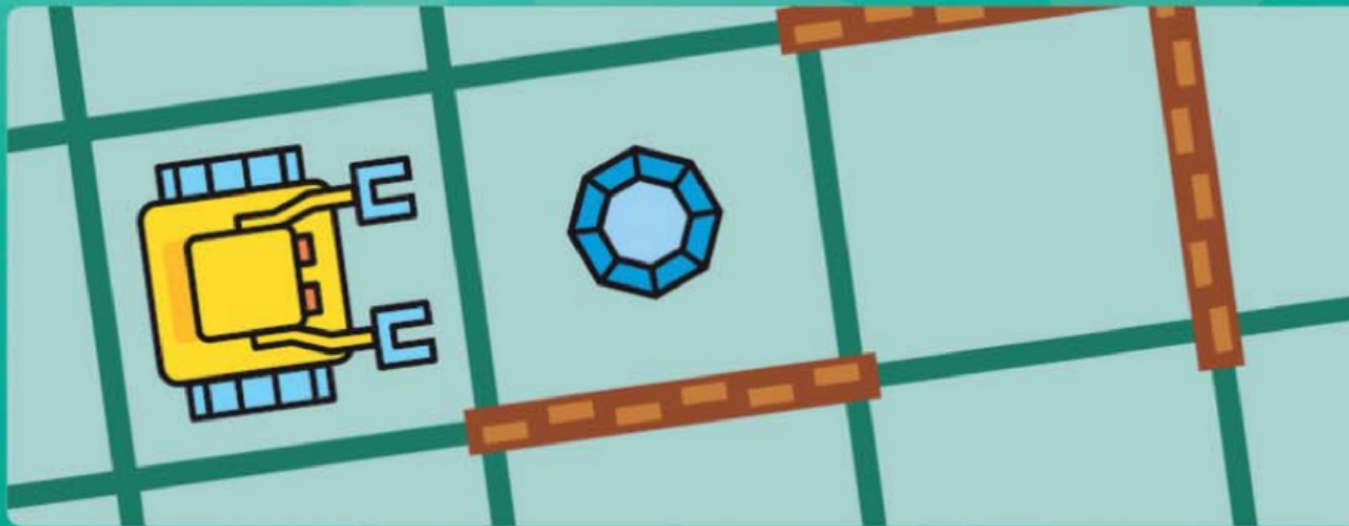




nclab

Learning for Industry 4.0

Learn how
to **Think** with
Karel
the Robot



Learn How to Think with Karel the Robot

Dr. Pavel Solin

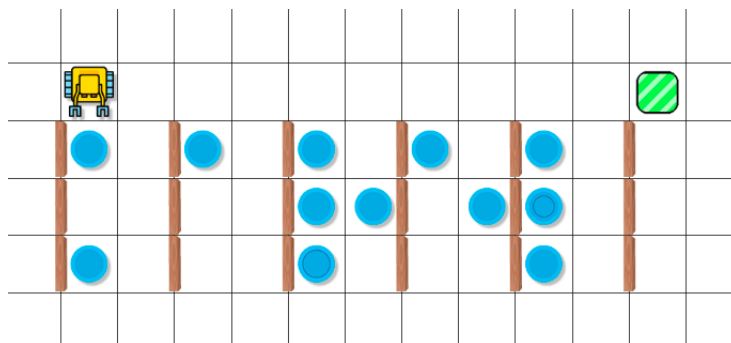
September 24, 2018

Preface

Computer programming is fun. Telling a machine what to do, and then watching it actually do it, is amazing. Programming is all about breaking complex problems into simpler ones which are easier to solve. Interacting with the computer will teach you how to be accurate, use logic, solve problems, persevere, overcome failure, and get things done. These are tremendously important life skills which will help you succeed in anything you will do in the future.

Now, you might ask: *"Why should I lose time with Karel the Robot - an educational programming language - when I can start right away learning a real programming language such as Python, C++ or Java?"* The answer is that computer programming is like driving a car: One thing is to learn how to operate the vehicle - start it, shift gears, push the gas pedal and brakes, etc. But another, even more important thing is to learn the traffic rules really well: What should you do when you are coming to an intersection? Who has the right of way? What are the meanings of the traffic lights and symbols? In short - how to become a good driver. In the context of computer programming, Karel will teach you how to be such a good driver. And after that, it will be incredibly easy for you to learn how to operate various other "cars" which are other languages such as Python, Java or C++.

By the way, Karel is not a toy language at all. It can solve very hard problems including classical world-class programming challenges such as the Eight Queens puzzle which you can find on Wikipedia. Karel also can parse binary trees using recursion, implement sorting algorithms, perform statistical experiments, or read Braille text:



Karel reading his own name in Braille.

PREFACE

About Karel the Robot

The educational programming language Karel the Robot was created at the Stanford University by Dr. R.E. Pattis who also wrote the original textbook *Karel the Robot: A Gentle Introduction to the Art of Programming* in the 1980s. At that time, its syntax was influenced by Pascal, a major programming language of that era. We have updated the language to be compatible with Python, while preserving Dr. R.E. Pattis' original ideas. Python is a major programming language of modern engineering and science.

About the Author

Dr. Pavel Solin is Professor of Computational Science at the University of Nevada, Reno. He has used the first 8-bit computers to draw ornaments on the screen of small black and white TV when he was 9. There were no computer monitors yet, and games were stored on audio cassettes. Much changed since then but Dr. Solin's passion for computers and programming remained the same. Today he is using the most powerful supercomputers to understand what happens inside of collapsing stars and other natural processes that cannot be observed or measured. He is fluent in several computer languages, wrote hundreds of thousands of lines of code, leads open source software projects, and enjoys learning new things every day.

Acknowledgment

We would like to thank educators and students for reporting bugs, suggesting new features, and providing valuable feedback. This is helping us to continuously improve the self-paced interactive Karel course in NCLab, as well as this textbook and the NCLab computing platform itself.

Contents

Preface	i
About Karel the Robot	ii
About the Author	ii
Acknowledgment	ii
1. Introduction	1
1.1. Karel Čapek and his 1921 play R.U.R.	1
1.2. Brief history of the Karel language	2
1.3. Various past implementations	2
1.4. New Karel language based on Python	3
1.5. Command <code>right</code> and some other new features	4
1.6. What can Karel do?	4
1.7. How does Karel differ from Python?	5
1.8. Review questions	5
2. Basic Commands	8
2.1. Manual mode	8
2.2. Steps and operations	10
2.3. Using keyboard controls	10
2.4. Seeing the world through the robot's eyes	10
2.5. Programming mode	11
2.6. Algorithm vs. program	12
2.7. Usually, a task has more than one solution	13
2.8. How to recognize the best solution	14
2.9. Logical errors	15
2.10. Syntax rules and syntax errors	16
2.11. Debugging and where did the word "bug" come from	18
2.12. Karel's bag	19
2.13. FILO (stack) and FIFO (queue)	20
2.14. Karel is case-sensitive	21

CONTENTS

2.15. Review questions	21
3. Counting Loop	25
3.1. Counting loop and the keyword <code>repeat</code>	25
3.2. Commenting your code	27
3.3. Repeating a sequence of commands	28
3.4. Indentation matters	29
3.5. How to figure out repeating patterns quickly and correctly	30
3.6. Stashing coins	32
3.7. Cooking potatoes	35
3.8. When it is not be possible to start a loop right away	36
3.9. Additional commands might be needed after a loop finishes	38
3.10. Nested loops	39
3.11. Revisiting previous programs	42
3.12. Crab cake	43
3.13. Starfish square	44
3.14. Triple-nested loops	47
3.15. Pearl necklace	48
3.16. Even more levels of nesting	49
3.17. Review questions	50
4. Conditions	53
4.1. What are conditions?	53
4.2. Why does Karel need them?	53
4.3. Karel's wall sensor and the <code>if</code> statement	54
4.4. What happens if there is no wall?	55
4.5. Other types of obstacles	56
4.6. Combining loops and conditions	57
4.7. Collectible objects	58
4.8. Collecting bananas	59
4.9. Containers	60
4.10. Collecting coconuts	61
4.11. The <code>else</code> branch	61
4.12. Hurdle race	63
4.13. Function <code>print</code>	63
4.14. Sensor <code>north</code>	64
4.15. Command <code>pass</code>	65

4.16.	Finding North	65
4.17.	Keyword <code>not</code>	66
4.18.	Sensor <code>empty</code>	67
4.19.	Keyword <code>and</code>	69
4.20.	Bounty	69
4.21.	Keyword <code>or</code>	70
4.22.	Danger	72
4.23.	Sensor <code>home</code>	73
4.24.	The full <code>if-elif-else</code> statement	73
4.25.	More is coming!	75
4.26.	Review questions	75
5.	Conditional Loop	79
5.1.	Conditional loop and the keyword <code>while</code>	79
5.2.	The difference between the conditional and counting loops	79
5.3.	Step by step	80
5.4.	Climbing a pyramid	81
5.5.	Narrow escape	84
5.6.	Combining the <code>repeat</code> and <code>while</code> loops	85
5.7.	First Maze Algorithm (FMA)	87
5.8.	Three possible failures of the FMA	89
5.9.	Right-handed version of the FMA	91
5.10.	Other types of mazes the FMA can handle	92
5.11.	Improving Program 4.9 "Finding North"	92
5.12.	Improving Program 4.12 "Emptying bag"	93
5.13.	Improving Program 4.17 "Danger"	93
5.14.	Improving Program 4.18 "Tunnel"	94
5.15.	Other types of loops: <code>do-while</code> , <code>until</code> , and <code>for</code>	94
5.16.	More is coming!	94
5.17.	Review questions	95
6.	Custom Commands	97
6.1.	Why are custom commands useful?	97
6.2.	Defining a custom command <code>star</code>	101
6.3.	Assembling the solution of the main task	102
6.4.	Collecting water bottles	106
6.5.	The optional <code>return</code> statement	109

CONTENTS

6.6.	Arcade game	110
6.7.	Second Maze Algorithm (SMA)	113
6.8.	Right-handed version of the SMA	117
6.9.	Rock climbing	118
6.10.	Program length vs. efficiency	119
6.11.	Writing monolithic code vs. using custom commands	121
6.12.	Defining commands inside other commands	123
6.13.	Create your own Karel language!	124
6.14.	Review questions	126
7.	Variables	128
7.1.	What are variables and why are they useful?	128
7.2.	Types of variables	129
7.3.	Choosing the names of variables	129
7.4.	Creating and initializing numerical variables	130
7.5.	Assignment operator =	130
7.6.	Displaying the values of variables	130
7.7.	Basic math operations with variables	131
7.8.	Keywords <code>inc</code> and <code>dec</code>	132
7.9.	Arithmetic operators <code>+=</code> , <code>-=</code> , <code>*=</code> and <code>/=</code>	132
7.10.	Comparison operators <code>==</code> , <code>!=</code> , <code><</code> , <code><=</code> , <code>></code> and <code>>=</code>	133
7.11.	Counting maps	135
7.12.	Water supply	136
7.13.	Karel and the Fibonacci sequence	137
7.14.	Review questions	138
8.	Functions	141
8.1.	Defining and using functions	141
8.2.	Builder	142
8.3.	Storekeeper	144
8.4.	Local variables and local scope	146
8.5.	Global variables and global scope	147
8.6.	Functions that accept arguments	149
8.7.	Can a function change the values of its arguments?	150
8.8.	Review questions	152
9.	Text Strings	153

9.1.	Raw text strings and text string variables	153
9.2.	Assigning, displaying, and comparing text strings	154
9.3.	Concatenating text strings	155
9.4.	Multiplying text strings with integers	155
9.5.	Length of a text string	156
9.6.	Parsing a text string with the <code>for</code> loop	156
9.7.	Reversing text strings	156
9.8.	Extracting individual characters by their indices	157
9.9.	Using negative indices	158
9.10.	Slicing text strings	159
9.11.	Displaying quotes in text strings	160
9.12.	Checking for substrings	161
9.13.	Counting occurrences of substrings	162
9.14.	Finding the positions of substrings	163
9.15.	Searching and replacing in text strings	164
9.16.	Removing substrings	165
9.17.	Swapping substrings	166
9.18.	Executing text strings as code	167
9.19.	Your homework	169
9.20.	Review questions	169
10.	Testing Your Programs	172
10.1.	Morse code project - Part I	172
10.2.	Testing the function <code>row</code>	174
10.3.	Morse code project - Part II	176
10.4.	Testing the function <code>english(symbol)</code>	178
10.5.	Morse code project - Part III	180
10.6.	Your homework	181
10.7.	Review questions	181
11.	Boolean Values, Variables, Expressions, and Functions	182
11.1.	George Boole	182
11.2.	Quick introduction to logic	183
11.3.	What is <i>Boolean algebra</i> ?	184
11.4.	Boolean values and variables in Karel	185
11.5.	GPS sensors <code>gpsx</code> and <code>gpsy</code>	187
11.6.	Comparison operators <code>==</code> , <code>!=</code> , <code><</code> , <code>></code> , <code><=</code> and <code>>=</code>	190

CONTENTS

11.7.	Boolean functions	190
11.8.	Karel's sensors are Boolean functions	192
11.9.	The <code>if</code> statement revisited	193
11.10.	Using the <code>if</code> statement to display debugging information	194
11.11.	Another look at the <code>while</code> loop	195
11.12.	Infinite loop <code>while True</code>	196
11.13.	Review questions	198
12.	Randomness and Probability	200
12.1.	Why is randomness useful in computing	200
12.2.	Generating random integers	201
12.3.	Karel is rolling a die	202
12.4.	Placing objects at random locations	203
12.5.	Building a random skyline	205
12.6.	Calculating the maximum	206
12.7.	Measuring the height of a skyline	207
12.8.	Measuring the height of a building	209
12.9.	Calculating the minimum	211
12.10.	Measuring the clearance of a cave	212
12.11.	Generating random Booleans	213
12.12.	Karel is tossing a coin	214
12.13.	Calculating probabilities	214
12.14.	Probability of events vs. their frequency	216
12.15.	Karel and random walks	218
12.16.	Using randomness to solve difficult tasks	219
12.17.	Review questions	220
13.	Lists	223
13.1.	What are lists and why they are useful	223
13.2.	Creating empty and nonempty lists	224
13.3.	Appending items to a list	225
13.4.	Measuring the length of a list	226
13.5.	Accessing list items via their indices	226
13.6.	Creating a list of lists	227
13.7.	Parsing lists with the <code>for</code> loop	228
13.8.	Checking if an item is in a list	229
13.9.	Removing and returning ("popping") items from a list	231

13.10.	Adding lists	232
13.11.	Multiplying lists with integers	232
13.12.	Deleting items from a list	233
13.13.	Gardener	234
13.14.	Expedition Antarctica	235
13.15.	Copycat	238
13.16.	Review questions	241
14.	Recursion	243
14.1.	What is recursion and why is it useful	243
14.2.	Which tasks are suitable for recursion?	245
14.3.	Collecting shields	247
14.4.	Under the hood	248
14.5.	Forgetting the stopping condition	249
14.6.	Moving shields to boxes	251
14.7.	Museum heist	251
14.8.	Mutually recursive commands	252
14.9.	Using variables and functions in recursion	253
14.10.	Parsing lists using recursion	255
14.11.	Recursion at its best	256
14.12.	Review questions	257
15.	Advanced Applications	258
15.1.	Flood (recursion)	258
15.2.	Contraband (recursion)	260
15.3.	Traversing binary trees (recursion)	262
15.4.	Bubble sort	263
15.5.	Reading Braille	266
15.6.	Cardan grille	270
15.7.	Land surveyor	274
15.8.	The Eight Queens puzzle	276
A.	Karel App in NCLab	281
A.1.	Launching the Karel app	281
A.2.	Building mazes	282
A.3.	Create your own game!	284
A.4.	Check how difficult your game is	287

CONTENTS

A.5.	Convert worksheet into a game	288
A.6.	Define game goals	290
A.7.	Final test	291
A.8.	Publish your game in the Internet!	292
B.	Self-Paced Karel Course in NCLab	294
B.1.	Age groups and prerequisites	294
B.2.	Brief overview	294
B.3.	What does it take to be an instructor	295
B.4.	Role of the instructor	295
B.5.	Instructor training	295
B.6.	Course structure	295
B.7.	Syllabus, lesson plans, pacing guide	295
B.8.	Student journals, cheat sheets, solution manuals	296
B.9.	Creative projects	296
B.10.	Explore NCLab	297
B.11.	Contact us	297
Index	298

Full text is available upon enrollment.