

1 - Meet Karel

2 - Programming

3 - New Functions

4 - Decomposition

5 - For Loops

6 - While Loops

7 - Conditionals

8 - Refinement

9 - Extra Features

10 - Reference

11 - Code

Chapter 7: If Statements

The final core programming control-flow construct to learn are conditional statements (if and if/else).

Basic Conditionals

An if/else statement executes an "if" code-block if and only if the provided `test` is true for the state of the world at the time the program reaches the statement. Otherwise the program executes the "else" code-block.

```

if test:
    if code-block
else:
    else code-block
    
```

To get a sense of where conditional statements might come in handy, let's write a program that has Karel invert a line of beepers. If a square previously had a beeper, Karel should pick it up. If a square has no beeper, Karel should put one down.

```

from karel.stanfordkarel import *

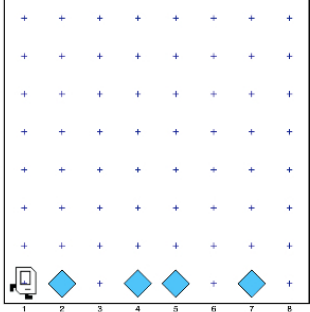
# the start of the program
def main():
    while front_is_clear():
        invert_beeper()
        move()
    # to prevent a fencepost bug
    invert_beeper()

# picks up a beeper if one is present
# puts down a beeper otherwise
def invert_beeper():
    # an if/else statement
    if beepers_present():
        pick_beeper()
    else:
        put_beeper()
        
```

▶ Run Program

Show Text Descriptions

Change World ▾



Note that an if statement does not need to have an else block -- in which case the statement operates like a while loop that only executes one time:

```

if test:
    if code-block
    
```

Conditions

That last example used a new condition. Here is a list of all of the conditions that Karel knows of:

Test	Opposite	What it checks
<code>front_is_clear()</code>	<code>front_is_blocked()</code>	Is there a wall in front of Karel?
<code>beepers_present()</code>	<code>no_beepers_present()</code>	Are there beepers on this corner?
<code>left_is_clear()</code>	<code>left_is_blocked()</code>	Is there a wall to Karel's left?
<code>right_is_clear()</code>	<code>right_is_blocked()</code>	Is there a wall to Karel's right?
<code>beepers_in_bag()</code>	<code>no_beepers_in_bag()</code>	Does Karel have any beepers in its bag?
<code>facing_north()</code>	<code>not_facing_north()</code>	Is Karel facing north?
<code>facing_south()</code>	<code>not_facing_south()</code>	Is Karel facing south?
<code>facing_east()</code>	<code>not_facing_east()</code>	Is Karel facing east?
<code>facing_west()</code>	<code>not_facing_west()</code>	Is Karel facing west?

Putting it all together

Congrats! You now know all of the core programming control-flow blocks. While you learned them with Karel, methods, while loops, for loops, if/else statements work in the same way in almost all major languages, including Python.

Now that you have the building blocks you can put them together to build solutions to ever more complex problems. To a large extent, programming is the science of solving problems by computer. Because problems are often difficult, solutions—and the programs that implement those solutions—can be difficult as well. In order to make it easier for you to develop those solutions, you need to adopt a methodology and discipline that reduces the level of that complexity to a manageable scale.