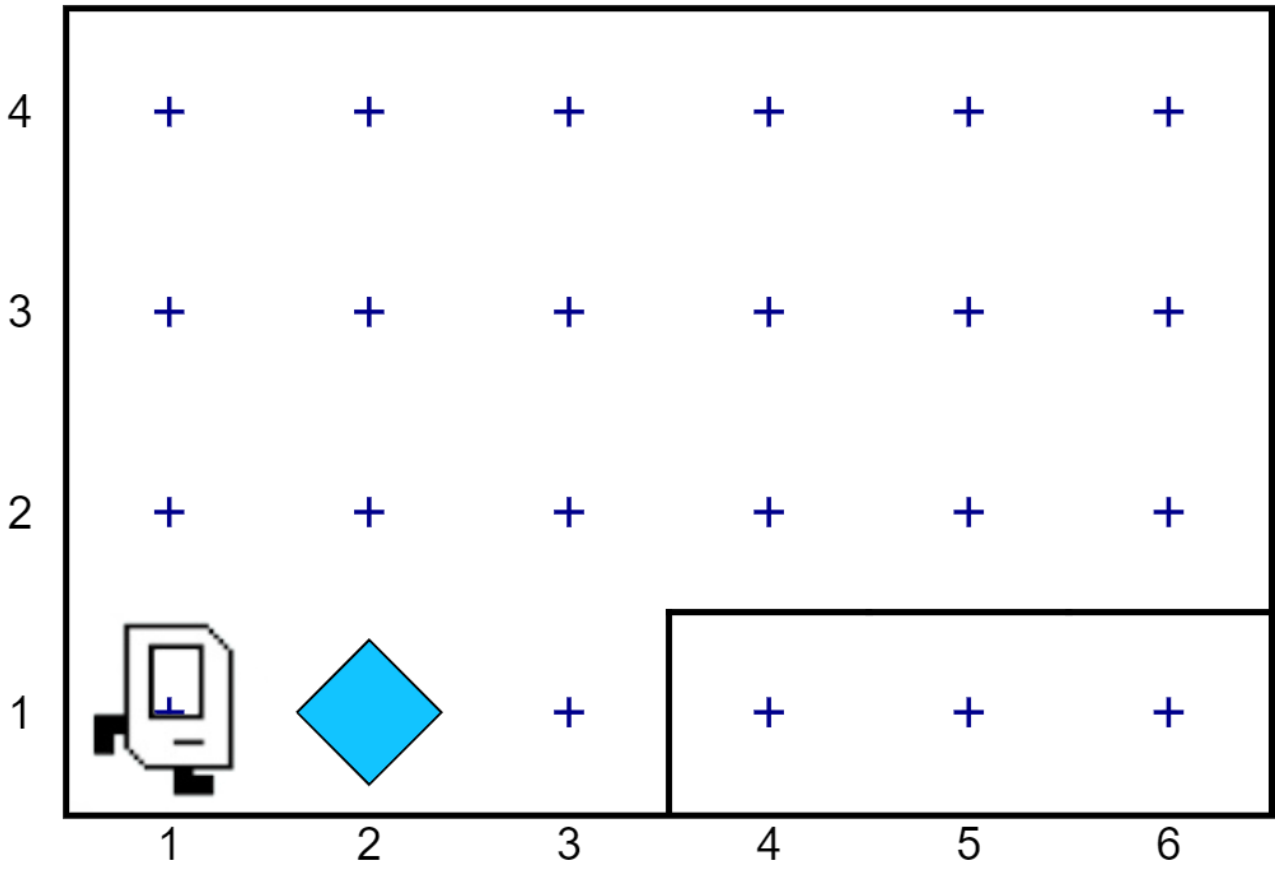
Chapter 3: Defining New Functions

In the last chapter we wrote a program to help Karel climb a simple ledge:

Example: FirstKarel

```
# File: FirstKarel.py

# ---------------------------

# Karel picks up a beeper and places it on a ledge.

from karel.stanfordkarel import *

def main() :

move()

pick_beeper()

move()

turn_left()

move()

turn_left()

turn_left()

turn_left()

move()

move()

put_beeper()

move()
```



Even though the `FirstKarel` program above demonstrates that it is possible to perform the `turn_right()` operation using only Karel's built-in commands, the resulting program is not particularly clear conceptually. In your mental design of the program, Karel turns right when it reaches the top of the ledge. The fact that you have to use three `turn_left()` commands to do so is annoying. It would be much simpler if you could simply say `turn_right()` and have Karel understand this command. The resulting program would not only be shorter and easier to write, but also significantly easier to read.

Defining New Commands

Fortunately, the Karel programming language makes it possible to define new commands simply by including new function definitions. Whenever you have a sequence of Karel commands that performs some useful task--such as turning right--you can define a new

function that executes that sequence of commands. The format for defining a new Karel function has much the same as the definition of `main()` in the preceding examples, which is a function definition in its own right. A typical function definition looks like this:

```
   def name():
```
commands that make up the body of the function

In this pattern, name represents the name you have chosen for the new function. To complete the definition, all you have to do is provide the sequence of commands in the lines after the colon, which are all indented by one tab. For example, you can define `turn_right()` as follows:

```
   def turn_right():
turn_left()
turn_left()
turn_left()
```

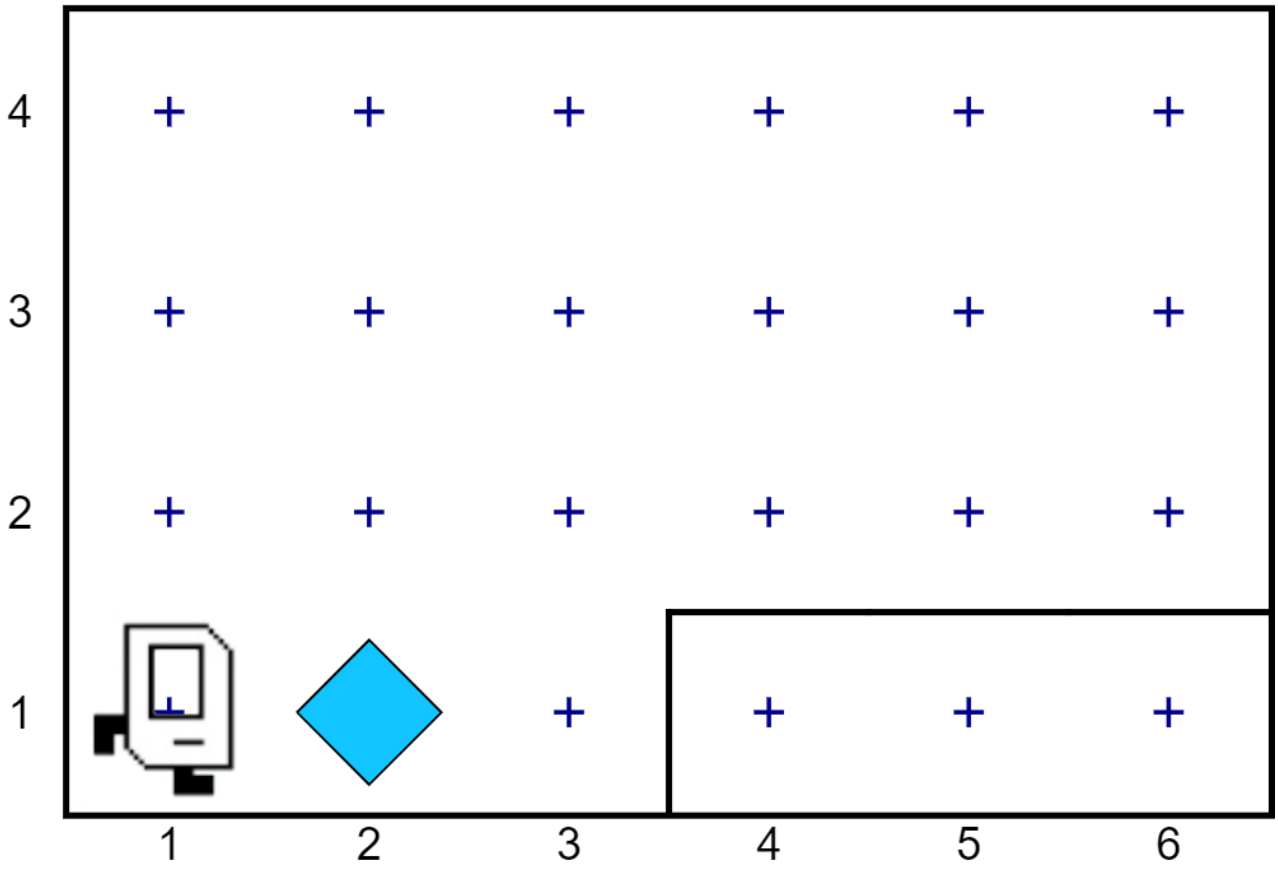
Similarly, you could define a new `turn_around()` function like this:

```
   def turn_around():
turn_left()
turn_left()
```

You can use the name of a new function just like any of Karel's built-in commands. For example, once you have defined `turn_right()`, you could replace the three `turn_left()` commands in the program with a single call to the `turn_right()` function. Here is a revised implementation of the program that uses `turn_right()`:

```
# File: BeeperPickingKarel.py

# ---------------------------

# The BeeperPickingKarel program defines a "main"

# function with three commands. These commands cause

# Karel to move forward one block, pick up a

# beeper and then move ahead to the next corner.

from karel.stanfordkarel import *

def main():

move()

pick_beeper()

move()

turn_left()
```

```
move()
turn_right()
move()
move()
put_beeper()
move()
def turn_right():
turn_left()
turn_left()
turn_left()
```



Function Code Blocks

A group of commands follow the colon `:` which are indented, is called a **code block**. The body of your function is a code block. Notice how the contents of a code block are all indented one tab in. This is important functionally since it allows Python to know what lines of code are in a given block.

You can define as many functions as you want. They should all be written one after another. You can't define a function inside another function.

Next Chapter