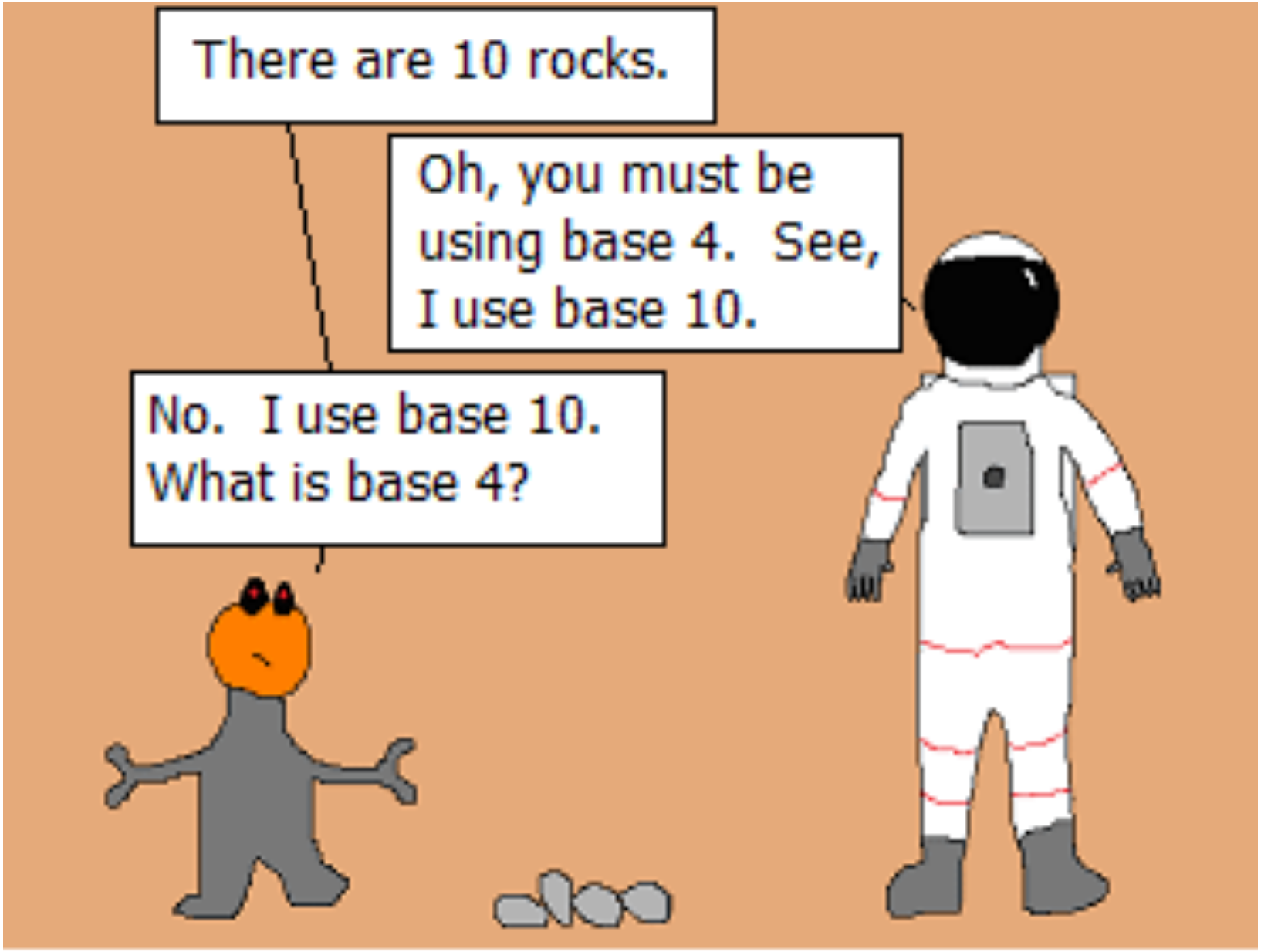


Every base is base 10.

Binary

- Computers represent everything as bits
- Recall: a byte is 8 bits
- Int: 4 bytes (32 bits)

Binary

- Computers represent everything as bits
- Recall: a byte is 8 bits
- Int: 4 bytes (32 bits)
- What's the largest int we can represent?

Binary

- Computers represent everything as bits
- Recall: a byte is 8 bits
- Int: 4 bytes (32 bits)
- What's the largest int we can represent?

2^32 - 1

(unsigned)

Hexadecimal (base 16)

- Binary takes up a lot of space
- Hexadecimal takes few digits but can easily be converted to binary (and vice versa)
  - Hex uses digits 0-9 and a-f
  - 1 hex digit = 4 bits
- 0000 0000 0000 0001 1101 0011 0101 1011
- 1d35b

In C

- Format ints
  - %d for decimal
  - %b for binary
  - %x for hex
- Assign ints
  - 0b for binary (ex: 0b11011 is 27)
  - 0x for hex (ex: 0x83fa9 is 540585)

Bitwise Operators

- You know logical operators…&&,||,!
- We will now learn &,|,~,^,<<,>>
- These operate at the bit level

&

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

|

a	b	a | b
1	1	1
1	0	1
0	1	1
0	0	0

^

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

^

a	~a
1	0
0	1

Operators on multiple bits

AND

```
  0110
& 1100
  ----
  0100
```

OR

```
  0110
| 1100
  ----
  1110
```

XOR

```
  0110
^ 1100
  ----
  1010
```

NOT

```
  
~ 1100
  ----
  0011
```

Bitmasks

- We often want to manipulate or isolate specific bits from a collection
- A **bitmask** is a bit pattern that achieves this
- We can use and/or create bitmasks using bitwise operators

Example: CSCI courses

- Array of ints vs. storing bits

Example: CSCI courses

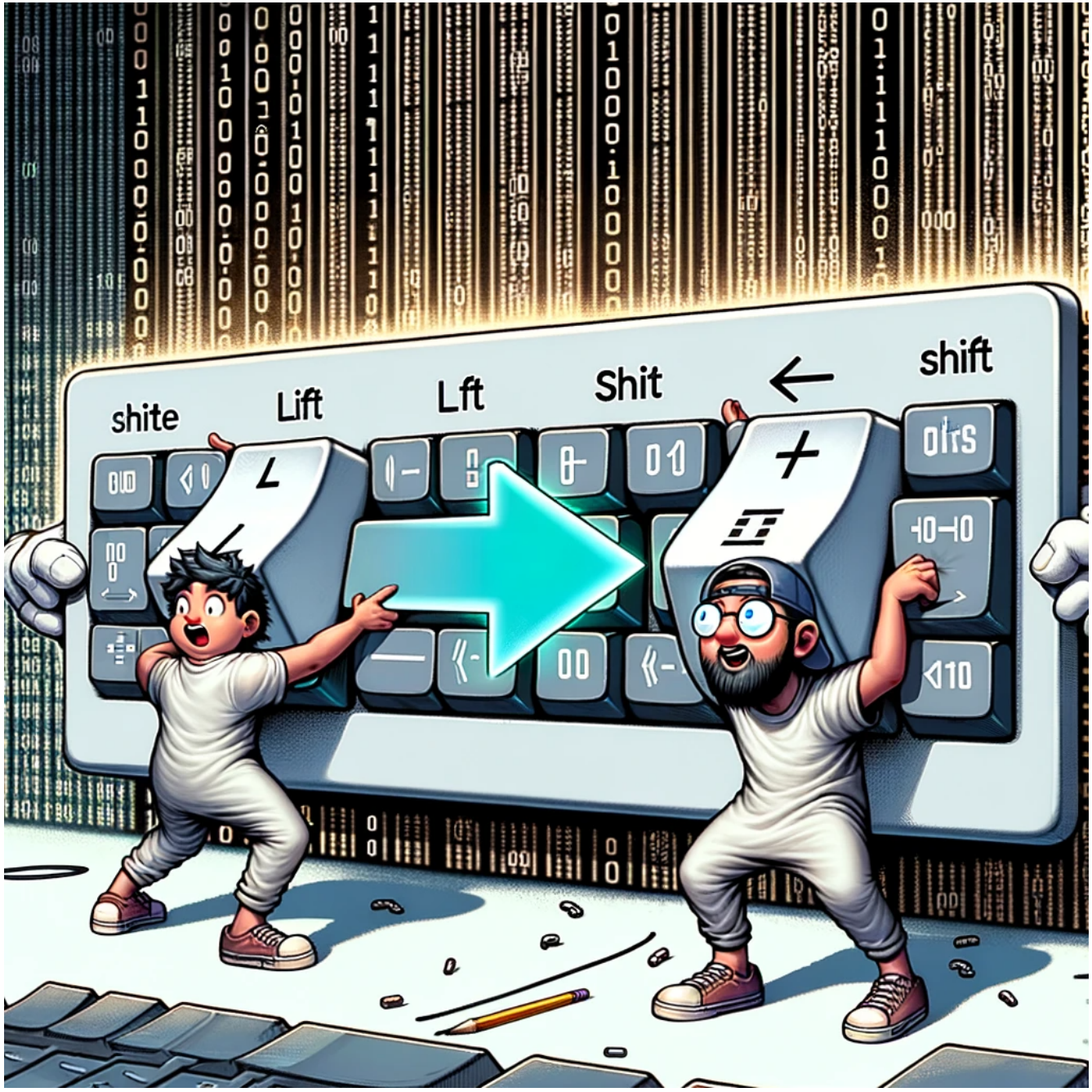
- Array of ints vs. storing bits
- Bitmasks
  - Setting bits to 1 with |
  - Setting bits to 0 with &
  - Computing union and intersection
  - ”Masking off” unwanted bits
- But how do we mask an arbitrary position?

<< and >>

- x << k shifts x left by k

00**110111** << 2 results in **11011**100
0110**0011** << 4 results in **0011**0000
1001**0101** << 4 results in **0101**0000

- x >> k shifts x right by k
- Careful with unsigned ints for >>


shite
Lift
Lft
Shit
shift