algorithms and programs

prof. dr. nicolas meseth

1 be the machine
2 the instruction follower
3 four building blocks
4 it does what is written
5 working with the assistant

part 1

be the machine

who is playing?



image: ai-generated (gpt-image-2)

1. look at the card.
2. decide which colour it is.
3. write down its first letter.
4. wait for the next card.

that was an algorithm,
and you were the machine that ran it.

steps so precise that you can follow them without understanding what they are for.

1 look at the card.
2 decide which colour it is.
3 write down its first letter.
4 wait for the next card.

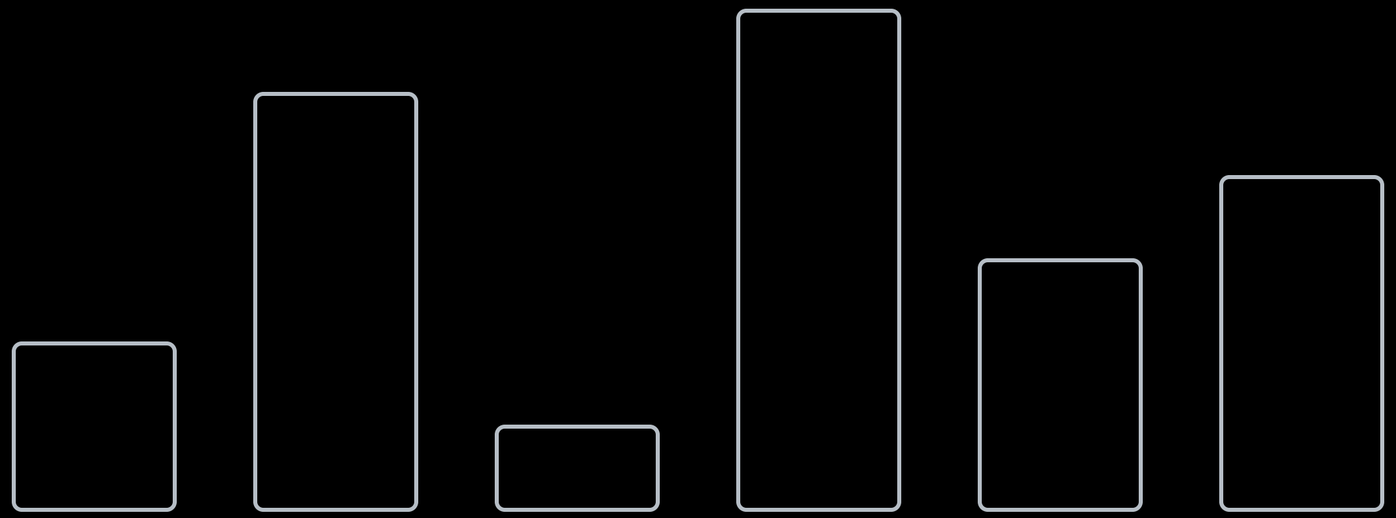
three of these lines are easy for a program, but deciding is the hard part.

1. stand in a row.
2. look at your right neighbour.
3. if they are shorter than you, swap.
4. at the end of the row, start again.
5. if a full pass had no swap, stop.

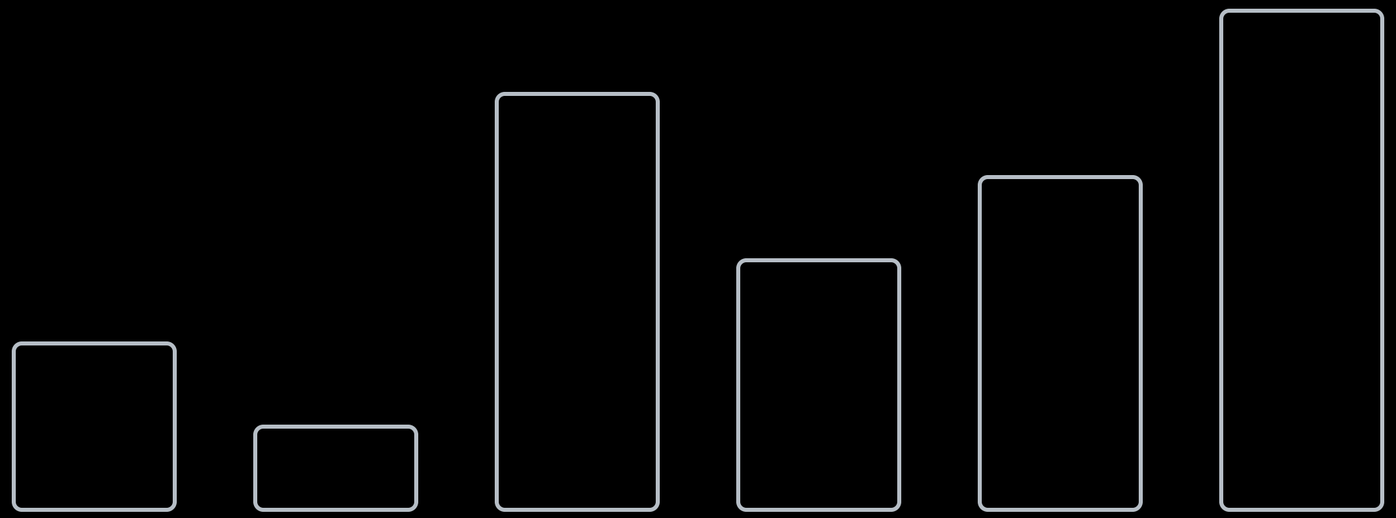
what you just ran is called bubble sort, and a computer can sort the same way.

it follows the same five rules, only millions of times faster.

bubble sort on six bars

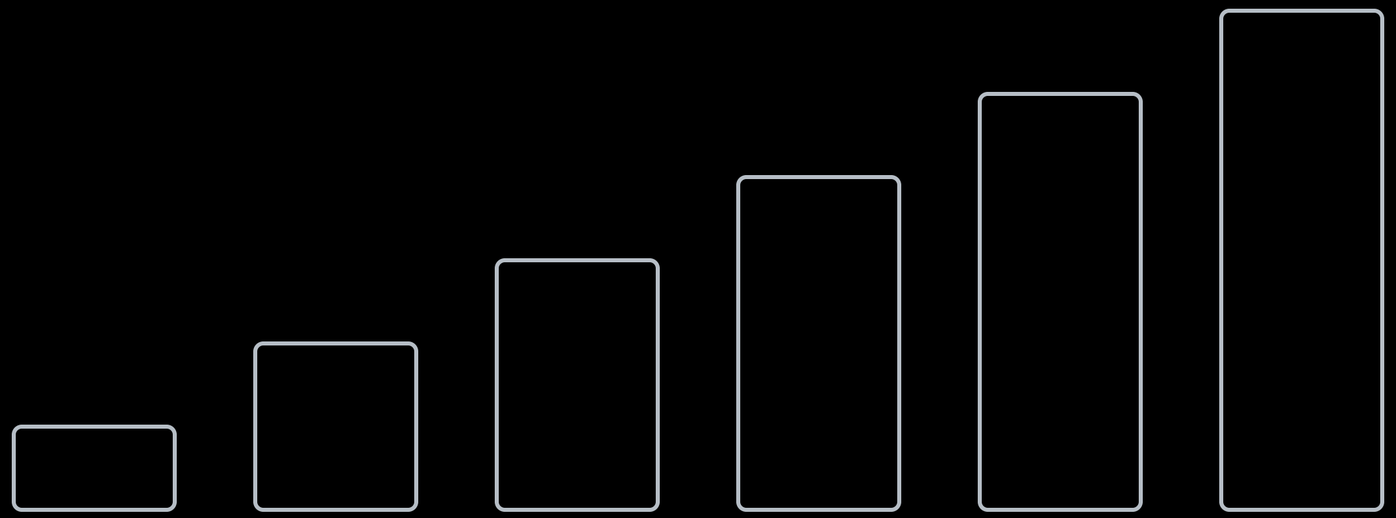


bubble sort on six bars



pass 1: 3 swaps

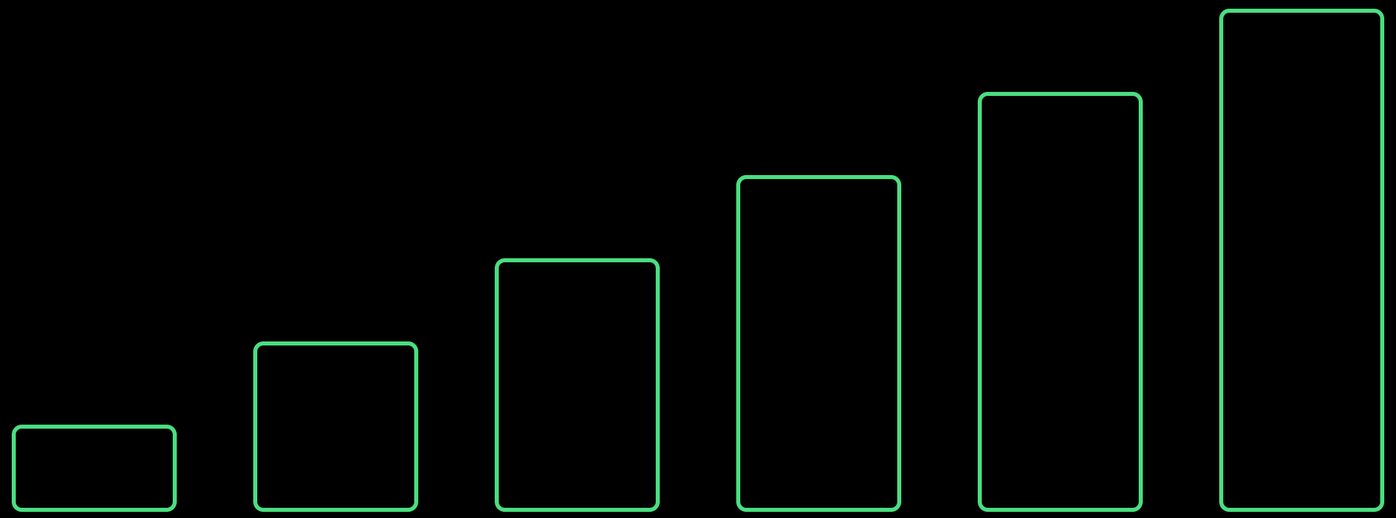
bubble sort on six bars



pass 1: 3 swaps

pass 2: 3 swaps

bubble sort on six bars



pass 1: 3 swaps

pass 2: 3 swaps

pass 3: no swap, done

the procedure knows it is done when a full pass finds nothing left to swap.

a program is an algorithm written for a machine.

and it must be complete, because the machine adds nothing.

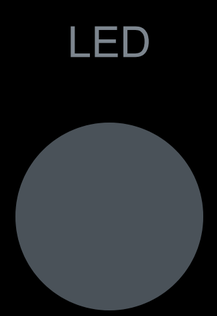
part 2

the instruction follower

a program is a list of instructions
that a computer follows
exactly and in order.

```
import time
from lifi_hardware import led

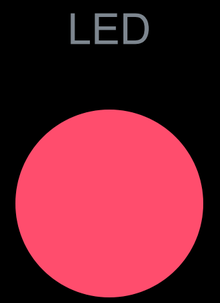
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```



```
import time
from lifi_hardware import led

led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```

turns red



time

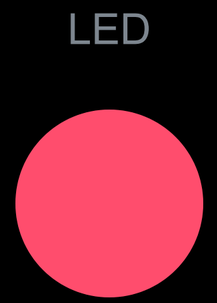
0.0 s

```
import time
from lifi_hardware import led

led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```

turns red

waits one second



time

1.0 s

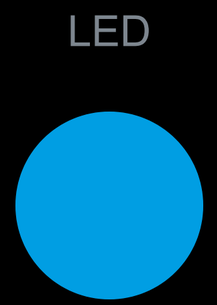
```
import time
from lifi_hardware import led

led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```

turns red

waits one second

turns blue



time

1.0 s

```
import time
from lifi_hardware import led

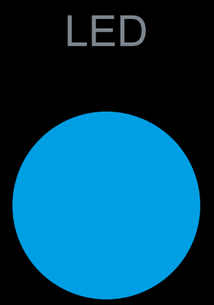
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```

turns red

waits one second

turns blue

waits one second



time

2.0 s

the machine runs one line after the other, from top to bottom, exactly as written.

```
led.set_color(255, 0, 0)
```

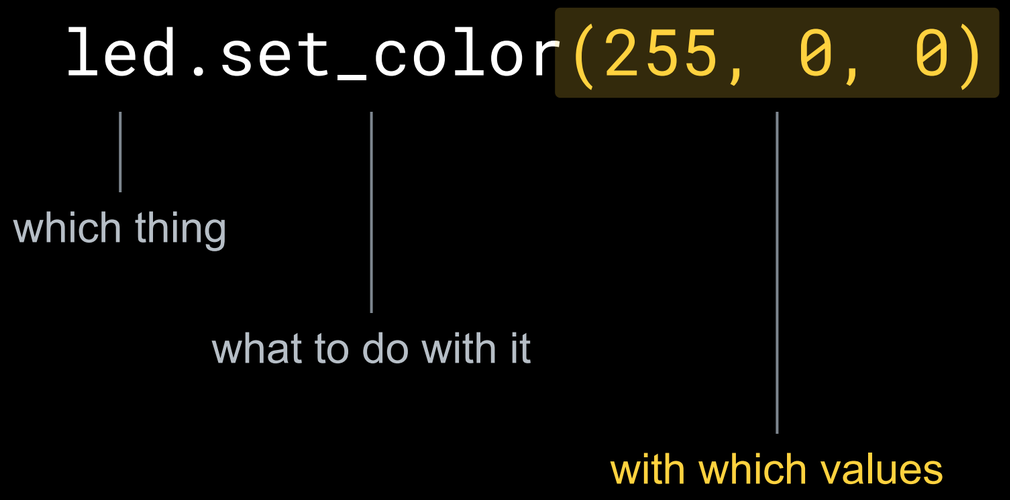
```
led.set_color(255, 0, 0)
```

which thing

```
led.set_color(255, 0, 0)
```

which thing

what to do with it


led.set_color(255, 0, 0)
which thing
what to do with it
with which values

```
led.set_color(255, 0, 0)
```



red

```
led.set_color(0, 0, 255)
```



blue

```
led.set_color(255, 255, 0)
```



yellow

the instruction stays the same, and the values decide what the LED shows.

part 3

four building blocks

```
import time
from lifi_hardware import led

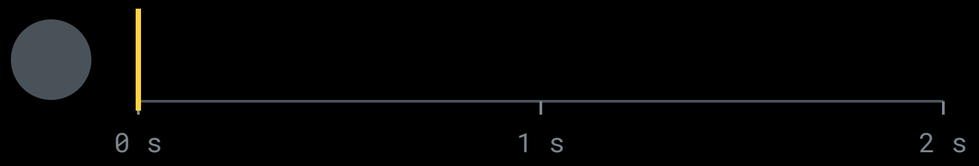
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```

swap two lines, and you have a different program.

with a pause, and without

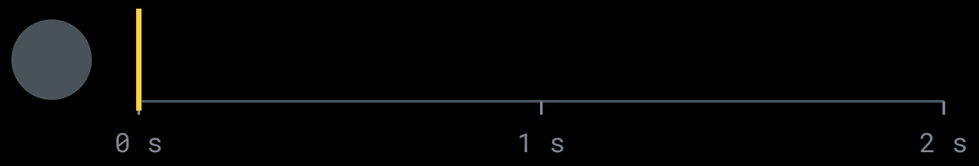
a pause between the colours

```
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```



no pause between them

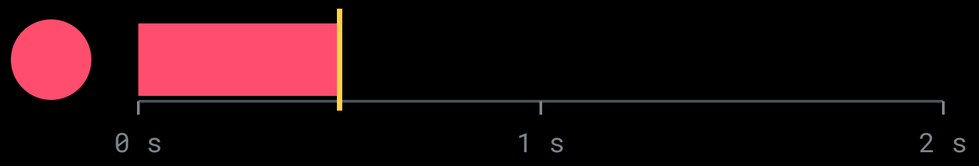
```
led.set_color(255, 0, 0)
led.set_color(0, 0, 255)
time.sleep(2)
```



with a pause, and without

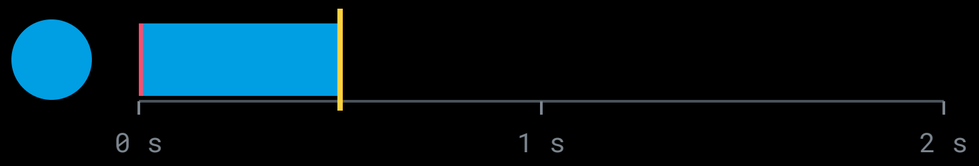
a pause between the colours

```
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```



no pause between them

```
led.set_color(255, 0, 0)
led.set_color(0, 0, 255)
time.sleep(2)
```



with a pause, and without

a pause between the colours

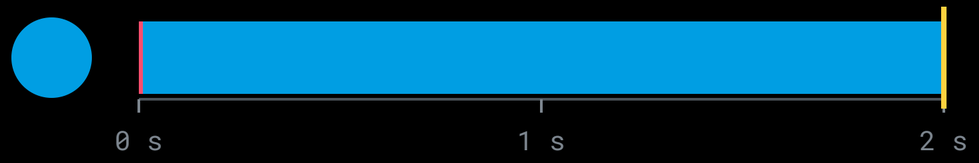
```
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
time.sleep(1)
```



you see red, then blue.

no pause between them

```
led.set_color(255, 0, 0)
led.set_color(0, 0, 255)
time.sleep(2)
```



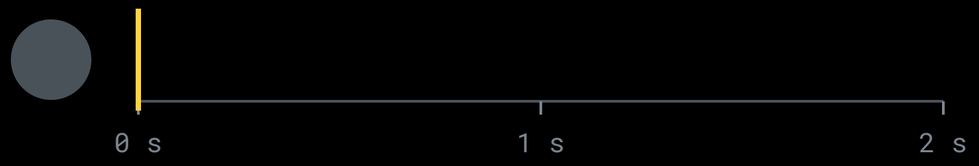
you only see blue.

the program is not broken, it is only too fast for your eyes.

pause in the middle, or at the end

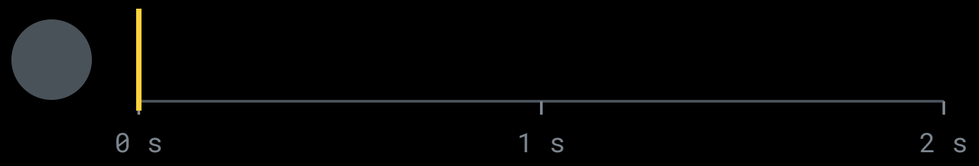
pause in the middle

```
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
```



pause at the end

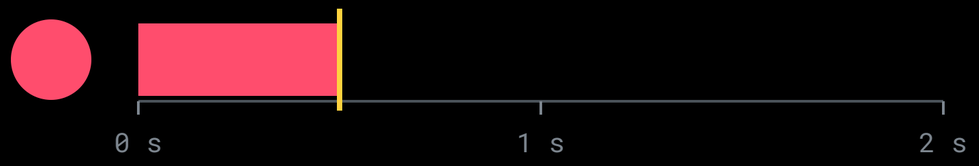
```
led.set_color(255, 0, 0)
led.set_color(0, 0, 255)
time.sleep(1)
```



pause in the middle, or at the end

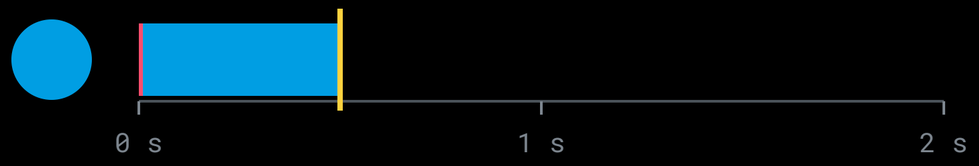
pause in the middle

```
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
```



pause at the end

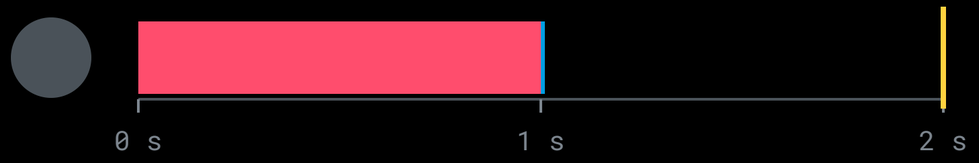
```
led.set_color(255, 0, 0)
led.set_color(0, 0, 255)
time.sleep(1)
```



pause in the middle, or at the end

pause in the middle

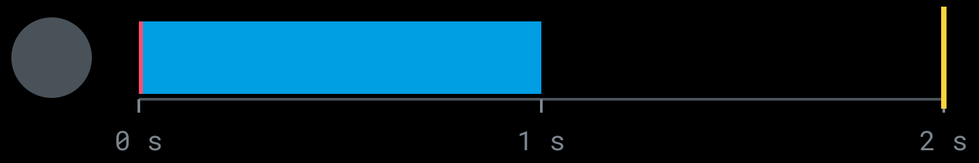
```
led.set_color(255, 0, 0)
time.sleep(1)
led.set_color(0, 0, 255)
```



you see red.

pause at the end

```
led.set_color(255, 0, 0)
led.set_color(0, 0, 255)
time.sleep(1)
```



you see blue.

after the last line the program ends, and the LED goes off.

no line is missing, and still it is a different program.

```
import time
from lifi_hardware import led

bright = 120
led.set_color(bright, 0, 0)
time.sleep(1)
led.set_color(0, bright, 0)
time.sleep(1)
led.set_color(0, 0, bright)
time.sleep(1)
```

one place to change instead of three.

```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    time.sleep(1)
```

```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    time.sleep(1)
```



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    time.sleep(1)
```



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    time.sleep(1)
```



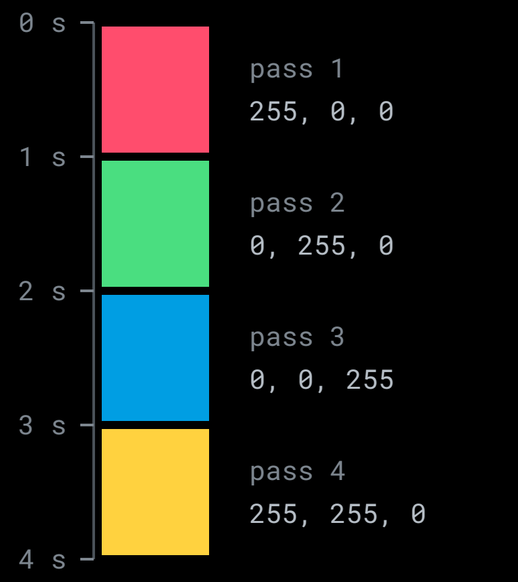
```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    time.sleep(1)
```



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    time.sleep(1)
```



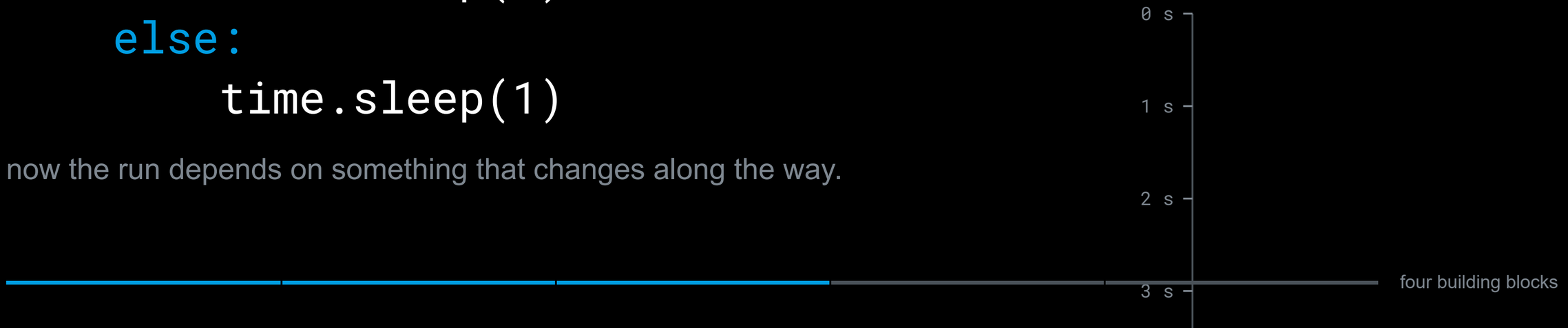
four passes, four seconds

the same two lines run four times, with one value and one second per pass.

```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    if (r, g, b) == (255, 255, 0):
        time.sleep(3)
    else:
        time.sleep(1)
```

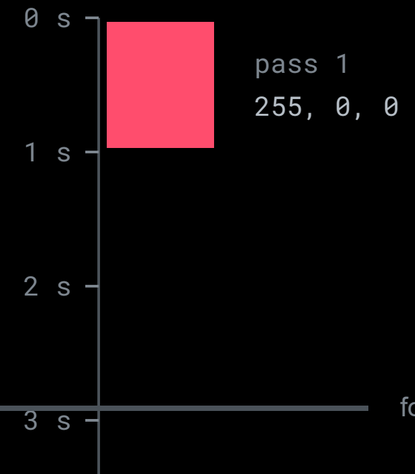
now the run depends on something that changes along the way.



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    if (r, g, b) == (255, 255, 0):
        time.sleep(3)
    else:
        time.sleep(1)
```

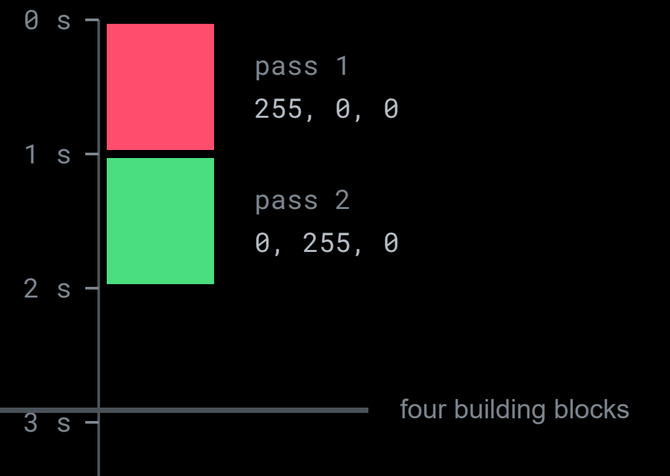
now the run depends on something that changes along the way.



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    if (r, g, b) == (255, 255, 0):
        time.sleep(3)
    else:
        time.sleep(1)
```

now the run depends on something that changes along the way.



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    if (r, g, b) == (255, 255, 0):
        time.sleep(3)
    else:
        time.sleep(1)
```

now the run depends on something that changes along the way.



```
import time
from lifi_hardware import led

colors = [(255, 0, 0), (0, 255, 0),
          (0, 0, 255), (255, 255, 0)]
for r, g, b in colors:
    led.set_color(r, g, b)
    if (r, g, b) == (255, 255, 0):
        time.sleep(3)
    else:
        time.sleep(1)
```

now the run depends on something that changes along the way.



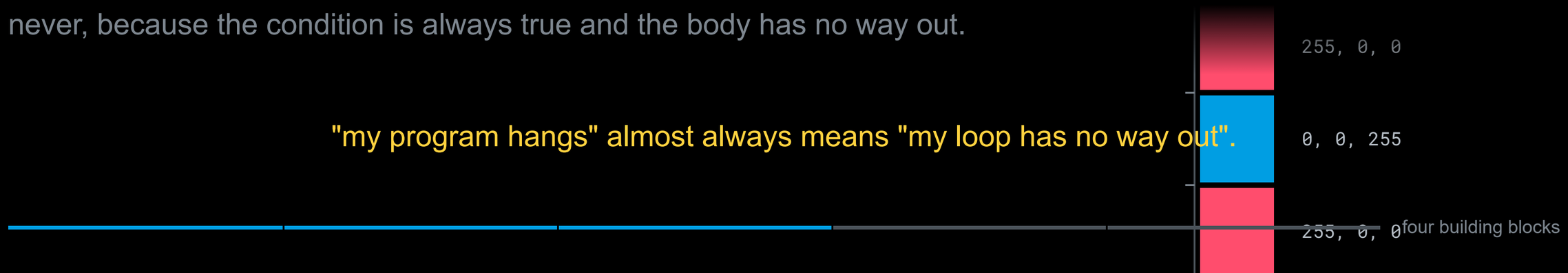
when does this end?

```
import time
from lifi_hardware import led

while True:
    led.set_color(255, 0, 0)
    time.sleep(1)
    led.set_color(0, 0, 255)
    time.sleep(1)
```

never, because the condition is always true and the body has no way out.

"my program hangs" almost always means "my loop has no way out".



working alone is not enough.

two correct programs can still differ by far in how long they take.

	100 numbers	1000 numbers
ask every number	99	999
halve the range	7	10

both find the number, but only one of them stays fast as the range grows.

being correct is the minimum, and after that you compare how the cost grows.

part 4

it does what is written

an error message

```
Traceback (most recent call last):
  File "C:\lifi\blink.py", line 5, in <module>
    time.slep(1)
    ^^^^^^^^^
AttributeError: module 'time' has no attribute 'slep'. Did you mean: 'sleep'?
```

an error message

```
Traceback (most recent call last):
  File "C:\lifi\blink.py", line 5, in <module>
    time.slep(1)
    ^^^^^^^^^
AttributeError: module 'time' has no attribute 'slep'. Did you mean: 'sleep'?
```

which file, and which line

an error message

```
Traceback (most recent call last):
  File "C:\lifi\blink.py", line 5, in <module>
    time.slep(1)
    ^^^^^^^^^^
AttributeError: module 'time' has no attribute 'slep'. Did you mean: 'sleep'?
```

which file, and which line

what stands there

an error message

```
Traceback (most recent call last):
  File "C:\lifi\blink.py", line 5, in <module>
    time.slep(1)
    ^^^^^^^^^
AttributeError: module 'time' has no attribute 'slep'. Did you mean: 'sleep'?
```

which file, and which line

what stands there

what does not exist, and what Python guesses you meant

an error message is information, not a verdict.

read it out loud, because most of the time it says exactly what is wrong.

part 5

working with the assistant

two ways to ask

```
write me a program for the LED.
```

this gets you code.

two ways to ask

```
write me a program for the LED.
```



this gets you code.

```
the LED should be red for three seconds, then blue.
explain each line before you write it.
```



this gets you understanding.

never keep code
you cannot change.

every hand-over ends with a small change: another colour, another waiting time.

if you cannot change it, you have not taken it over but only copied it.