

```
$ cryptography.py
```

Cryptography Challenge

```
// before we can break codes, let's see how data gets encoded in the first place
```

— Dr. Hughes

What Does "Encoding" Mean?

Encoding = writing the *same* information in a *different* way.

plain text

HELP

Morse code

. - . . . - - .

A=1, B=2, C=3 ...

8 5 12 16

binary (what
computers use)

01001000 01000101 01001100 01010000

// one message, four codes. Nothing was lost: you can get HELP back from any of them.

You Already Do It

~~laughing out loud~~



lol

17 → 3 characters

~~on my way~~



omw

9 → 3 characters

~~I agree with that~~



17 → 1 symbol

Compression is encoding with a goal: say the same thing using as few symbols as possible.

Computers Encode Everything

Text, pictures, sound, video: a computer stores all of it as **numbers**.

A



65

a letter



255, 0, 0

a red pixel
(red, green, blue)



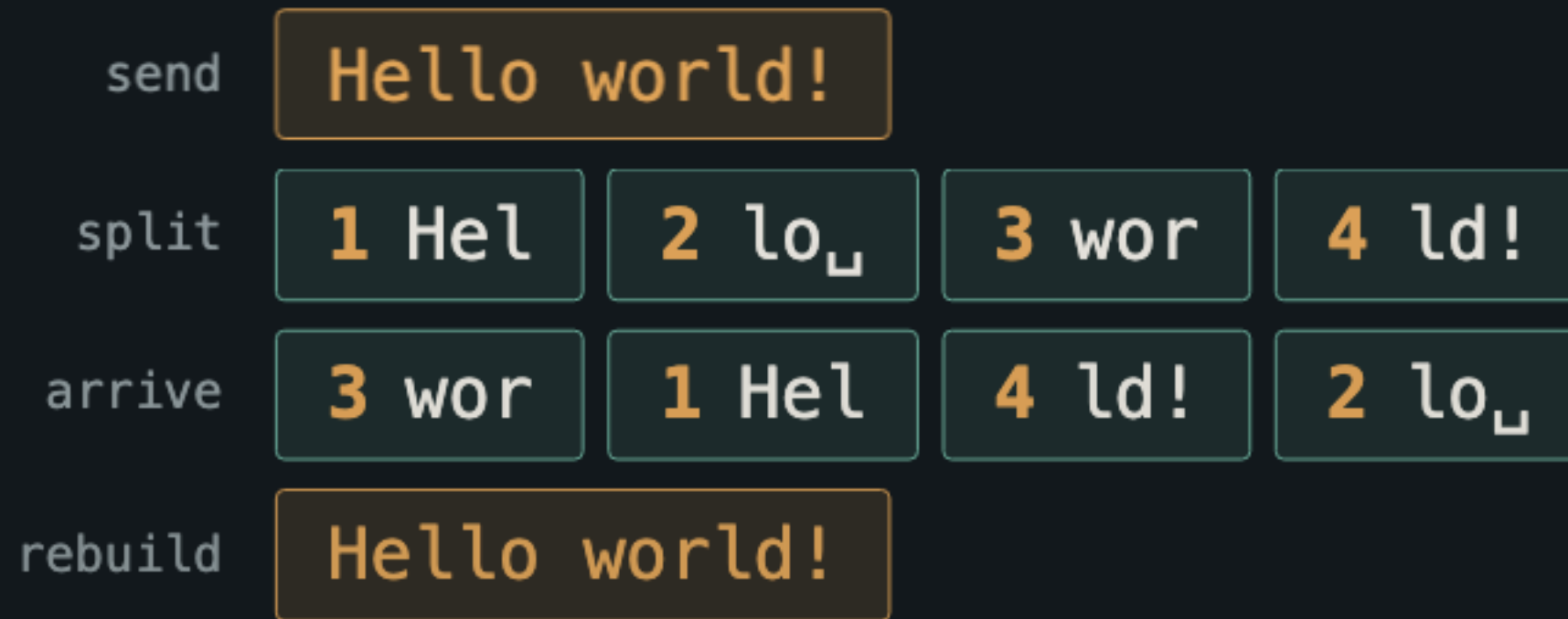
440

a musical note
(wobbles per second)

// different jobs use different codes. Every app and file format is built on them.

What About the Internet?

Data travels in **packets**: small chunks with a number on them.



Packets organise data for the journey. **Compression** shrinks it: the same big idea (find a pattern, use it), with cleverer methods than RLE.

Why Compress Data?

Less data = less storage, faster sending. Here is a real example from today:

Picture A: 46 × 38

pixel by pixel	1,748 values
run-length encoded	182

Picture B: 47 × 47

pixel by pixel	2,209 values
run-length encoded	183

// RLE keeps every detail (it is **lossless**). Some methods, like MP3, throw a little away to save even more.

What Is Run-Length Encoding?

Run-length encoding (RLE) replaces a repeated run of the same value with just two things: **what** is being repeated, and **how many times**.

AAAABBBCCCCD → 4A 3B 4C 1D

A Few Things Worth Knowing

LOSSLESS

AAAABBB → 4A3B → AAAABBB

Decode it and you get the exact original back.

LOVES REPEATS

AAAAAAAA → 8A

8 characters become 2.

HATES NOISE

ABABAB → 1A1B1A1B1A1B

6 become 12. Bigger!

still used in: fax machines

BMP

TIFF

PCX

Compressing a Sentence

RLE isn't just for pictures — it works on any run of repeated symbols, including text.

Y E SSSSS !!!!!

↓ run-length encode ↓

1Y 1E 5S 4!

11 characters → 8. Long runs of the same character compress well.

...But Not Every Sentence Compresses

H A H A H A

↓ run-length encode ↓

1H 1A 1H 1A 1H 1A

6 characters → 12. No repeats means RLE makes it **bigger**, not smaller.

Compressing a Picture

Instead of storing every pixel, store **how many blanks, then how many coloured pixels**: one row at a time.

6, 4 

6, 4 = 6 blanks, then 4 coloured pixels.

// a clue that starts with 0 means the row starts with a coloured pixel.

Decoding messages #1: the clues

2, 7, 3											
1, 10, 1											
1, 8, 2, 1											
1, 8, 2, 1											
1, 10, 1											
2, 6, 4											
3, 4, 5											
4, 3, 5											
0, 12											

// decode it on paper, then press ↓ to check

Decoding messages #1: decoded

2, 7, 3	
1, 10, 1	
1, 8, 2, 1	
1, 8, 2, 1	
1, 10, 1	
2, 6, 4	
3, 4, 5	
4, 3, 5	
0, 12	

Your Turn: By Hand

Pencil at the ready. Shade the grids from the clues.

01 Warm-up rows

02 Decode it: Mystery picture 1

03 Decode it: Mystery pictures 2 and 3

04 Decode it: Mystery picture 4

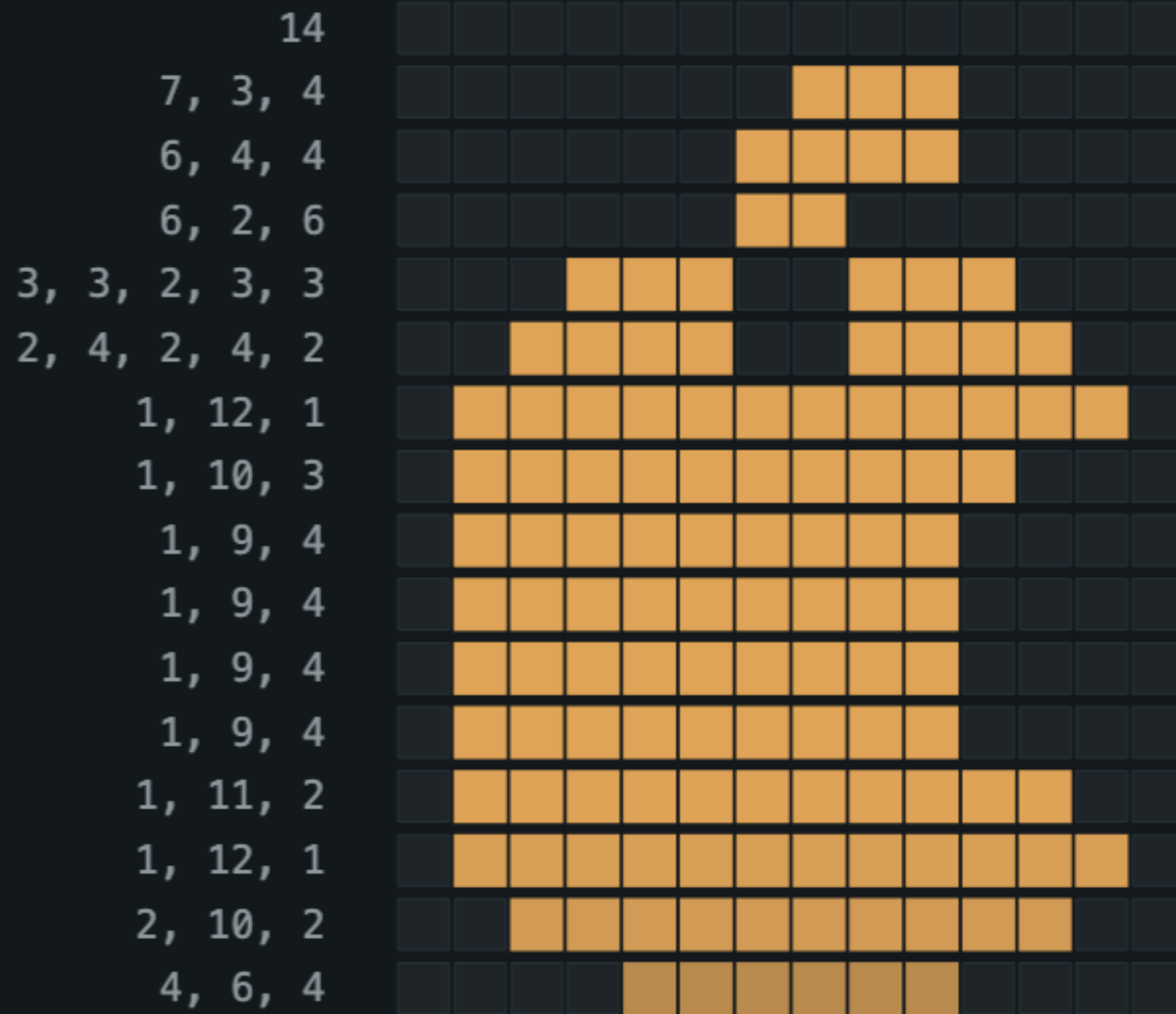
// work out what each picture is. Shade row by row, then check against the next slides (↓ shows the answer).

Decoding messages #2: the clues

	14	
7, 3, 4		
6, 4, 4		
6, 2, 6		
3, 3, 2, 3, 3		
2, 4, 2, 4, 2		
1, 12, 1		
1, 10, 3		
1, 9, 4		
1, 9, 4		
1, 9, 4		
1, 9, 4		
1, 11, 2		
1, 12, 1		
2, 10, 2		
4, 6, 4		

// decode it on paper, then press ↓ to check

Decoding messages #2: decoded

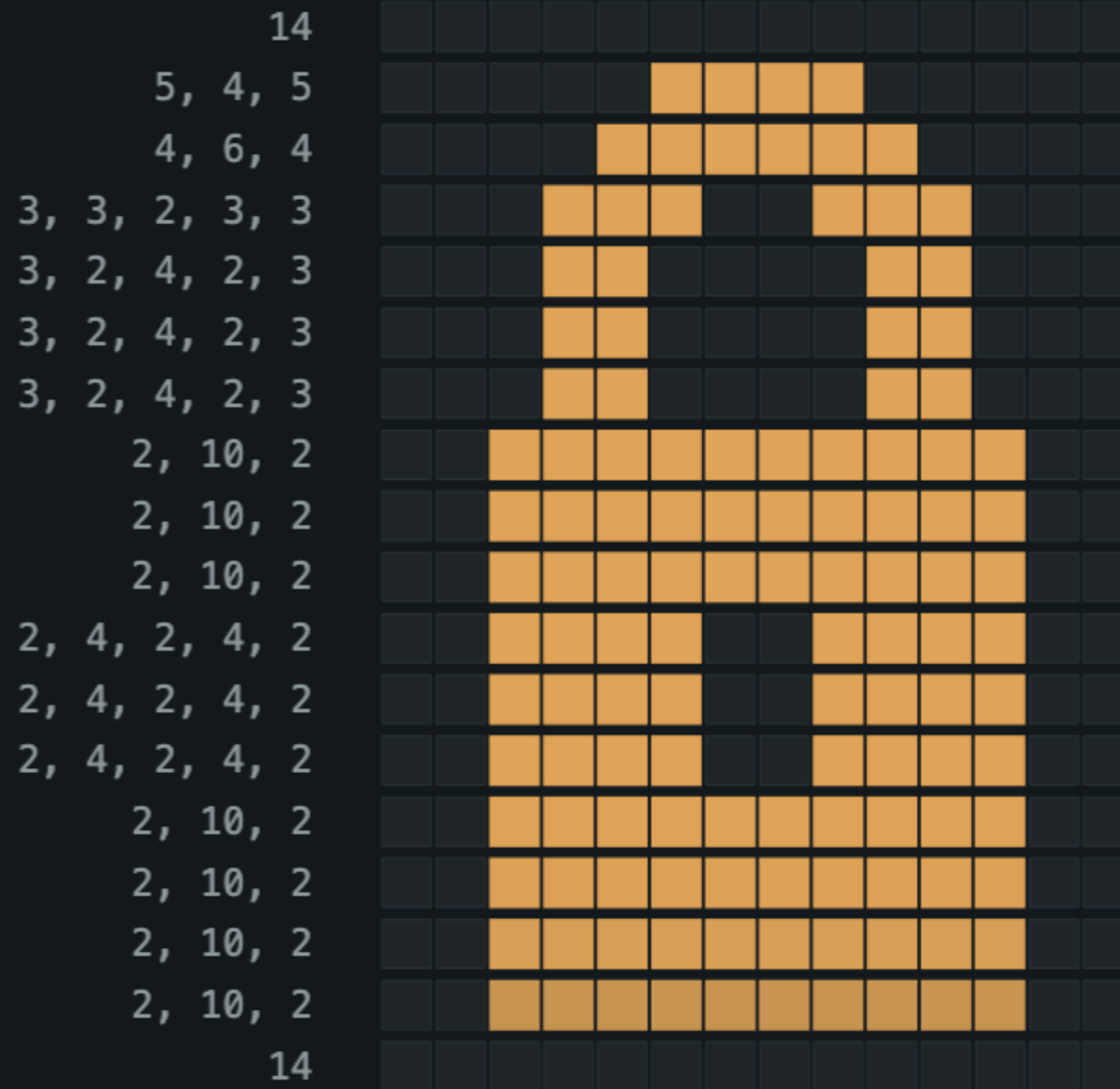


Decoding messages #3: the clues

14	
5, 4, 5	
4, 6, 4	
3, 3, 2, 3, 3	
3, 2, 4, 2, 3	
3, 2, 4, 2, 3	
3, 2, 4, 2, 3	
2, 10, 2	
2, 10, 2	
2, 10, 2	
2, 4, 2, 4, 2	
2, 4, 2, 4, 2	
2, 4, 2, 4, 2	
2, 10, 2	
2, 10, 2	
2, 10, 2	
2, 10, 2	
14	

// decode it on paper, then press ↓ to check

Decoding messages #3: decoded

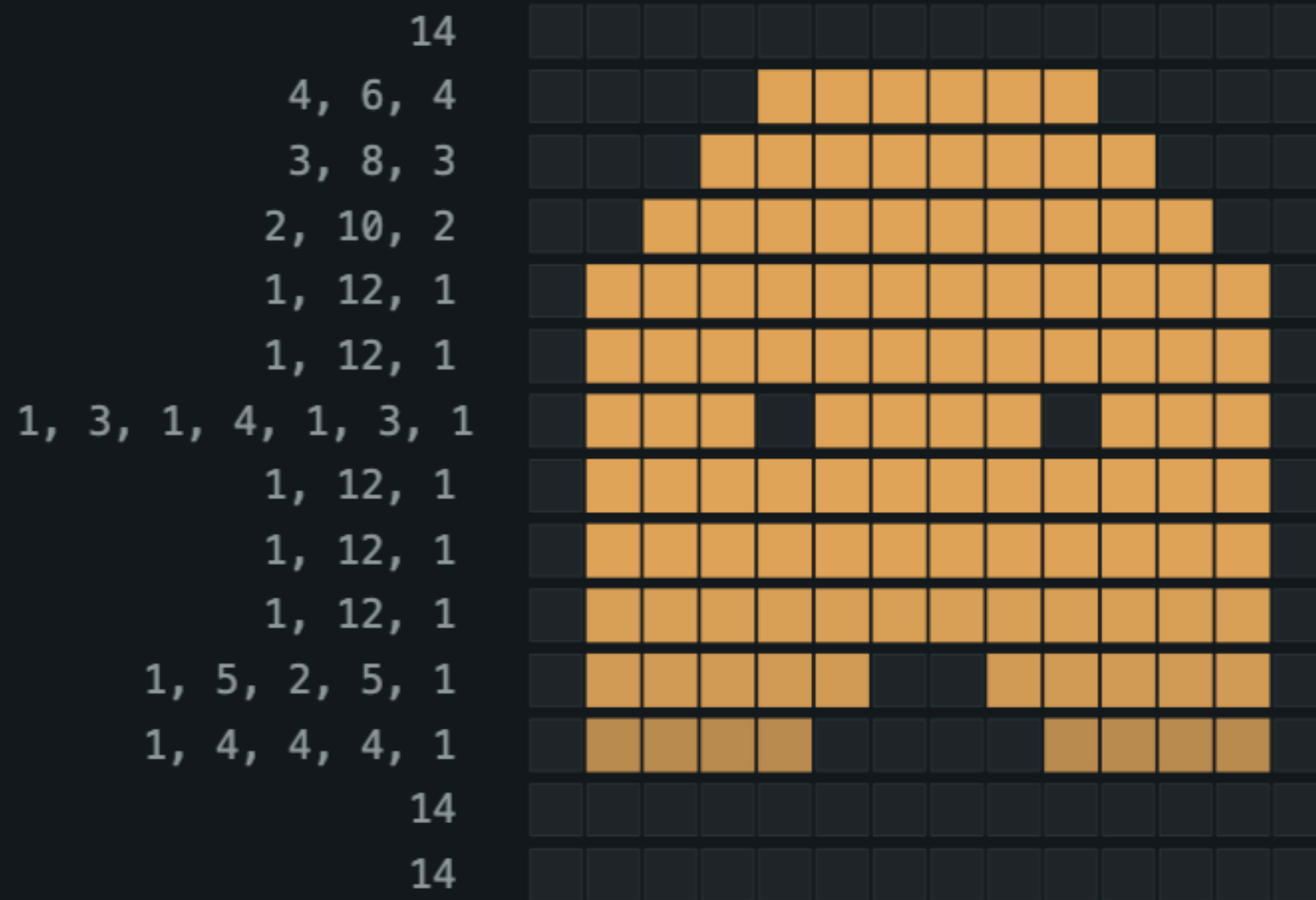


Decoding messages #4: the clues

	14													
	4, 6, 4													
	3, 8, 3													
	2, 10, 2													
	1, 12, 1													
	1, 12, 1													
1, 3, 1, 4, 1, 3, 1														
	1, 12, 1													
	1, 12, 1													
	1, 12, 1													
1, 5, 2, 5, 1														
1, 4, 4, 4, 1														
	14													
	14													

// decode it on paper, then press ↓ to check

Decoding messages #4: decoded



That took a while.

Fun, but slow. And those were the small ones.

How Long Did That Take?

mystery 1: 9 rows

mystery 3: 18 rows

the big one: 46 rows × 38 columns

What if it were a photo with a million rows?

Let the Computer Do It

Computers are great at repeating things.

» You already know how: loops and functions.

A Row Is a List

```
1 row = [0, 0, 1, 1, 1, 0]
```

0 = blank

1 = coloured

0, 0, 1, 1, 1, 0



A Picture Is a List of Rows

```
1 mug = [  
2  [0,0,1,1,1,1,1,1,1,0,0,0],  
3  [0,1,1,1,1,1,1,1,1,1,1,0],  
4  [0,1,1,1,1,1,1,1,1,0,0,1],  
5  ...  
6  ]
```

// a list of lists. Each inner list is one row of the grid

Same Thing, Two Ways

2, 7, 3



Clue: [2, 7, 3]

Row: [0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

» Our job: turn the clue into the row.

A Function We Give You

```
1 def show_picture(picture):  
2     # draws the picture
```

Don't read the inside. Just use it.

// like print(): you use it without knowing how it works

Try It on a Heart

```
1 heart = [  
2  [0, 1, 0, 1, 0],  
3  [1, 1, 1, 1, 1],  
4  [0, 1, 1, 1, 0],  
5  [0, 0, 1, 0, 0],  
6  ]  
7 show_picture(heart)
```

// the picture appears in the Plots pane, just like your graphs

The Heart



1 = dark square, 0 = blank. Check it matches the list.

One Line We Need

```
1 from pictures import MUG, DINOSAUR, MARIO
```

"Fetch the clues from another file."

» `pictures.py` must be in the same folder as your file.

Decode One Row

Clue `[2, 7, 3]` becomes

`[0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]`

What the Computer Does

1. start with an empty row, colour = 0 (blank)
2. take a number: add that many of the colour
3. swap the colour: 0 becomes 1, 1 becomes 0
4. repeat for the next number

Follow

[2, 7, 3]

2 blanks: [0, 0]

7 ones: [0, 0, 1, 1, 1, 1, 1, 1, 1]

3 blanks: [0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

New Trick: `.append`

```
>>> row = []  
>>> row.append(0)  
>>> row.append(0)  
>>> row  
[0, 0]
```

// append = stick one more thing on the end of a list

A Loop Inside a Loop

```
1 for number in [2, 3]:  
  2 for i in range(number):  
    3 print(number)
```

2

2

3

3

3

// the outer loop picks each number. The inner loop repeats that many times.

Your `decode_row`

```
1 def decode_row(clue):
2     row = []
3     colour = 0
4     # for each number in the clue:
5     #     add that many colour: a loop inside a loop, row.append(colour)
6     #     swap colour: if / else (or the shortcut colour = 1 - colour)
7     return row
```

Test It

```
1 print(decode_row([2, 7, 3]))
```

```
[0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]
```

// if yours matches, it works

Now a Whole Picture

A picture is many rows. Decode each row, one by one.

» You already wrote the row decoder. Reuse it!

Your `decode_picture`

```
1 def decode_picture(clues):  
2     picture = []  
3     # for each clue in clues:  
4     #     row = decode_row(clue)  
5     #     picture.append(row)  
6     return picture
```

// write once, reuse: same idea as `add(10, 20)`, `add(100, 1)`

The Big Moment

```
1 show_picture(decode_picture(MUG))
```

MUG: clues

decode_picture: rows

show_picture: draw

Then swap MUG for UK_IRELAND.

Your Turn: Computer Time

- ✓ Task 1: open `rle_starter.py`, run it. See the heart.
- ✓ Task 2: finish `decode_row`. Test on `[2, 7, 3]`.
- ✓ Task 3: finish `decode_picture`. Show the mug.
- ✓ Task 4: try `DINOSAUR`, `MARIO`, then `UK_IRELAND`.

Task 5: `UK_IRELAND` is 46 rows. Roughly how long would that take by hand? By computer?

Stuck? Use the Worksheets

05 Be The Computer

06 Plan Before You Type

07 Fill In The Functions

08 The Starter Code (printed copy)

// printed sheets: work it out on paper first, then type it

Finished Early?

» **Stretch:** write `encode_row(row)` and go backwards: row in, clue out.

Off the computer: **12 Design Your Own** and **13 Encode It**. Or race the computer by hand on the big pictures: **09, 10, 11**.

The Finished Product

Same code. Much bigger pictures.

```
1 show_picture(decode_picture(DINOSAUR))  
2 show_picture(decode_picture(MARIO))  
3 show_picture(decode_picture(UK_IRELAND))
```

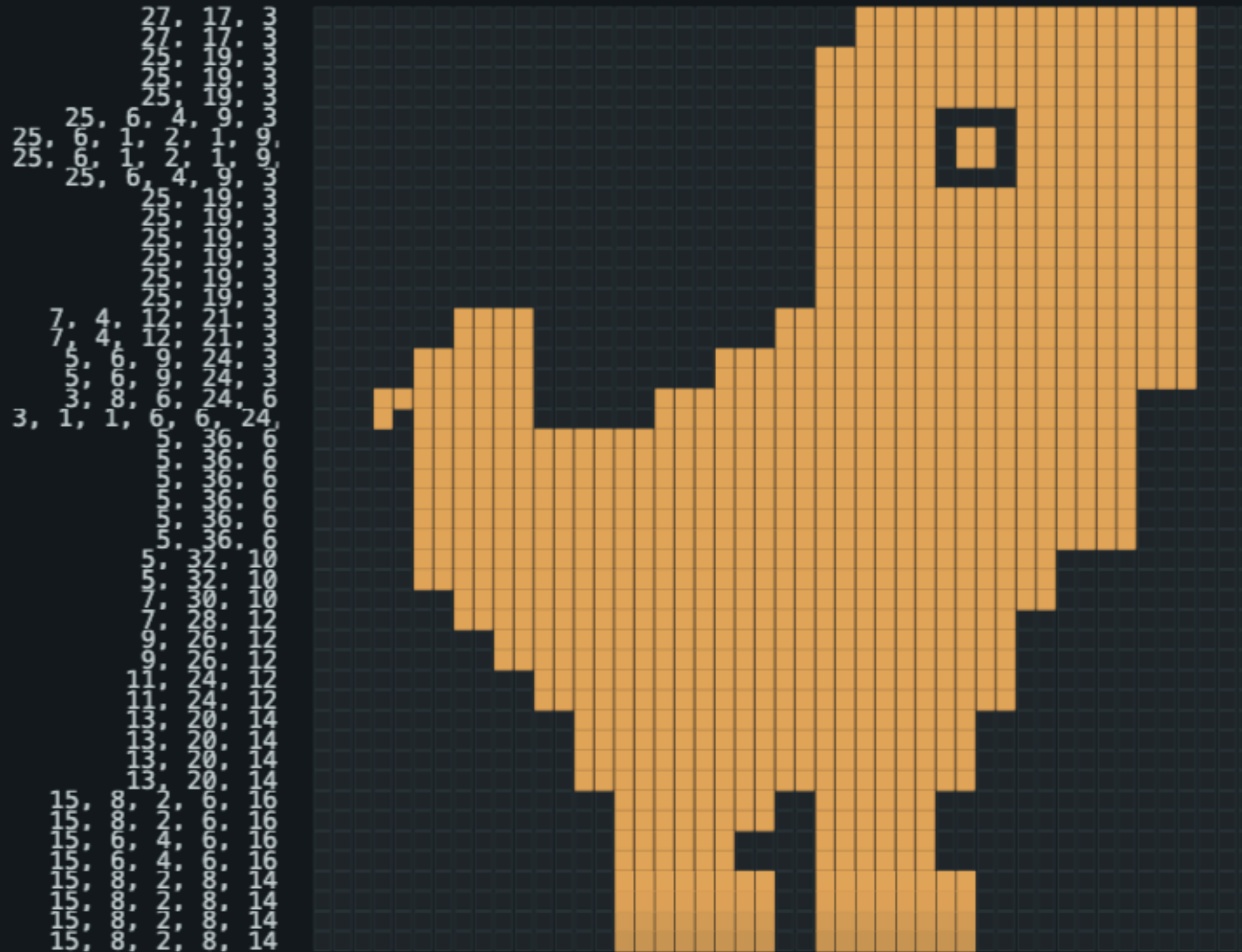
// the next slides show what your plots should look like (↓ reveals each picture)

Decoding messages #5: the clues

25,	25,	6,	27,	17,	3
25,	6,	1,	27,	17,	3
25,	6,	1,	25,	19,	3
	25,	6,	25,	19,	3
		1,	25,	19,	3
		6,	4,	9,	3
		2,	2,	1,	9
		2,	4,	1,	9
		4,	25,	19,	3
			25,	19,	3
			25,	19,	3
			25,	19,	3
			25,	19,	3
			25,	19,	3
	7,	4,	12,	21,	3
	7,	4,	12,	21,	3
	5,	6,	9,	24,	3
	5,	6,	9,	24,	3
	3,	8,	6,	24,	6
3,	1,	1,	6,	6,	24
			5,	36,	6
			5,	36,	6
			5,	36,	6
			5,	36,	6
			5,	36,	6
			5,	36,	6
			5,	32,	10
			5,	32,	10
			7,	30,	10
			7,	28,	12
			9,	26,	12
			9,	26,	12
		11,	24,	12	12
		11,	24,	12	12
		13,	20,	14	14
		13,	20,	14	14
		13,	20,	14	14
		13,	20,	14	14
15,	8,	2,	6,	16	16
15,	8,	2,	6,	16	16
15,	6,	4,	6,	16	16
15,	6,	4,	6,	16	16
15,	8,	2,	8,	14	14
15,	8,	2,	8,	14	14
15,	8,	2,	8,	14	14
15,	8,	2,	8,	14	14

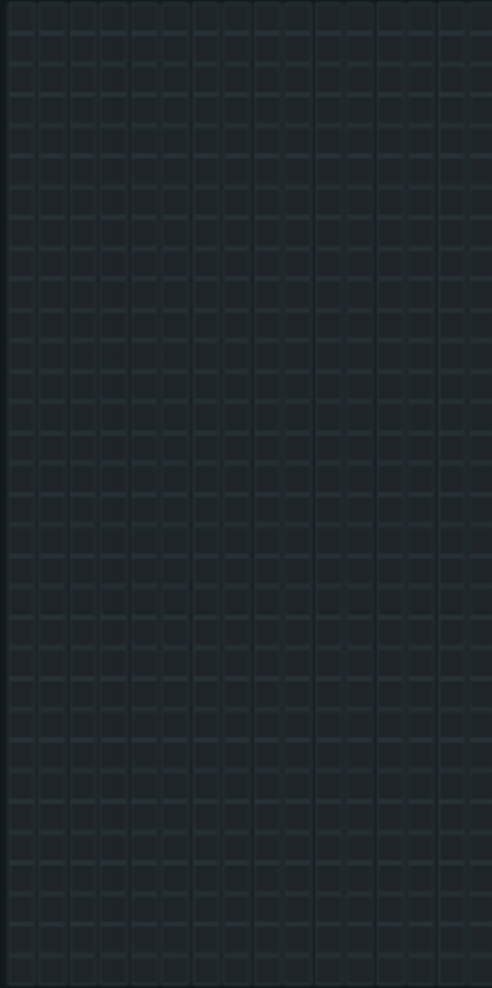
```
// press ↓ to see what your code draws
```

Decoding messages #5: decoded



Decoding messages #6: the clues

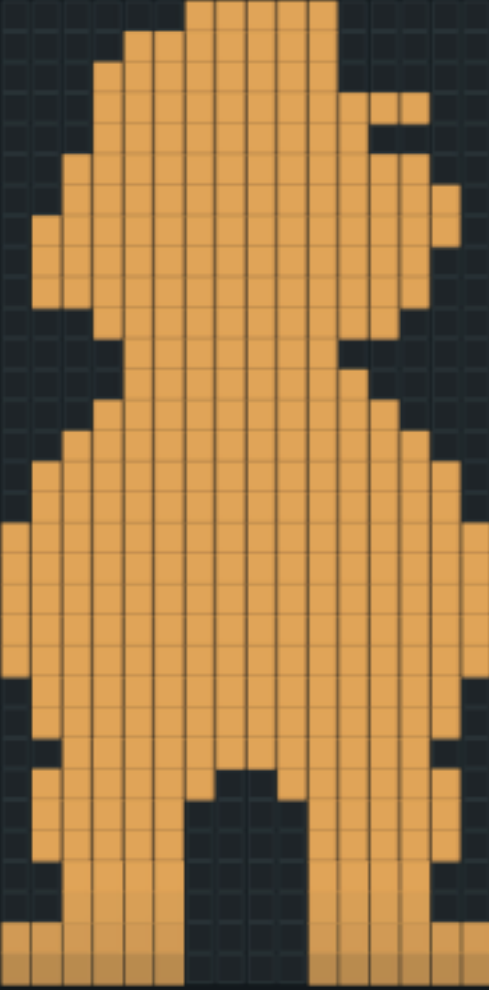
```
6. 5. 5
4. 7. 5
3. 8. 5
3. 11. 2
3. 9. 4
2. 12. 2
2. 13. 1
1. 14. 1
1. 13. 2
1. 13. 2
3. 10. 3
4. 7. 5
4. 8. 4
3. 10. 3
2. 12. 2
1. 14. 1
1. 14. 1
0. 16
0. 16
0. 16
0. 16
0. 16
1. 14. 1
1. 14. 1
2. 12. 2
1. 1
1. 1
1. 1
2. 1
2. 1
0. 2
0. 2
```



// press ↓ to see what your code draws

Decoding messages #6: decoded

			6.	5.	5
			4.	7.	5
			3.	8.	5
		3.	3.	11.	2
		3.	3.	9.	4
		2.	2.	12.	2
		2.	1.	13.	1
		1.	1.	14.	1
		1.	1.	13.	2
		3.	3.	10.	3
		4.	4.	7.	5
		4.	4.	8.	4
		3.	2.	10.	3
		2.	1.	12.	2
		1.	1.	14.	1
		1.	1.	14.	1
			0.	16	
			0.	16	
			0.	16	
			0.	16	
			0.	16	
		1.	14.	1	
		1.	14.	1	
		2.	12.	2	
			6.	1	
			4.	5.	
			4.	5.	
			4.	4.	
			6.	4.	
			6.	4.	
1.	6.				6
1.	5.				
1.	4.				
2.	4.				
2.	0.				
2.	0.				

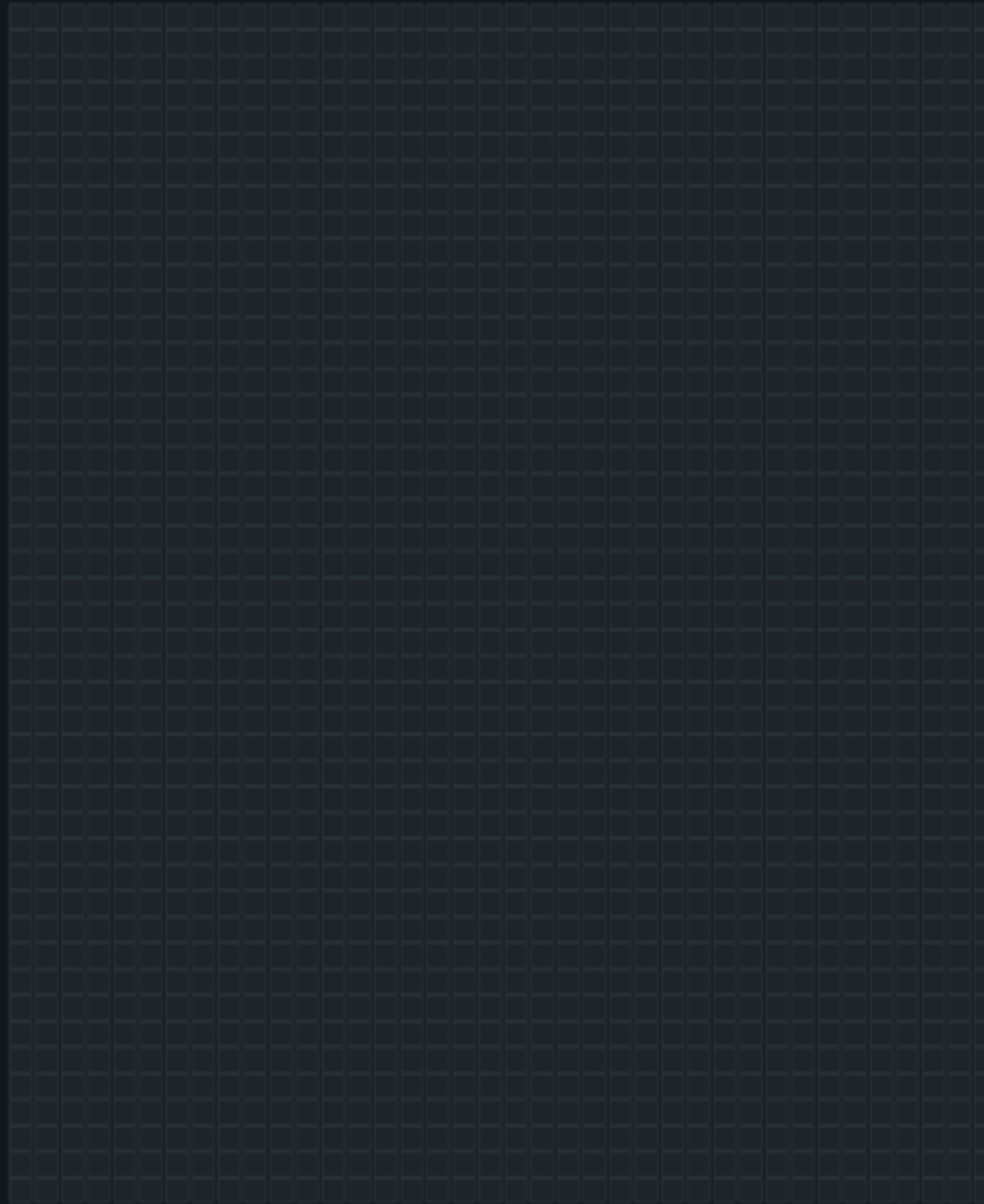


The Big One: the clues

```

17, 6, 15, 38
16, 6, 16, 38
16, 5, 17, 38
15, 5, 18, 38
15, 1, 12, 38
15, 11, 12, 38
15, 11, 12, 38
14, 11, 13, 38
14, 11, 13, 38
16, 8, 14, 38
15, 9, 14, 38
15, 9, 14, 38
15, 1, 1, 12, 38
15, 1, 1, 10, 11, 38
15, 1, 2, 9, 11, 38
9, 2, 1, 2, 3, 10, 38
7, 7, 3, 11, 10, 38
7, 8, 2, 11, 10, 38
6, 10, 5, 8, 9, 38
7, 9, 5, 9, 8, 7, 38
2, 13, 7, 9, 8, 6, 38
3, 11, 9, 9, 6, 38
3, 11, 9, 9, 6, 38
2, 12, 8, 10, 5, 5, 38
4, 10, 5, 14, 6, 38
4, 10, 4, 14, 6, 38
4, 10, 5, 17, 2, 38
3, 11, 6, 17, 1, 38
3, 10, 6, 18, 1, 38
2, 11, 6, 17, 2, 38
2, 8, 7, 19, 2, 38
2, 6, 8, 19, 3, 38
3, 3, 13, 15, 4, 38
    21, 14, 3, 38
    19, 17, 2, 38
    18, 17, 3, 38
18, 10, 1, 4, 5, 38
17, 5, 2, 1, 13, 38
    16, 5, 17, 38
    15, 2, 21, 38
    38
    38

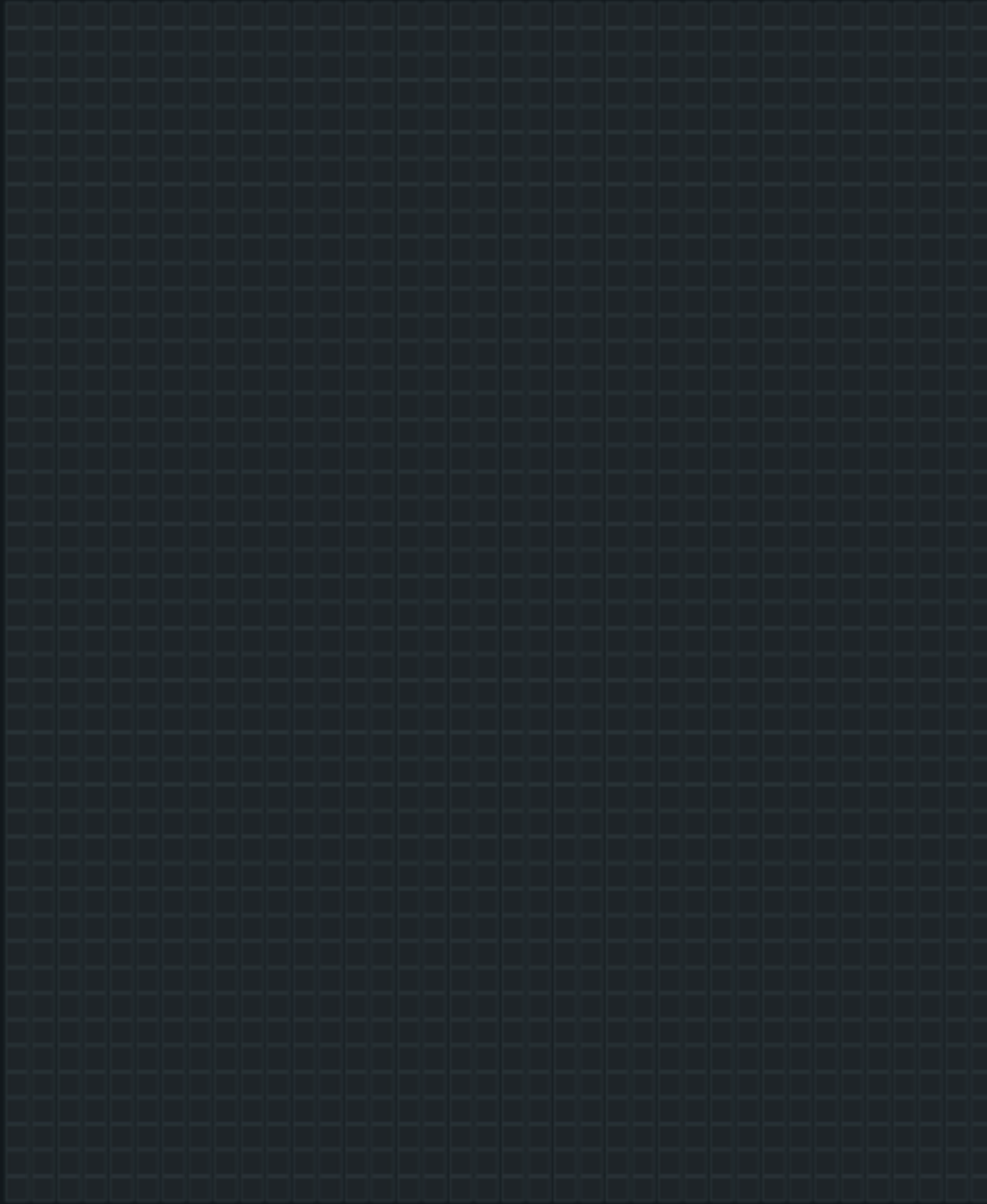
```



// press ↓ for the decoded picture

The Big One: clues and decoded

38
38
15
16
17
18
12
12
12
12
13
13
14
14
14
12
11
11
10
10
10
10
9
10
9
8
7
7
6
7
2
3
3
3
2
4
4
4
3
3
2
2
2
3
18
17
15
17
6
6
5
5
5
11
11
11
8
9
9
10
9
7
8
9
9
9
10
14
17
18
19
15
14
17
1
2
3
4
3
2
5
13
17
21
38
38



What You Learned Today

- ✓ encoding = writing information a different way
- ✓ compression = encoding to use fewer symbols
- ✓ run-length encoding: store what repeats, and how many times
- ✓ it works great on repetitive data, badly on noisy data
- ✓ you decoded small pictures by hand
- ✓ you got Python to decode a dinosaur, Mario, and the UK & Ireland

You Can Read Code Now

Every file on your phone or laptop is just numbers, encoded — you just decoded some by hand.

» Challenge: design your own picture on worksheet 12 as an RLE clue for a partner to decode. Stretch: write `encode_row`.