Karel - List of Commands

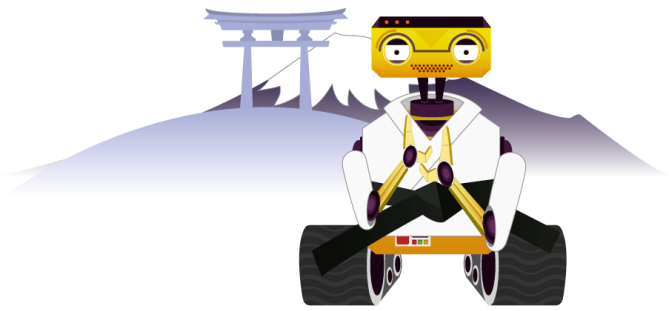
April 18, 2014[1]

Contents

[1] This document was prepared using the LaTeX module in NCLab

1 About this document

This document only covers selected basic functionality of Karel the Robot, and it is meant to be a reference rather than a learning material. If you like to learn computer programming with Karel, take the NCLab's interactive self-paced course *Earn Black Belt in Computer Programming!* More information about the course can be found at `https://nclab.com/karel/`.



2 Language variants

Karel can be used with English, Spanish, German, Czech, Polish, Italian, and French commands. All language versions of this document can be found in the menu of the Karel module. Any of these seven languages can also be chosen in Settings to be the main language for NCLab.

3 Karel module in NCLab

The Karel module in NCLab has four modes:

- First Steps (manual mode): Guide the robot using the mouse.
- Programming: Write and run programs.
- Designer: Design your own mazes.
- Games: Design your own games.

You can publish all your programs and games on the web and on social networks using the function "Publish to the web".

4 Basic commands

Karel knows five basic commands that are equivalent to the five buttons in manual mode:

- `go`: Make one step forward.

- `left`: Turn 90 degrees left.
- `right`: Turn 90 degrees right.
- `get`: Collect an object from the ground.
- `put`: Put an object on the ground.

5 Conditions

The `if - else` conditions help the robot check his surroundings and make decisions at runtime. Notice the indentation. The `else` branch can be omitted if not needed. Example:

```
if wall
  left
  left
else
  go
```

6 Loops

There are two types of loops: The `repeat` loop is used when the number of repetitions is known in advance:

```
repeat 5
  go
```

The `while` loop should be used when we do not know in advance how many repetitions will be needed:

```
while not wall
  go
```

Both loops can contain basic commands, conditions, other loops, and custom commands.

7 Custom commands

Custom commands should be defined for "smaller tasks" that are done more than once in the program:

```
def turnback
  repeat 2
    left
```

8 Logical expressions

Keyword `not` means *negation*. It returns `True` if the operand is `False` and vice versa. Example:

```
while not wall
  go
```

Keyword `and` returns `True` if both statements are `True`, else it returns `False`. Example:

```
while not wall and not home
  go
```

Keyword `or` returns `True` if at least one of the statements is `True`, otherwise it returns `False`. Example:

```
if gem or nugget
  get
```

9 GPS coordinates

Karel can retrieve his GPS coordinates using the commands `gpsx` and `gpsy`. In the Southwest corner of the maze, both `gpsx` and `gpsy` are `0`. In the Northeast corner, `gpsx` is `14` and `gpsy` is `11`. Both GPS coordinates can be used together with variables - see Section 11.

10 Randomness

The command `rand` returns randomly `True` or `False`. This allows Karel to make random moves. Example:

```
if rand
  left
else
  right
```

11 Variables

Variables are used to store information. Karel knows three types of variables:

1. *Numerical variables* store integer numbers:

   ```
   n1 = gpsx
   n2 = gpsy
   ```

   They can be increased by one via the command `inc` and decreased by one via the command `dec`:

```
inc(n1)
dec(n2)
```

They can also be increased or decreased by more than one:

```
inc(n1, 3)
dec(n2, 2)
```

2. *Text variables* store text strings:

```
my_name = "Karel"
```

3. *Logical variables* store logical values (`True`, `False`):

```
karel_is_west = (gpsx == 0)
karel_faces_north = north
```

Variables can be printed using the `print` command:

```
print "Value of n1 is", n1
```

12 Lists

Karel provides basic functionality of Python lists. As in Python, indices start with zero:

- `L = []` ... creates an empty list `L`.
- `len(L)` ... returns the length of the list `L`.
- `L[i]` ... returns item at position `i`.
- `L.append(x)` ... appends item `x` at the end of `L`.
- `y = L.pop()` ... removes and returns the last item from `L`.
- `y = L.pop(i)` ... removes and returns item at position `i`.
- `y = L.pop(0)` ... removes and returns the first item of `L`.
- `del L[i]` ... removes item at position `i`.

13 Functions

Functions in Karel are similar to custom commands, but they can return values via the command `return`. Example:

```
def measure_distance
  n = 0
  while not wall
    inc(n)
    go
  return n
```

14 Recursion

Karel is capable of recursion, which means that a custom command or function can call itself. Example:

```
def climb_stairs
  while wall
    left
    go
    right
    go
    climb_stairs
```

Commands or functions can also be mutually recursive (function A calls function B and at the same time function B calls function A).