ANSWERS

F8.2 Binary and denary

Data representation · GCSE · OCR J277 1.2.4, AQA 8525 3.3.1, Edexcel 1CP2 2.1.3 · about 15 min

What this lesson is about

Converting 8-bit numbers both ways, and hearing them on the buzzer.

- Place values
- Binary to denary in Python
- Denary to binary
- Hear a number

Questions

6 marks in all

1. What is 01001101 in denary? [1 mark]

> **Answer:** 77. 64 + 8 + 4 + 1.

2. Write 200 as an 8-bit binary number. [1 mark]

> **Answer:** 11001000. 128 + 64 + 8 = 200.

3. What is the largest number 8 bits can hold? [1 mark]

> **Answer:** 255. 11111111 is 255.

4. Write 5 as an 8-bit binary number. [1 mark]

> **Answer:** 00000101. 4 + 1, with leading zeros to make 8 bits.

5. What does this program print? [1 mark]

```
total = 0
value = 1
for bit in reversed("1010"):
    if bit == "1":
        total = total + value
    value = value * 2
print(total)
```

> **Answer:**
>
> ```
> 10
> ```
>
> 1010 is 8 + 2 = 10.

6. What is the place value of the leftmost bit in an 8-bit binary number? [1 mark]

- ☐ **A** 128
- ☐ **B** 256
- ☐ **C** 100
- ☐ **D** 8

> **Answer:** A. The places are 128, 64, 32, 16, 8, 4, 2, 1.

The task: play in binary

Write `to_binary(n)` yourself, without `bin`, `format` or `int(..., 2)`, that returns an 8-character string of 1s and 0s. Use it to print the binary of 77, 5 and 200, one per line, and then play the bits of 77: a note of 880 Hz for each 1 and 440 Hz for each 0, each for 0.2 seconds.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

def to_binary(n):
    return ""
```

The hint students can ask for: Work down the place values from the largest. If the number is at least the place value, the bit is a 1 and you take that value off; otherwise it is a 0. Then sound a high or low note for each bit in turn.

A solution

```
from bugbot import *
connect()
def to_binary(n):
    bits = ""
    for value in [128, 64, 32, 16, 8, 4, 2, 1]:
        if n >= value:
            bits = bits + "1"
            n = n - value
        else:
            bits = bits + "0"
    return bits
for n in [77, 5, 200]:
    print(to_binary(n))
for bit in to_binary(77):
    if bit == "1":
        tone(880, 0.2)
    else:
        tone(440, 0.2)
    wait(0.1)
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.