Question-2

This Code basically performs three primary tasks:

- **Decoding and Verifying a JWT**: It extracts and verifies a given JSON Web Token (JWT) using HMAC-based signatures (HS256 or HS512).
- **Brute-force Secret Discovery**: It attempts to determine the secret key used to sign the JWT by testing common 5-character lowercase-alphanumeric secrets.
- **Generating a New JWT**: Once the secret key is found, it creates a new JWT with user-defined parameters.

Main important Libraries

- base64: Used for encoding and decoding JWT components.
- hmac: Used for generating HMAC signatures for verifying and creating JWTs.
- json: Used for handling JWT payloads and headers.
- binascii: Used for hexadecimal and binary data conversion.

PART-A

JwtVerification function:

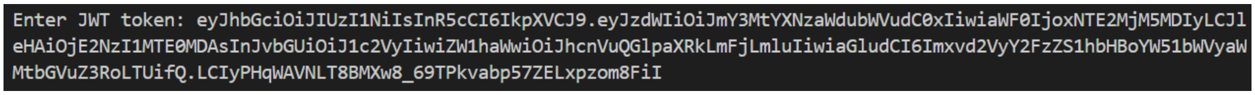
Decodes the JWT header and payload (from Base64).
Extracts the hashing algorithm (alg).
Computes the expected signature:

- Uses HMAC-SHA256 or HMAC-SHA512, depending on alg.
- Converts the computed HMAC from hex to bytes.
- Encodes the result in Base64 URL format.

Compares the computed signature with the provided one.

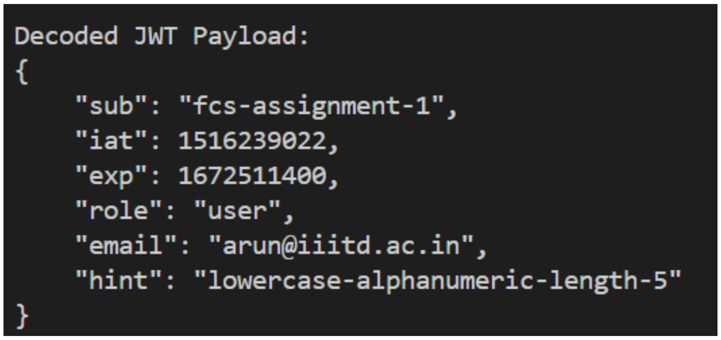
- If they match, the JWT is valid.
- Otherwise, it raises an "Invalid signature" error.

Main Function:
First we are taking a jwt token from the user as input.



```
Enter JWT token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJmY3MtYXNzaWdubWVudC0xIiwiaWF0IjoxNTE2MjM5MDIyLCJl
eHAiOjE2NzI1MTE0MDAsInJvbGUiOiJ1c2VyIiwiZW1haWwiOiJhcnVuQGlpaXRkLmFjLmluIiwiaGludCI6Imxvd2VyY2FzZS1hbHBoYW51bWVyaW
MtbGVuZ3RoLTUifQ.LCIyPHqWAVNLT8BMXw8_69TPkvabp57ZELxpzom8FiI
```

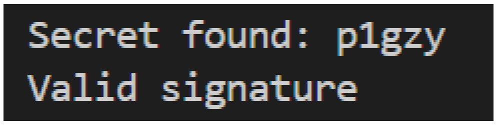
As we know a jwt token contains Header, Payload and Signature so Separating them into Header, Payload and Signature.
We are decoding the payload data to check its information. From this we got to know that our secret key is of length 5 and it is an alpha-numeric value.



```
Decoded JWT Payload:
{
    "sub": "fcs-assignment-1",
    "iat": 1516239022,
    "exp": 1672511400,
    "role": "user",
    "email": "arun@iiitd.ac.in",
    "hint": "lowercase-alphanumeric-length-5"
}
```

After this I applied a brute force method to calculate the secret key and checked all the permutations and stores that in a list.

Secret key found was : p1gzy



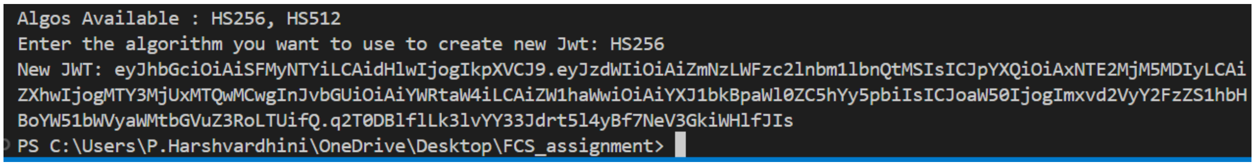
```
Secret found: p1gzy
Valid signature
```

Logic to find secret key:
Iterates through the list of **possible secrets created above**.
Computes the **HMAC-SHA256 signature** for each secret.
Converts it to **Base64 URL format**.
Compares it with the original JWT signature.
If a match is found, it prints the **discovered secret**.
If no matching secret is found, an exception is raised.
If found, it verifies the JWT using the JwtVerification function if it is valid or not.

PART-B

Creating new jwt for role = admin:

The user selects an algorithm (HS256 or HS512).
Uses the discovered secret.
Preparing new jwt components
Header: Specifies the algorithm and type.
Payload: Contains claims such as role, email, hint, and timestamps.
Encodes both in Base64.



```
Algos Available : HS256, HS512
Enter the algorithm you want to use to create new Jwt: HS256
New JWT: eyJhbGciOiAiSFMyNTYiLCAidHlwIjogIkpXVCJ9.eyJzdWIiOiAiZmNzLWFzc2lnbm1lbnQtMSIsICJpYXQiOiAxNTE2MjM5MDIyLCAi
ZXhwIjogMTY3MjUxMTQwMCwgInJvbGUiOiAiYWRtaW4iLCAiZW1haWwiOiAiYXJ1bkBpaWl0ZC5hYy5pbiIsICJoaW50IjogImxvd2VyY2FzZS1hbH
BoYW51bWVyaWMtbGVuZ3RoLTUifQ.q2T0DBlflLk3lvYY33Jdrt5l4yBf7NeV3GkiWHlfJIs
PS C:\Users\P.Harshvardhini\OneDrive\Desktop\FCS_assignment>
```

Computes the HMAC signature using the chosen algorithm.
Converts to Base64 URL format.
Constructs the new JWT.

PART-C

To prevent widespread damage from leaked secrets we can use the following methods:

- Use asymmetric cryptography method (RSA algorithm) this prevents key leakage by using private and public key.
- We can use key rotation which says regularly change the signing secrets to minimize this risk of leakage.
- Setting the short expiration time for JWTs, forcing clients to re-authenticate frequently, minimizing the window of vulnerability if a key is compromised.
- Generating each key per user or session instead of keeping a global key.