ANSWERS

F12.6 The digital divide and accessibility

Technology and society · GCSE · OCR J277 1.6.1, AQA 8525 3.8, Edexcel 1CP2 5.1.1 · about 20 min

What this lesson is about

Who is shut out of technology, designing for disability, and measuring colour contrast.

- The digital divide
- Accessibility
- Measuring contrast

Questions

5 marks in all

1. What is the digital divide? [1 mark]

- ☐ **A** The gap between those with good access to technology and those without
- ☐ **B** The gap between fast and slow computers
- ☐ **C** The difference between hardware and software
- ☐ **D** The split between two networks

> **Answer:** A. Income, geography, age and disability all create it.

2. Why does the digital divide matter more now than 20 years ago? [1 mark]

- ☐ **A** Essential services such as banking and homework assume internet access
- ☐ **B** Computers are cheaper
- ☐ **C** Screens are bigger
- ☐ **D** There are more games

> **Answer:** A. Being offline now means missing services, not just entertainment.

3. Why is "errors are shown in red" a problem? [1 mark]

- ☐ **A** Someone who is colour blind may not see the difference
- ☐ **B** Red is hard to print
- ☐ **C** Red text is slower to load
- ☐ **D** It uses more power

> **Answer:** A. Never use colour alone to carry meaning: add a label or an icon.

4. Which helps a screen reader user most? [1 mark]

- ☐ **A** Text alternatives for images
- ☐ **B** Brighter colours
- ☐ **C** More animation
- ☐ **D** A larger mouse pointer

> **Answer:** A. A screen reader reads text, so images need text alternatives.

5. The accepted minimum contrast ratio for normal text is: [1 mark]

- ☐ **A** 4.5
- ☐ **B** 1.0
- ☐ **C** 21
- ☐ **D** 0.5

> **Answer:** A. Large text may go as low as 3.0.

The task: check the contrast

Using the provided `luminance`, write `contrast(a, b)` returning the contrast ratio of two colours. For each `(name, foreground, background)` in `pairs`, print `<name>: <ratio> pass` if the ratio is 4.5 or more, or `<name>: <ratio> FAIL` if not, with the ratio rounded to 1 decimal place. At the end print `failed: <n>`, and set the LED to the foreground colour of the first pair that passes.

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

def luminance(rgb):
    """How bright a colour looks, from 0 (black) to 1 (white)."""
    out = []
    for value in rgb:
        v = value / 255
        out.append(v / 12.92 if v <= 0.03928 else ((v + 0.055) / 1.055) ** 2.4)
    return 0.2126 * out[0] + 0.7152 * out[1] + 0.0722 * out[2]

# name, foreground, background
pairs = [
    ("grey on white", (150, 150, 150), (255, 255, 255)),
    ("navy on white", (0, 40, 120), (255, 255, 255)),
    ("yellow on white", (255, 220, 0), (255, 255, 255)),
    ("white on navy", (255, 255, 255), (0, 40, 120)),
]
```

The hint students can ask for: The ratio is the lighter luminance plus a small offset, divided by the darker one plus the same offset. Compare it with the threshold to decide pass or fail, count the fails, and remember the first foreground that passed for the LED.

A solution

```
from bugbot import *
connect()
def luminance(rgb):
    """How bright a colour looks, from 0 (black) to 1 (white)."""
    out = []
    for value in rgb:
        v = value / 255
        out.append(v / 12.92 if v <= 0.03928 else ((v + 0.055) / 1.055) ** 2.4)
    return 0.2126 * out[0] + 0.7152 * out[1] + 0.0722 * out[2]

pairs = [
    ("grey on white", (150, 150, 150), (255, 255, 255)),
    ("navy on white", (0, 40, 120), (255, 255, 255)),
    ("yellow on white", (255, 220, 0), (255, 255, 255)),
    ("white on navy", (255, 255, 255), (0, 40, 120)),
]

def contrast(a, b):
    la, lb = luminance(a), luminance(b)
    lighter, darker = max(la, lb), min(la, lb)
    return (lighter + 0.05) / (darker + 0.05)

failed = 0
best = None
```

```
for name, fg, bg in pairs:
    ratio = contrast(fg, bg)
    if ratio >= 4.5:
        print(f"{name}: {ratio:.1f} pass")
        if best is None:
            best = fg
    else:
        failed = failed + 1
        print(f"{name}: {ratio:.1f} FAIL")
print("failed:", failed)
if best:
    led(*best)
```

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