WORKSHEET

A7.4 Errors, range and precision

Data representation · A level · OCR H446 1.4.1, AQA 7517 4.5.4.5, Eduqas A500QS 2.3 · about 30 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Rounding errors, absolute and relative error, range against precision, overflow and underflow, with a real sensor reading.

- Rounding errors
- Absolute and relative errors
- Range and precision
- Overflow and underflow
- A real reading

Questions

6 marks in all

1. A true value of 2.7 is stored as 2.625. What is the absolute error? [1 mark]

2. A true value of 2.7 is stored as 2.625. What is the relative error as a percentage, to 2 decimal places? [1 mark]

3. A floating point format keeps 12 bits in total. The mantissa is given more bits and the exponent fewer. What changes? [1 mark]

- ☐ **A** Precision increases and range decreases
- ☐ **B** Range increases and precision decreases
- ☐ **C** Both increase
- ☐ **D** Neither changes

4. What is underflow? [1 mark]

- ☐ **A** A result too close to zero for the format to represent
- ☐ **B** A result too large for the format
- ☐ **C** A negative result from adding two positive numbers
- ☐ **D** A loss of the carry bit

5. Why can 0.1 not be stored exactly in binary floating point? [1 mark]

- ☐ **A** Its binary fraction repeats for ever, so it must be cut off
- ☐ **B** It is an irrational number
- ☐ **C** It is too small for the exponent
- ☐ **D** Binary cannot store fractions

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

```
a = 0.1 + 0.2
print(a == 0.3)
print(abs(a - 0.3) < 1e-9)
```

The task: store a reading

The robot faces a wall. Read `distance()` once (it returns centimetres) and convert it to metres. Store the metres in a byte of unsigned fixed point with all 8 bits after the point: the stored whole number is `int(metres * 256)`, which truncates. Then print exactly these five lines, calculating every value: - `reading: <metres> m`, the metres as Python prints the float - `stored: <bits>`, the stored whole number as 8 binary digits (you may use `format(q, "08b")`) - `stored value: <value> m`, the stored whole number divided by 256 - `absolute error: <error> m`, the absolute error, then `round(error, 7)` - `relative error: <percent>%`, the relative error as a percentage, then `round(percent, 2)`, with no space before the `%`

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

metres = distance() / 100
print("reading:", metres, "m")
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a7-4-errors-range-and-precision/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Round instead of truncating. Does the absolute error go down for this reading? Is that true for every reading?
2. Use 16 bits with 8 after the point. What are the new range and precision?
3. Find the largest reading this byte format can hold, and what happens to a reading of 1.2 m.