ANSWERS

F8.6 Characters: ASCII and Unicode

Data representation · GCSE · OCR J277 1.2.4, AQA 8525 3.3.5, Edexcel 1CP2 2.2.1 · about 12 min

What this lesson is about

Character sets, bits per character, and why Unicode was needed.

- ASCII
- The trouble with ASCII
- Unicode
- Storage: more characters cost more bits
- Working out text size

Questions

5 marks in all

1. How many characters can 7-bit ASCII represent? [1 mark]

> **Answer:** 128. 2 to the power 7.

2. Why was Unicode needed? [1 mark]

- ☐ **A** ASCII has no room for the world's other writing systems and symbols
- ☐ **B** ASCII was too slow
- ☐ **C** Unicode uses fewer bits
- ☐ **D** ASCII cannot store digits

> **Answer:** A. Unicode gives a code to characters from every language, and emoji.

3. ord('A') is 65. What is ord('D')? [1 mark]

> **Answer:** 68. Codes for letters are in order.

4. A 12-character message is stored in 7-bit ASCII. How many bits is it? [1 mark]

> **Answer:** 84. 12 × 7.

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

```
print(len("robot".encode("utf-8")), len("机器人".encode("utf-8")))
```

> **Answer:**
>
> ```
> 5 9
> ```
>
> ASCII letters take 1 byte each in UTF-8; these Chinese characters take 3 each.

The task: message sizes

For the message `BugBot says hi`, print three lines: `characters: <n>`, `ascii bits: <n>` for 7-bit ASCII, and `utf-8 bytes: <n>`. Then print the 7-bit binary code of each of its first three characters, one per line, such as `B 1000010`. Work everything out from the message.

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

message = "BugBot says hi"
```

The hint students can ask for: Count the characters for the first line, multiply by the bits per character for the second, and ask the encoded form for its length for the third. Then take each of the first three characters, find its code number and show it as seven binary digits.

A solution

```
from bugbot import *
connect()
message = "BugBot says hi"
print("characters:", len(message))
print("ascii bits:", len(message) * 7)
print("utf-8 bytes:", len(message.encode("utf-8")))
for ch in message[:3]:
    print(ch, format(ord(ch), "07b"))
```

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