode and complete journal based on Section 1.	0.5
L.x.1, 2 use precise syntax, grammar, spelling and punctuation	2.1-2.7	Scripted basic commands	Type commands go, left, right, get, put in simple programs.	0.5
		Performance Task/Journal	Game and journaling based on Section 2.	0.5
	3.1-3.7	Repeat loops	Repeat loops; purpose, syntax	0.3
		Performance Task	Game and journaling based on Section 3.	0.3
	4.1-4.7	Embedded repeat loops	Repeat loops that are included in a longer program. Syntax concerns.	0.3
		Performance Task	Game and journaling based on Section 4.	0.3
	5.1-5.7	Nested loops	Repeat loops within other repeat loops. Purpose and syntax	0.3
		Performance Task	Game and journaling based on Section 5.	0.3
			Review Unit 1. End of Unit Quiz.	0.5

Standards	Unit 2	Conditions	Concepts and Skills	4.9/1.9
OA. A, EE.A, 1,2,3 Expressions and equations, algebraic relationships	6.1-6.7	If conditions	Use if to detect sensor/condition; learn syntax for if conditions.	0.3
			Game and journaling based on Section 6.	0.5
	7.1-7.7	Else branch; north sensor	Else – alternate action when if condition is not fulfilled. Use north sensor to orient Karel in the maze prior to following a set of commands.	0.3
			Game and journaling based on Section 7.	0.5
	8.1-8.7	Sensors and logical operators	Use and, or, not sensors, empty pocket sensor to make decisions.	0.5
			Game and journaling based on Section 8.	0.5
	9.1-9.7	While loops	While not home, while wall, conditional loops	0.3
			Game and journaling based on Section 9.	0.5
	10.1-10.7	While loops in combination with other loops	More complex programs with embedded conditions.	0.5
			Game and journaling based on Section 10.	0.5
		Assessments	Review Unit 2; Quiz	0.5
Standards	**Unit 3**	**Defined commands, variables and functions**	**Concepts and Skills**	**6.1/3.1**
8.F.A.1 functions and variables.	11.1-11.7	Create defined commands	Define a command and call it from the main program.	0.5
			Game and journaling based on Section 11.	0.5
HS.F.BF.A.1 Determine, combine, compose functions. **CT.L2-12** Use abstraction to decompose a problem into sub problems.	12.1-12.7	Embed defined commands within programs. Understand specialized algorithms.	Complex problems with defined commands. Includes specialized, wall-following algorithms.	0.5
			Game and journaling based on Section 12.	0.5
CT.L3B-04 Evaluate algorithms by their efficiency, correctness, and clarity.	13.1-13.7	Programming skills	Shorter vs. effective programs; breaking complex tasks into simpler ones.	0.7
			Game and journaling based on Section 13.	0.5
SEP 5 Use mathematics and computational thinking.	14.1-14.7	Counting variables and functions, print command	Create counting variable; use inc() and dec() functions; print steps and/or results.	0.7
			Game and journaling based on Section 14.	0.5

CT.L3A-01 Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts.	15.1-15.7	Local and global variables; return function	Understand and use local and global variables and functions, within defined command and main program.	0.7
			Game and journaling based on Section 15.	0.5
		Quiz	Review Unit 3; Quiz	0.5
Standards	**Unit 4**	**Coordinates, Boolean Values, Random Integers, Python Lists**	**Concepts and Skills**	**6.5/3.5**
	16.1-16.7	Location sensors gpsx, gpsy	Use gpsx and gpsy to locate parts of the maze, and to control Karel's actions.	0.7
			Game and journaling based on Section 16.	0.5
	17.1-17.7	Boolean values, variables and sensors	Use True/False Boolean values within programs; understand sensors as Boolean values.	0.6
			Game and journaling based on Section 17.	0.5
HSS.MD.A.1,7, B6 Define and use random variables; display output.	18.1-18.7	Random integers; Max/min	Use randint() to determine outcomes, control actions, create patterns. Determine maximum and minimum values of a set.	0.6
			Game and journaling based on Section 18.	0.5
CT.L3B-06 Compare and contrast simple data structures and their uses (e.g., arrays and lists).	19.1-19.7	Lists, part 1	Create empty and non-empty lists; append to a list, parse a list, give the length of a list.	0.8
			Game and journaling based on Section 19.	0.5
	20.1-20.7	Lists, part 2	Reduce lists/ remove items from a list; merge lists, use for loops.	0.8
			Game and journaling based on Section 20.	0.5
			Review Unit 4; Quiz	0.5
Standards	**Unit 5**	**Probability, Recursion, Advanced Problems, Review**	**Concepts and Skills**	**13.1/ 7.6**
	21.1-21.7	Probability, using rand function	Determine 50/50 or 25/75 outcomes; use rand function to solve problems.	0.8
			Game and journaling based on Section 21.	0.5
HS.F.BF.A.1,2,3 Building and interpreting recursive functions.	22.1-22.7	Recursion, part 1	Use recursion to control a repeated task; use stopping conditions, understand infinite loops.	0.6
			Game and journaling based on Section 22.	0.5

	23.1-23.7	Recursion, part 2	Use recursion in more complex tasks, incorporating coordinates, inequalities, lists. Use a second recursion within a stopping condition.	0.7
			Game and journaling based on Section 23.	0.5
CPP.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements, logic, expressions, variables, and functions.	24.1-24.7	Review	Solve problems by using skills learned in the course.	1.5
			Game and journaling based on Section 24.	0.5
SEP 2 Develop and use models	25.1-25.7	Challenges	Solve challenging problems, including classic logic problems.	4.0
			Game and journaling based on Section 25.	0.5
			Review Unit 5; Quiz	1.0
		End of course Assessment	Review Final test on Karel Jr	2
Total instructional days for Karel 1-5 (with journal, performance task activities, review and assessments)				**36**
Total instructional days for Karel 1-5 (without extra activities or assessments)				**19**

References: Sources used for Standards

Computer Science Teachers Association (2011): CSTA K–12 Computer Science Standards. https://csta.acm.org/Curriculum/sub/K12Standards.html last viewed 8/17/2016

Comment: These standards are currently being revised in collaboration with several other organizations, including state governments, ACM, CSTA, Code.org, CIC, and NMSI. Visit https://k12cs.org/ "A Framework for K-12 Computer Science Education" for more information. Last viewed 8/17/2016

Common Core State Standards Initiative (2010): K-12 Math and English Language Arts standards. http://www.corestandards.org/ last viewed 8/17/2016

Comment: Most states have adopted these standards in some form. The State standard labels vary from state to state. Math Standards are divided into Concept standards and Practice standards. Although the Practice standards have not been listed in the chart, they are at work throughout the course.

Next Generation Science Standards (2014): K-12 Science Standards. http://www.nextgenscience.org/ last viewed 8/17/2016

Comment: NGSS has a unique structure, which includes SEP (Science and Engineering Practices); CCC (Cross-cutting Concepts), and DCI (Disciplinary Core Ideas). Again, most states have adopted some variation of the NGSS.