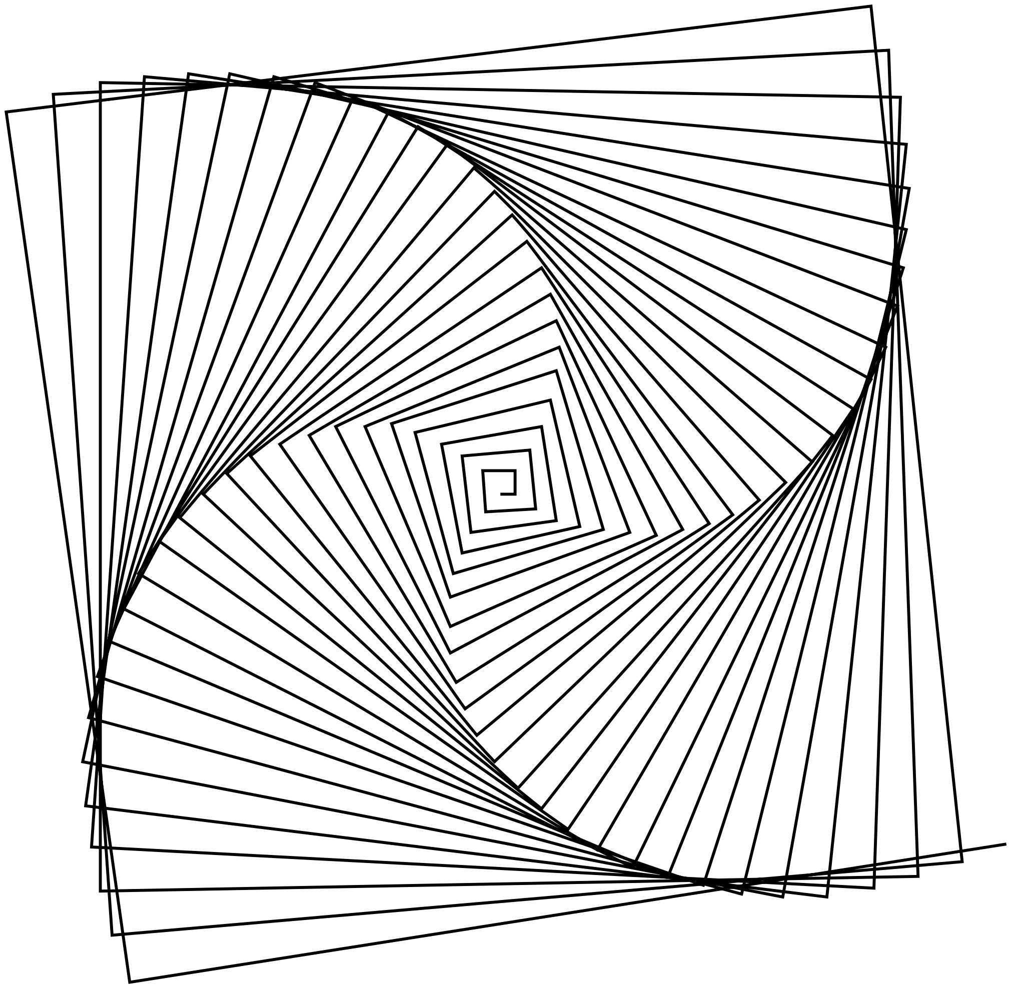
1 Loops and Variables



In this chapter we show how to use variables and loops together. We start by analyzing a simple problem that shows the need for using variables with loops. Then you shall experiment with some other problems.

1 A Motivating Example

Try to generate the odd stairway shown in Figure 1.1. One way to start would be by generating a normal stairway and modifying it. One of the problems that you should be facing and need to solve is that the length of each step is constantly growing.

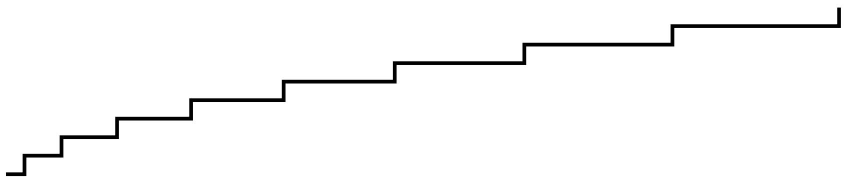


Figure 1.1: A strange stair

One of the simplest solutions is shown in Script 1.1, where the length of each step grows by 10 pixels. However, such a solution is not satisfactory because we have to compute the length of the next each step manually. And also because we have to stupidly repeat the same sequence of messages.

Script 1.1 (*Strange stair)*

```
| pica |
pica := Bot new.
pica go: 10.
pica turnLeft: 90.
pica go: 5.
pica turnRight: 90.
pica go: 20.
pica turnLeft: 90.
pica go: 5.
pica turnRight: 90.
pica go: 30.
pica turnLeft: 90.
pica go: 5.
pica turnRight: 90.
pica go: 40.
pica turnLeft: 90.
pica go: 5.
pica turnRight: 90.
...
```

We would like to be able to use the power of variables combind with the power of loops. First you can avoid repeating the sequence of messages by using the `timesRepeat:` method. Second if we study Script 1.1, we see that the length of a step is the length of the previous step plus 10 pixels. Here, like this: 20 = 10 + 10, 30 = 20 + 10, 40 = 30 + 10, and so on as shown by Figure 1.2.

If we use the variable `length` to represent the length of a step, the length of the second step is the length of the first step plus 10, and the length of the third step is the length of the second step plus 10... and likewise for the other steps. We can express this by the expression `length := length + 10` which increases the value of the variable `length` by 10.

Let's combine everything! Now if we take the script of a normal stairway (Script 1.2), introduce the variable `length` we obtain the same stairway (Script 1.3). Then if we add the line `length := length + 10` to change the `length` value in each step of the loop, we get“ the stairway we want (Script 1.4).

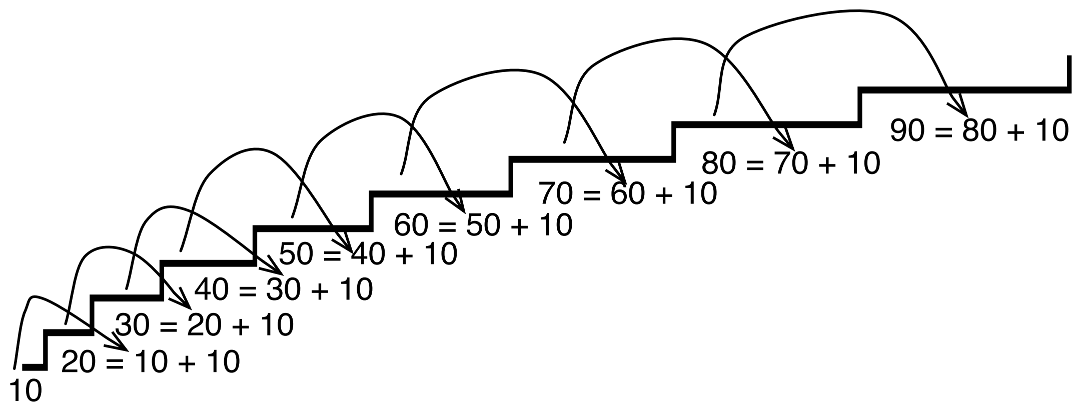


Figure 1.2: The length of a step is the length of the previous step plus 10 pixels.

Script 1.2 (*A stairway with normal steps)*

```
| pica |
pica := Bot new.
10 timesRepeat:
        [ pica go: 10.
        pica turnLeft: 90.
        pica go: 5.
        pica turnRight: 90 ]
```

Script 1.3 (*A stairway with normal steps using* length*)*

```
| pica **length** |
pica := Bot new.
**length := 10.**
10 timesRepeat:
        [ pica go: **length.**
        pica turnLeft: 90.
        pica go: 5.
        pica turnRight: 90 ]
```

Script 1.4 (*The Solution)*

```
| pica length |
pica := Bot new.
length := 10.
10 timesRepeat:
        [ pica go: length.
        pica turnLeft: 90.
        pica go: 5.
        pica turnRight: 90.
        length := length + 10 ]
```

In Script 1.4, the sequence of messages in the loop first makes the robot go forward a distance given by the value of the variable length (10 the first time). Then the robot turns, draws the riser (straight up) and turns again. Then the value of the variable length is increased by 10 and the loop restarts but with the variable length having the new larger value (20 for the second repetition). The whole process is repeated 10 times.

You can't leave out the expression length := length + 10 otherwise the value of the variable will never change.

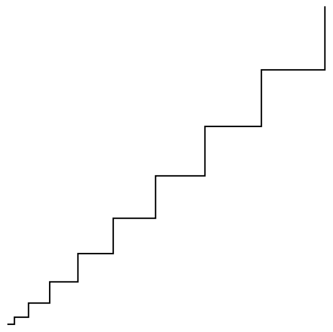
Experiment 1.1
Try changing the last line of the loop — for example put length := length + 15. Then try moving the line to different places in the loop. Can you explain what happens when you move the last line of the loop to the beginning of the loop?

If you need more time to understand Script 1.4, we suggest you think carefully about the value of the variable length, especially at the beginning and end of the loop. Figure out what value length has in each message for three repetitions of the loop. If necessary read Chapter **??** again.

2 Practicing: Mazes, Spirals and More

Let's see how combining variables and loops helps us solve some other problems.

Experiment 1.2
Change Script 1.4 to produce the picture shown on the right.



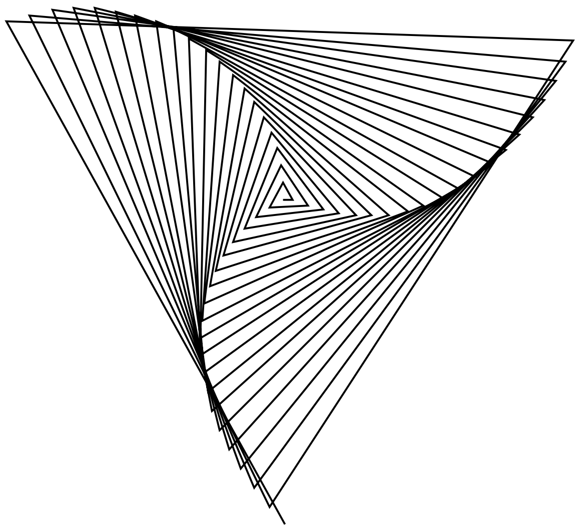
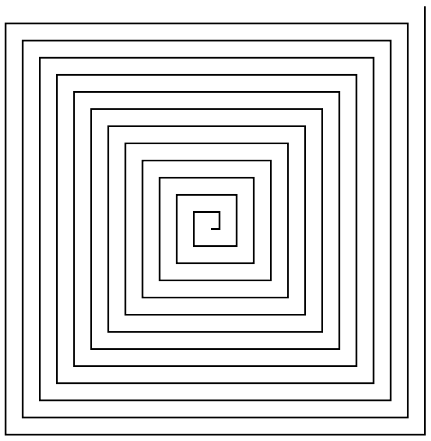
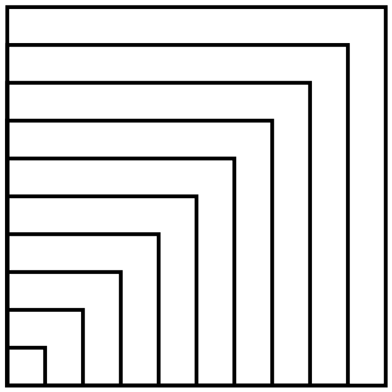


Figure 1.3: A nice spiral

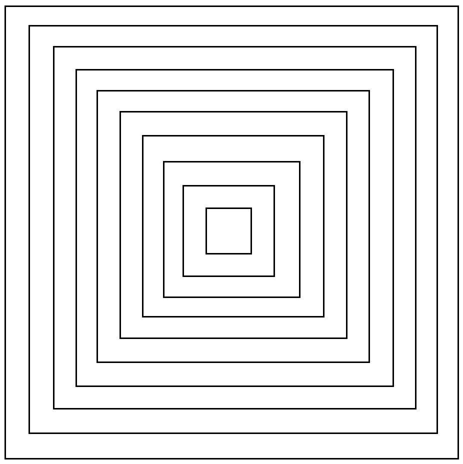
Experiment 1.3 (*Maze)*
Define a script that reproduces the drawing shown on the right. In addition, by turning through different angles you should be able to recreate the picture at the beginning of the chapter, as well as the spiral shown in Figure 1.3.



Experiment 1.4 (*Russian squares)*
Build the squares of different sizes as shown on the right. Start for example by defining a loops drawing 10 times the same square, then introduce a variable representing the size of the square and finally make it grows over time.



Experiment 1.5 (*Corridor)*
Build the squares of different size shown on figure on the right. Again start by drawing a square but draw it starting from its center so that changing its size will automatically draw another square centered around the first one. Then define the square inside a loop, then introduce a variable representing the size of the square and finally make it grows over time.



3 Some Important Points

Now that you saw the overall process and experimented, we want to stress some important points. Script 1.5 shows a skeleton of a typical situation: (1) first a variable is declared, then (2) it is initialized. After inside the loop the variable is used to perform some computations and (4) its value is changed.

Script 1.5 (*Skeleton of using a variable in a loop)*

```
| length pica |                        "variable declaration"
...
length := 10.                          "initialization of the variable"
...
10 timesRepeat:
        [ pica go: length.             "variable use"
        ....
        length := length + 10 ]        "variable change of value"
```

Variable Initialization. When you introduce a variable in a loop, you have to pay attention to the first value of the variable, that is, when it is initialized. Normally variable initialization should outside of the loop, else the variable value would be always reinitialized at each loop step and the variable value would not change its value.

Variable Value Use and Change. Inside the loop the variable value is often used to perform other computations such as computing other variable values. Then it eventually gets its value changed. In the stairway example, the expression `length := length + 10` increases the value of length based on its preceding value. What is important to understand is that the new change value will be the variable value for the next step of the loop as illustrated by Figure 1.4.

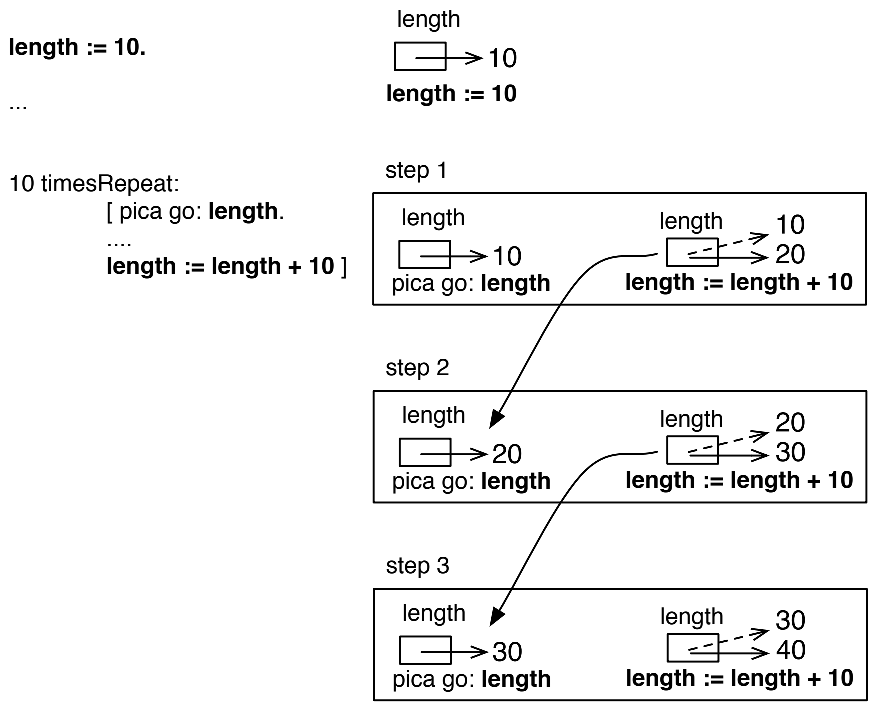
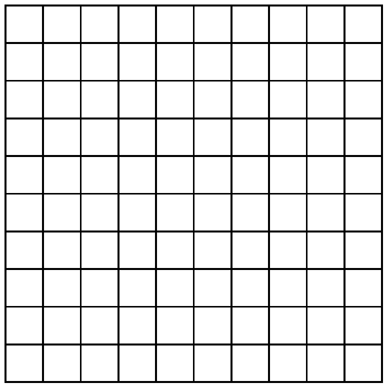


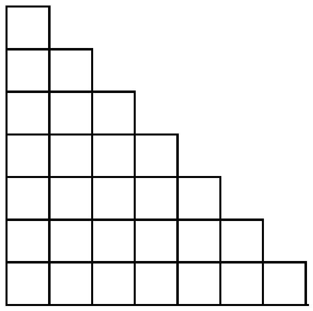
Figure 1.4: The length of a step is the length of the previous step plus 10 pixels. The last value of a variable in the loop body is the variable value for the next step.

4 Advanced Experiments

Experiment 1.6 (*Squares)*
Define a script that creates the construction shown on the right. This experiment is a bit more complicated than the earlier experiments. Several solutions are possible but to help you we propose some hints: For example start draw a line composed of a certain number of squares, then define a loop that draws this line several times, finally introduce another loop that draws several lines.



Experiment 1.7 (*Pyramid)*
Define a script that creates the construction shown on the right. This experiment is a bit more complicated than the earlier experiments. There are several different scripts that will work. Hints: in the solution of the previous experiment introduce a variable representing the number of lines and change it at each step of the loop.



Summary

- When introducing a variable inside a loop, be careful about the first value of the variable, that is how the variable is initialized. Normally the initialization occurs outside the loop else the value would always stay the same.
- Pay attention that the last value change of a variable in a loop body is the variable value of the next loop step.