ANSWERS

F11.5 Passwords and authentication

Cyber security · GCSE · OCR J277 1.4.2, AQA 8525 3.6.3, Edexcel 1CP2 6.4.4 · about 15 min

What this lesson is about

Ways to authenticate, strong passwords, 2FA, CAPTCHA and hashing.

- Ways to prove who you are
- What makes a password strong?
- Storing passwords safely
- Locking the robot

Questions

5 marks in all

1. Which makes a password stronger? [1 mark]

- ☐ **A** Making it longer and more varied
- ☐ **B** Using a real word
- ☐ **C** Using your birthday
- ☐ **D** Reusing it everywhere

> **Answer:** A. Length and variety multiply the guesses needed.

2. What is two-factor authentication? [1 mark]

- ☐ **A** Using two different kinds of proof, such as a password and a phone code
- ☐ **B** Two passwords
- ☐ **C** A longer password
- ☐ **D** Logging in twice

> **Answer:** A. A stolen password alone is then not enough.

3. What is a CAPTCHA for? [1 mark]

- ☐ **A** Proving you are a human, not an automated program
- ☐ **B** Encrypting your password
- ☐ **C** Storing your password safely
- ☐ **D** Speeding up login

> **Answer:** A. It helps stop automated attacks such as brute force.

4. Why should passwords be stored as a hash, not plain text? [1 mark]

- ☐ **A** If the file leaks, the real passwords are not exposed
- ☐ **B** It makes login faster
- ☐ **C** Hashes are shorter
- ☐ **D** It is required by law

> **Answer:** A. A hash cannot easily be turned back into the password.

5. Why do systems lock you out after a few wrong tries? [1 mark]

- ☐ **A** To defeat brute force guessing
- ☐ **B** To save electricity
- ☐ **C** To back up your data
- ☐ **D** To speed up the server

> **Answer:** A. It stops a program trying thousands of guesses.

The task: a password checker

Write `strength(password)` scoring one point for each rule met: at least 8 characters; a lower-case letter; an upper-case letter; a digit; a symbol (a character that is not a letter or digit). For each password in `passwords`, print `<password>: <score>/5 <verdict>`, where the verdict is `weak` for 0 to 2, `ok` for 3, and `strong` for 4 or 5. At the end print `strong passwords: <n>`.

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

passwords = ["robot", "Sunshine", "Bugbot42", "x9$Lq2!vT", "12345678"]
```

The hint students can ask for: Score one point for each rule the password meets: long enough, a lower-case letter, an upper-case letter, a digit, and a character that is neither a letter nor a digit. The verdict comes from bands on that score.

A solution

```
from bugbot import *
connect()
passwords = ["robot", "Sunshine", "Bugbot42", "x9$Lq2!vT", "12345678"]

def strength(password):
    score = 0
    if len(password) >= 8: score = score + 1
    if any(c.islower() for c in password): score = score + 1
    if any(c.isupper() for c in password): score = score + 1
    if any(c.isdigit() for c in password): score = score + 1
    if any(not c.isalnum() for c in password): score = score + 1
    return score

strong = 0
for password in passwords:
    s = strength(password)
    verdict = "strong" if s >= 4 else "ok" if s == 3 else "weak"
    if s >= 4:
        strong = strong + 1
    print(f"{password}: {s}/5 {verdict}")
print("strong passwords:", strong)
```

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