

what is a program?

digitalisierung und programmierung · prof. dr. nicolas meseth

- 1 you were the receiver
- 2 the instruction follower
- 3 four building blocks
- 4 it does what is written
- 5 working with the assistant

part 1

you were the receiver

who is playing?



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

you were the receiver

you have been the receiver already.

look at the card.

decide which colour it is.

write down the matching symbol.

wait for the next one.

that was an algorithm.
you were the machine.

steps so precise you could follow them without understanding a thing.

four lines, one of them hard

look at the card.

decide which colour it is.

write down the matching symbol.

wait for the next one.

one of these four will take you three weeks to hand over.

one more algorithm, live

- 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 a full pass without a swap: done.

you just ran bubble sort.
computers sort like this, only faster.

it ended because a full pass had no swap left. remember that.

a program is an algorithm
written for a machine.

and it must be complete: the machine adds nothing.

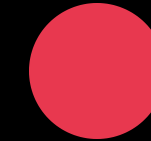
part 2

the instruction follower

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

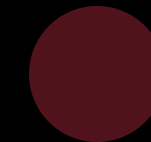
the list at work

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



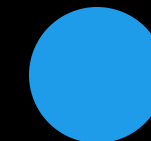
turns red

```
time.sleep(1)
```



stays red,
one second passes

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



turns blue

one line at a time, top to bottom. every line changes what you see.

one line

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

one line

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

|
which thing

one line

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

which thing

what to do with it

one line

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

which thing

what to do with it

with which values

same instruction, different values

```
from lifi_hardware import led
```

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

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

same instruction, different values, different result.

part 3

four building blocks

1 · one after another

```
import time
from lifi_hardware import led

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

two lines swapped is a different program.

with a pause, and without

with a pause

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



red for one second, then blue.

without a pause

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



red for a millisecond: you never see it.

the program is not broken. it is 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)
```



one second of red, then blue.

pause at the end

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



one millisecond of red, then a second of blue.

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

2 · a value with a name

```
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)
```

one place to change instead of three.

3 · again

```
import time
from lifi_hardware import led

colours = [(255, 0, 0), (0, 255, 0),
            (0, 0, 255), (255, 255, 0)]

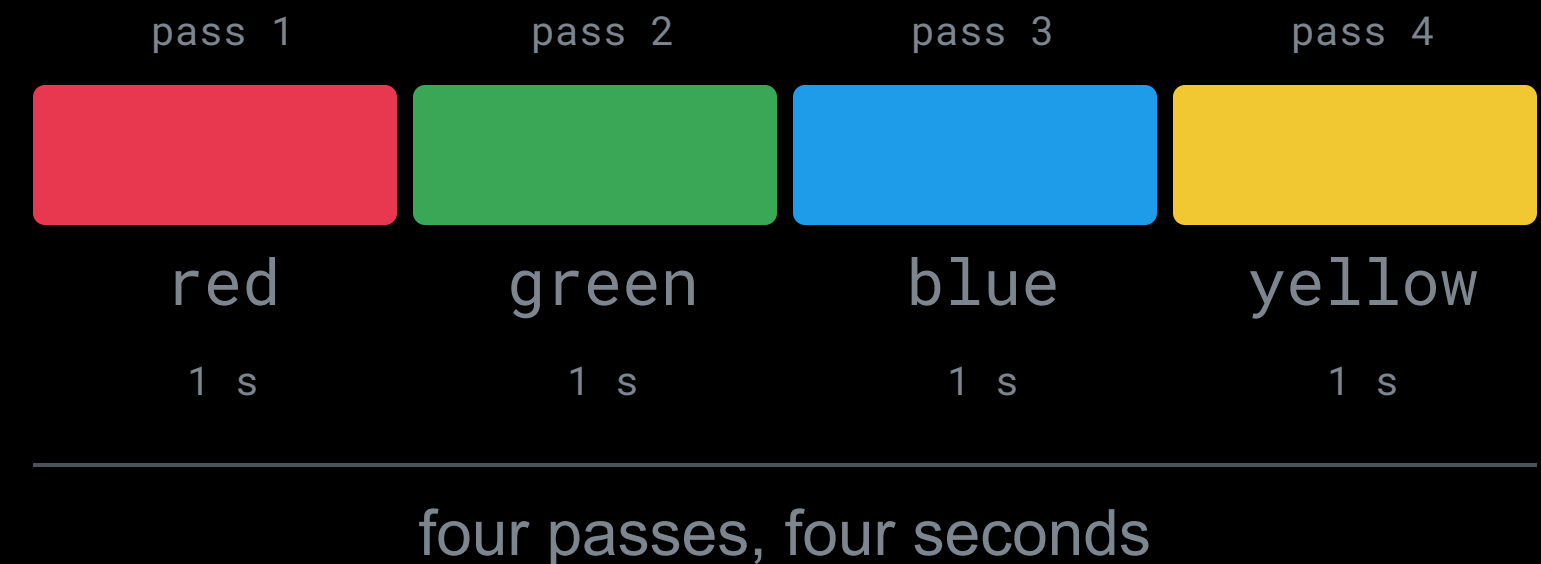
for r, g, b in colours:
    led.set_color(r, g, b)
    time.sleep(1)
```

the loop you write, and what actually happens

the loop you write

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

what actually happens



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

4 · only if

```
import time
from lifi_hardware import led

colours = [(255, 0, 0), (0, 255, 0),
           (0, 0, 255), (255, 255, 0)]

for r, g, b in colours:
    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: the condition is always true, and the body has no way out.

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

what does it cost?

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

both find the number. one of them is unusable at scale.

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

part 4

it does what is written

an error message

```
traceback (most recent call last):  
  file "blink.py", line 7, in <module>  
    led.set_color(255, 0, 0)  
attributeerror: no attribute 'set_colour'
```

an error message

```
Traceback (most recent call last):  
  File "blink.py", line 7, in <module>  
    led.set_color(255, 0, 0)  
AttributeError: no attribute 'set_colour'
```

—— which file, and which line

an error message

```
traceback (most recent call last):  
  file "blink.py", line 7, in <module>  
    led.set_color(255, 0, 0)  
attributeerror: no attribute 'set_colour'
```

—— which file, and which line

—— what stands there

an error message

```
traceback (most recent call last):  
  file "blink.py", line 7, in <module>  
    led.set_color(255, 0, 0)  
attributeerror: no attribute 'set_colour'
```

—— which file, and which line

—— what stands there

—— what does not exist: set_colour with a u

an error message is information, not a verdict.

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

now watch me get it 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 did not take it over. you copied it.

your turn

- 1 one colour, on.
- 2 four colours, in a loop.
- 3 blinking, and you can change how fast.

number 3 is the real one: the speed must change in one place, without digging through the code.