

# *The Guess-My-Number Game*<sup>1</sup>

*Eric Bailey*

*January 14, 2017*<sup>2</sup>

In this game, you pick a number from 1 to 100, and the computer has to guess it.

## *Outline*

⟨src/guess.lisp 1⟩≡  
⟨reset the global state 3⟩

⟨define the guess-my-number function 5⟩

⟨define the smaller function 9⟩

⟨define the bigger function 12⟩

⟨define the start-over function 13⟩

Root chunk (not used in this document).

## *Example Session*

After loading ⟨src/guess.lisp 1⟩, you might have ⟨a session 2⟩ like this:

```
⟨a session 2⟩≡
> (start-over)
50
> (smaller)
25
> (bigger)
37
> (bigger)
43
> (smaller)
40
> (bigger)
41
> (bigger)
42
```

Root chunk (not used in this document).

Uses bigger 12, smaller 9, and start-over 13.

<sup>1</sup>

Conrad Barski. *Land of Lisp: Learn to Program in Lisp, One Game at a Time!*, chapter 2, pages 21–30. No Starch Press, 2010. ISBN 9781593273491. URL <http://landoflisp.com>

<sup>2</sup> Last updated October 8, 2017

As the name suggests, this chunk gets tangled into `src/guess.lisp`.

```
$ rlwrap sbcl --load src/guess.lisp
```

### Defining the Small and Big Variables

To give the computer a range of numbers in which to guess, we define the lower and upper limits, `*small*` and `*big*`, respectively. We'll need to *reset the global state 3* as such whenever we want to restart the game,

```
reset the global state 3≡
  (defparameter *small* 1)
  (defparameter *big* 100)
```

This code is used in chunks 1 and 13.

Defines:

`*big*`, used in chunks 4 and 8.  
`*small*`, used in chunks 4 and 11.

“Global variable names should start and end with asterisks (also known in this context as earmuffs)” [Brown and Rideau, 2017].

### Defining the Guess-My-Number Function

With `*small*` and `*big*` defined, we can tell the computer how to guess a number (`guess-my-number`) within those limits.

The basic algorithm is to *halve the sum of the limits and shorten the result 4*. To achieve that, we use Common Lisp's `ash` function to perform an *arithmetic right shift* by 1, i.e.  $\lfloor \text{sum} \times 2^{-1} \rfloor$ .

To *define the guess-my-number function 5*, we simply implement the algorithm described in pseudocode in Figure 1.

```
halve the sum of the limits and shorten the result 4≡
  (ash (+ *small* *big*) -1)
```

This code is used in chunk 5.

Uses `*big*` 3 and `*small*` 3.

```
define the guess-my-number function 5≡
  (defun guess-my-number ()
    halve the sum of the limits and shorten the result 4)
```

This code is used in chunk 1.

Defines:

`guess-my-number`, used in chunk 6.

Now, when we want to *have the computer guess a number 6*, we simply call `guess-my-number` as follows.

```
have the computer guess a number 6≡
  (guess-my-number)
```

This code is used in chunks 7, 9, 10, 12, and 13.

Uses `guess-my-number` 5.

Figure 1: The guessing algorithm

```
sum ← small + big
right shift sum by 1
return sum
```

### Defining the Smaller and Bigger Functions

To *<define the smaller function 9>*, we need to update the global state such that the next guess is *smaller* than the last, i.e. *<set \*big\* to one less than the last guess 8>* then *<have the computer guess a number 6>*.

```
<set *big* to one less than the last guess 8>≡
  (setf *big* <subtract one from the most recent guess 7>)
```

This code is used in chunk 9.

Uses *\*big\** 3.

```
<define the smaller function 9>≡
  (defun smaller ()
    <set *big* to one less than the last guess 8>
    <have the computer guess a number 6>)
```

This code is used in chunk 1.

Defines:

*smaller*, used in chunk 2.

To *<define the bigger function 12>*, we need to update the global state such that the next guess is *bigger* than the last, i.e. *<set \*small\* to one greater than the last guess 11>* then *<have the computer guess a number 6>*.

```
<set *small* to one greater than the last guess 11>≡
  (setq *small* <add one to the most recent guess 10>)
```

This code is used in chunk 12.

Uses *\*small\** 3.

```
<define the bigger function 12>≡
  (defun bigger ()
    <set *small* to one greater than the last guess 11>
    <have the computer guess a number 6>)
```

This code is used in chunk 1.

Defines:

*bigger*, used in chunk 2.

### Defining the Start-Over Function

At this point, to *<define the start-over function 13>* is trivial: we simply *<reset the global state 3>* then *<have the computer guess a number 6>*.

```
<define the start-over function 13>≡
  (defun start-over ()
    <reset the global state 3>
    <have the computer guess a number 6>)
```

This code is used in chunk 1.

Defines:

*start-over*, used in chunk 2.

To appropriately adjust *\*big\**,  
*<subtract one from the most recent guess 7>*.

```
<subtract one from the most recent guess 7>≡
  (1- <have the computer guess a number 6>)
```

This code is used in chunk 8.

To appropriately adjust *\*small\**, *<add one to the most recent guess 10>*.

```
<add one to the most recent guess 10>≡
  (1+ <have the computer guess a number 6>)
```

This code is used in chunk 11.

*Full Listing**<src/guess.lisp 1>:*

```

(defparameter *small* 1)
(defparameter *big* 100)

(defun guess-my-number ()
  (ash (+ *small* *big*) -1))

(defun smaller ()
  (setf *big* (1- (guess-my-number)))
  (guess-my-number))

(defun bigger ()
  (setq *small* (1+ (guess-my-number)))
  (guess-my-number))

(defun start-over ()
  (defparameter *small* 1)
  (defparameter *big* 100)
  (guess-my-number))

```

*Index*

\*big\*: [3](#), [4](#), [8](#)  
 \*small\*: [3](#), [4](#), [11](#)  
 bigger: [2](#), [12](#)  
 guess-my-number: [5](#), [6](#)  
 smaller: [2](#), [9](#)  
 start-over: [2](#), [13](#)

## Chunks

⟨a session 2⟩ [2](#)  
 ⟨add one to the most recent guess 10⟩ [10](#), [11](#)  
 ⟨define the **bigger** function 12⟩ [1](#), [12](#)  
 ⟨define the **guess-my-number** function 5⟩ [1](#), [5](#)  
 ⟨define the **smaller** function 9⟩ [1](#), [9](#)  
 ⟨define the **start-over** function 13⟩ [1](#), [13](#)  
 ⟨halve the sum of the limits and shorten the result 4⟩ [4](#), [5](#)  
 ⟨have the computer guess a number 6⟩ [6](#), [7](#), [9](#), [10](#), [12](#), [13](#)  
 ⟨reset the global state 3⟩ [1](#), [3](#), [13](#)  
 ⟨set **\*big\*** to one less than the last guess 8⟩ [8](#), [9](#)  
 ⟨set **\*small\*** to one greater than the last guess 11⟩ [11](#), [12](#)  
 ⟨src/guess.lisp 1⟩ [1](#)  
 ⟨subtract one from the most recent guess 7⟩ [7](#), [8](#)

## References

Conrad Barski. *Land of Lisp: Learn to Program in Lisp, One Game at a Time!*, chapter 2, pages 21–30. No Starch Press, 2010. ISBN 9781593273491. URL <http://landoflisp.com>.

Robert Brown and François-René Rideau. Google Common Lisp Style Guide: Global variables and constants. [https://google.github.io/styleguide/lispguide.xml?showone=Global\\_variables\\_and\\_constants#Global\\_variables\\_and\\_constants](https://google.github.io/styleguide/lispguide.xml?showone=Global_variables_and_constants#Global_variables_and_constants), September 2017. Accessed: 2017-10-08.