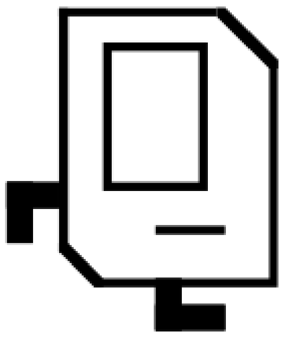
Chapter 1: Introducing Karel the Robot

compedu.stanford.edu/karel-reader/docs/python/en/chapter1.html

In the 1970s, a Stanford graduate student named Rich Pattis decided that it would be easier to teach the fundamentals of programming if students could somehow learn the basic ideas in a simple environment free from the complexities that characterize most programming languages. Rich designed an introductory programming environment in which students teach a robot to solve simple problems. That robot was named Karel, after the Czech playwright Karel Čapek, whose 1923 play R.U.R. (Rossum’s Universal Robots) gave the word robot to the English language.

Karel the Robot was quite a success. Karel has been used in introductory computer science courses all across the world and has been taught to millions of students. Many generations of Stanford students learned how programming works with Karel, and it is still the gentle introduction to coding used at Stanford.



What is Karel?

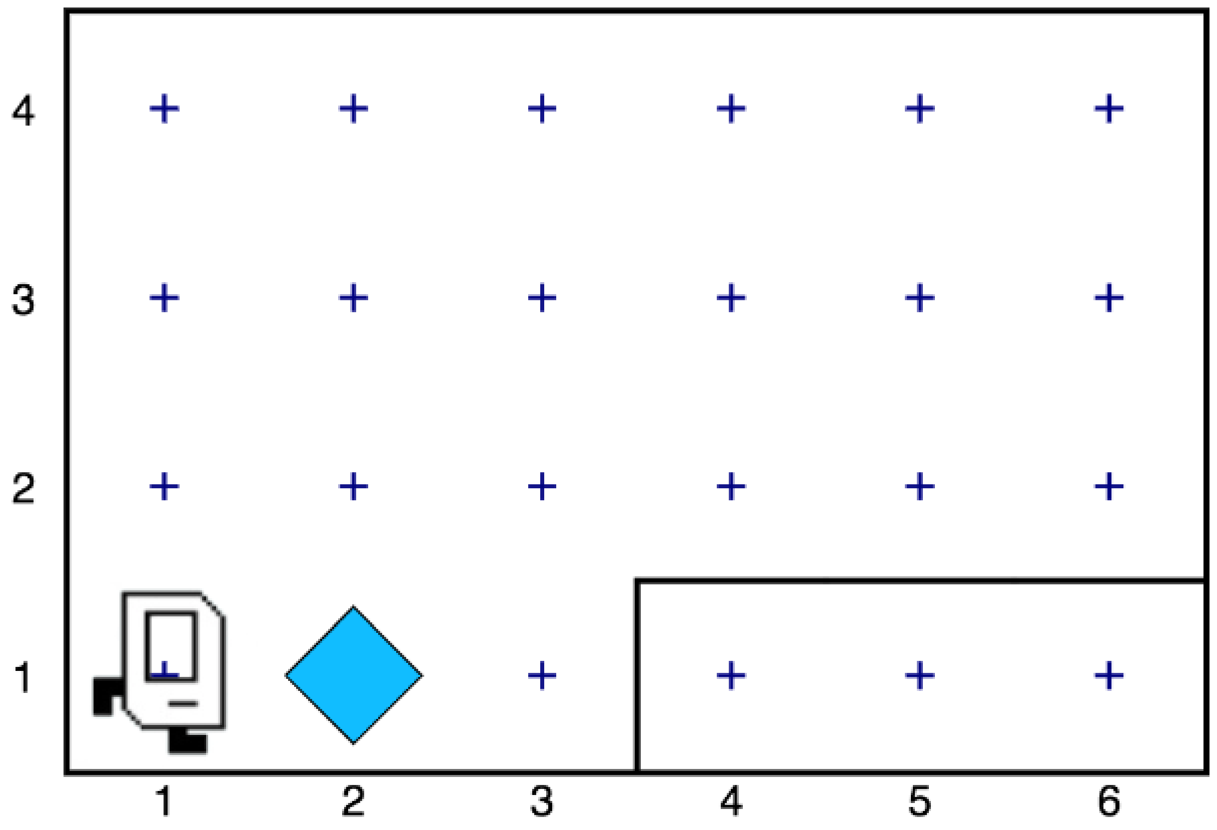
Karel is a very simple robot living in a very simple world. By giving Karel a set of commands, you can direct it to perform certain tasks within its world. The process of specifying those commands is called **programming**. Initially, Karel understands only a very small number of predefined commands, but an important part of the programming process is teaching Karel new commands that extend its capabilities.

Karel programs have much the same structure and involve the same fundamental elements as Python, a major programming language. The critical difference is that Karel’s programming language is extremely small and as such the details are easy to master. Even so, you will discover that solving a problem can be challenging.

By starting with Karel, you can concentrate on solving problems from the very beginning. Problem solving is the essence of programming. And because Karel encourages imagination and creativity, you can have quite a lot of fun along the way.

Karel’s world

Karel’s world is defined by rows running horizontally (east-west) and columns running vertically (north-south). The intersection of a row and a column is called a corner. Karel can only be positioned on corners and must be facing one of the four standard compass directions (north, south, east, west). A sample Karel world is shown below. Here Karel is located at the corner of 1st row and 1st column, facing east.

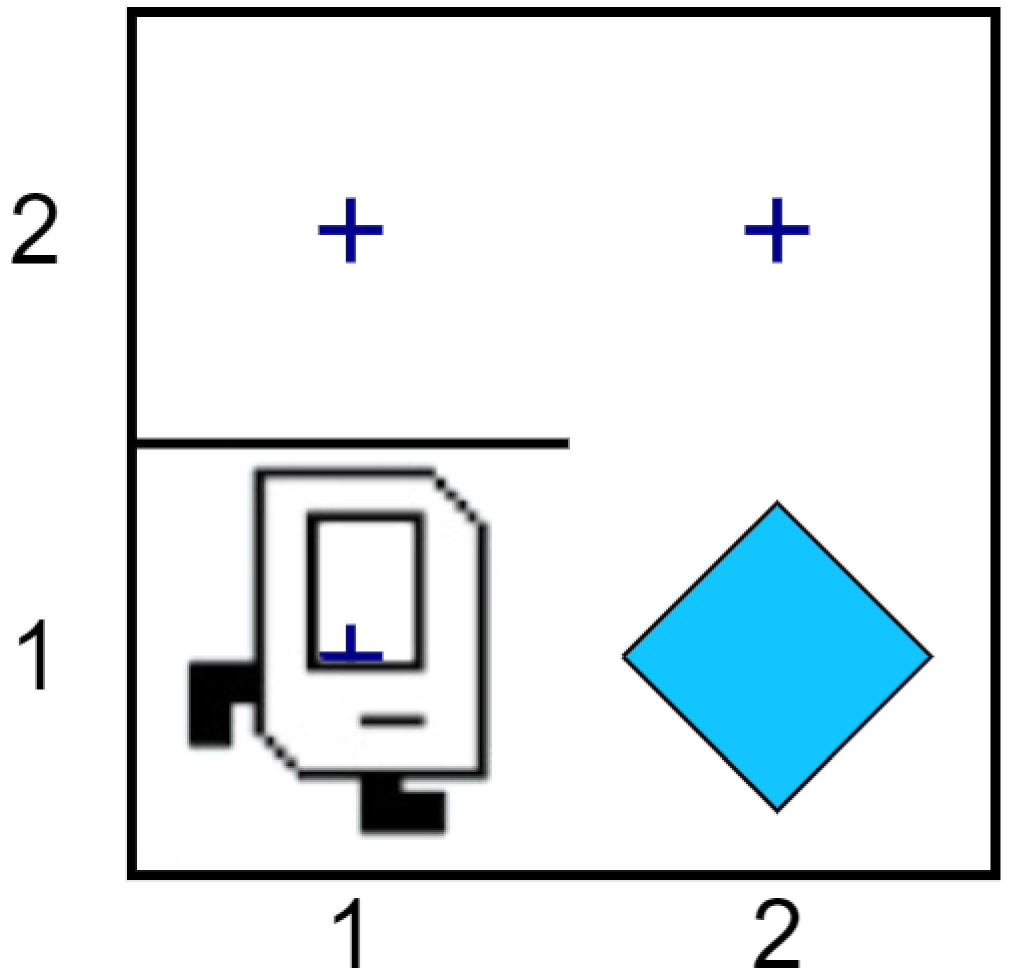


Several other components of Karel's world can be seen in this example. The object in front of Karel is a beeper. As described in Rich Pattis's book, beepers are "plastic cones which emit a quiet beeping noise." Karel can only detect a beeper if it is on the same corner. The solid lines in the diagram are walls. Walls serve as barriers within Karel's world. Karel cannot walk through walls and must instead go around them. Karel's world is always bounded by walls along the edges, but the world may have different dimensions depending on the specific problem Karel needs to solve.

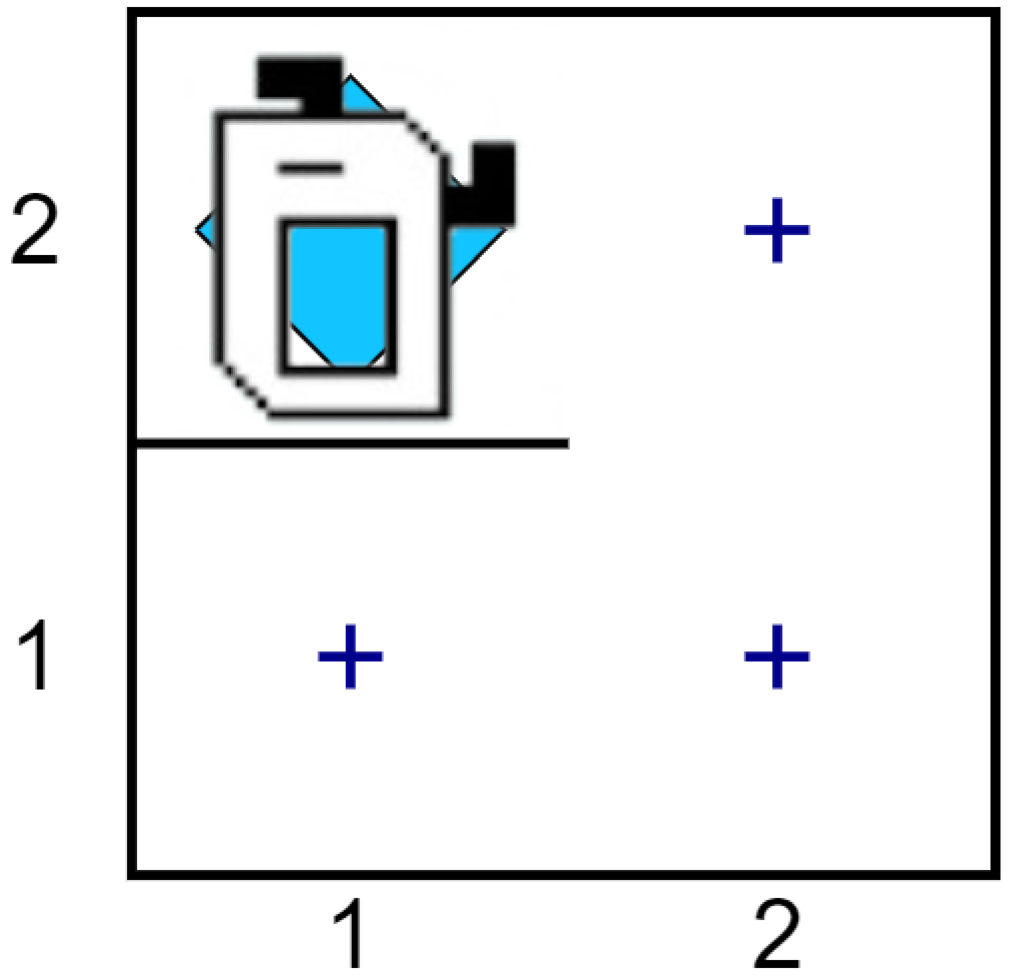
Karel's commands

When Karel is shipped from the factory, it responds to a very small set of commands. Lets try out the commands. Use the buttons bellow to get the "world" to match the "goal":

World:



Goal:



Here is what each command does:

Command	Description
`move()`	Asks Karel to move forward one block. Karel cannot respond to a `move()` command if there is a wall blocking its way.
`turn_left()`	Asks Karel to rotate 90 degrees to the left (counterclockwise).
`pick_beeper()`	Asks Karel to pick up one beeper from a corner and stores the beeper in its beeper bag, which can hold an infinite number of beepers. Karel cannot respond to a `pick_beeper()` command unless there is a beeper on the current corner.
`put_beeper()`	Asks Karel to take a beeper from its beeper bag and put it down on the current corner. Karel cannot respond to a `put_beeper()` command unless there are beepers in its beeper bag.

The empty pair of parentheses that appears in each of these commands is part of the common syntax shared by Karel and Python and is used to specify the invocation of the command. Eventually, the programs you write will include additional information in the space between the parentheses, but such information is not part of the Karel's primitive world. These parentheses will therefore be empty in standard Karel programs, but you must remember to include them nonetheless.

If Karel tries to do something illegal, such as moving through a wall or picking up a nonexistent beeper, an error condition occurs.

Karel’s commands, are not executed on their own. Instead, you need to incorporate them into a Karel program. You will have a chance to see a few simple Karel programs in Chapter 2!

Next Chapter