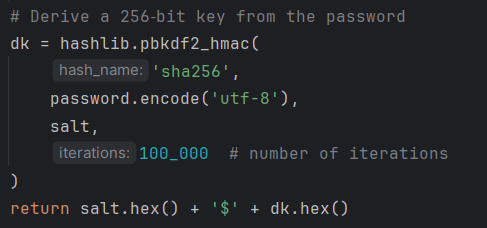
**COMP3028 Coursework 3 Group 12 (Task 1)**

Design choices and explanation

* Hashing algorithm: We chose to use SHA-256 with PBKDF2-HMAC. PBKDF is a popular and NIST-approved key derivation function that purposefully slows down hashing via a large number repetitions (in this case, 100000), making brute-force attacks far more costly.

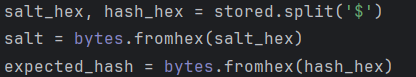


* Salt: For every password, a distinct 128-bit salt (os.urandom(16)) is created. Precomputed “rainbow tables’ are defeated when the salt is stored with the hash, preventing two users with the same password from having identical hashes.



* Storage format: The salt and hash are concatenated in hex, with a $ between them. We separate them, re-deduce the hash from the user’s input, and compare upon login.





* Verification: For a timing-attack resistant comparison, we utilize hmac.compare\_digest, which prevents an attacker from learning how long the check takes.

