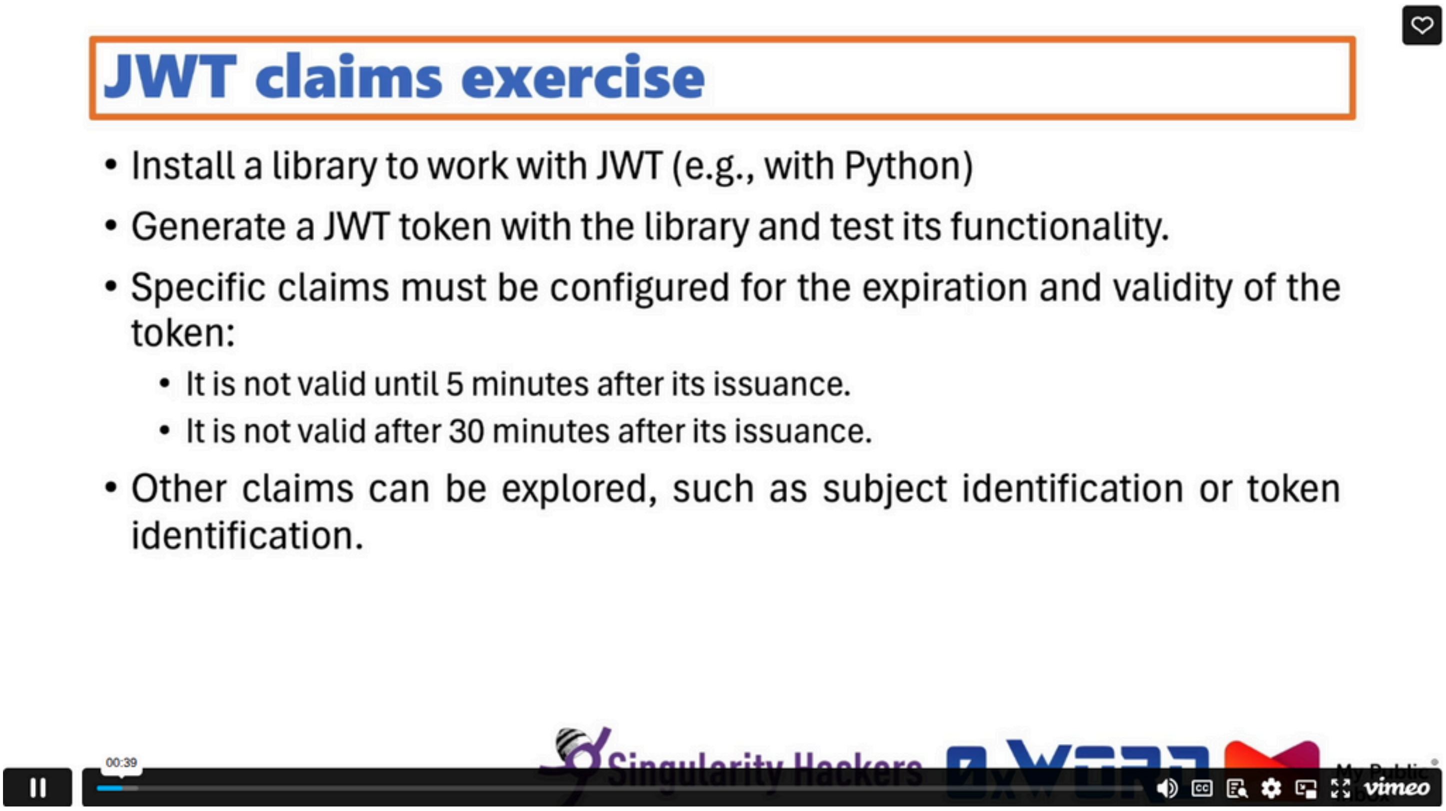
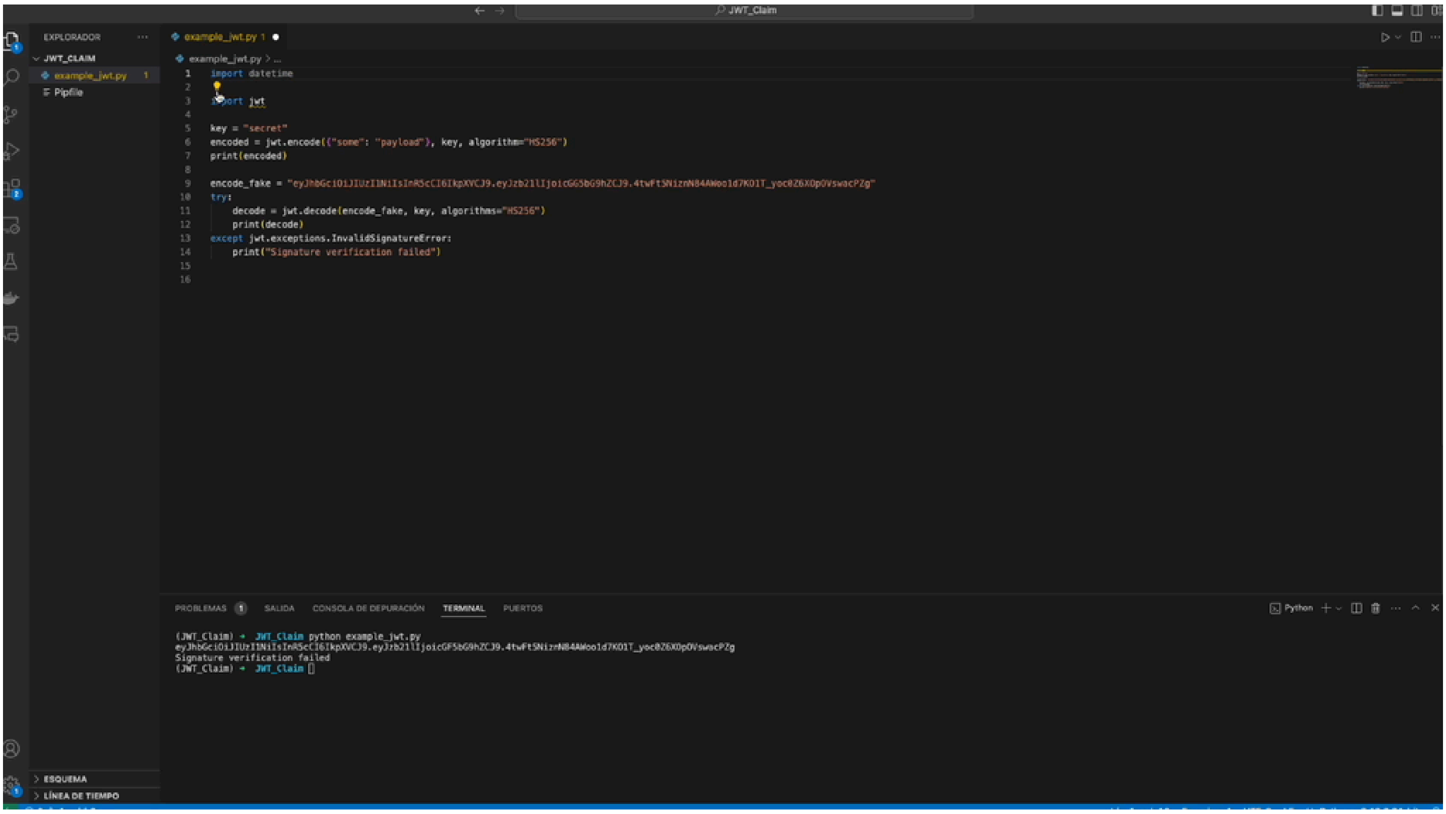


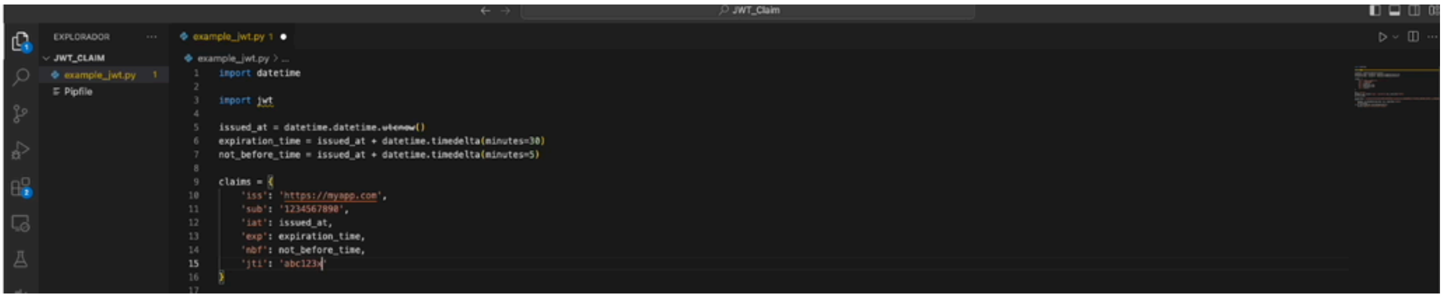
JWT claims exercise

- Install a library to work with JWT (e.g., with Python)
- Generate a JWT token with the library and test its functionality.
- Specific claims must be configured for the expiration and validity of the token:
  - It is not valid until 5 minutes after its issuance.
  - It is not valid after 30 minutes after its issuance.
- Other claims can be explored, such as subject identification or token identification.

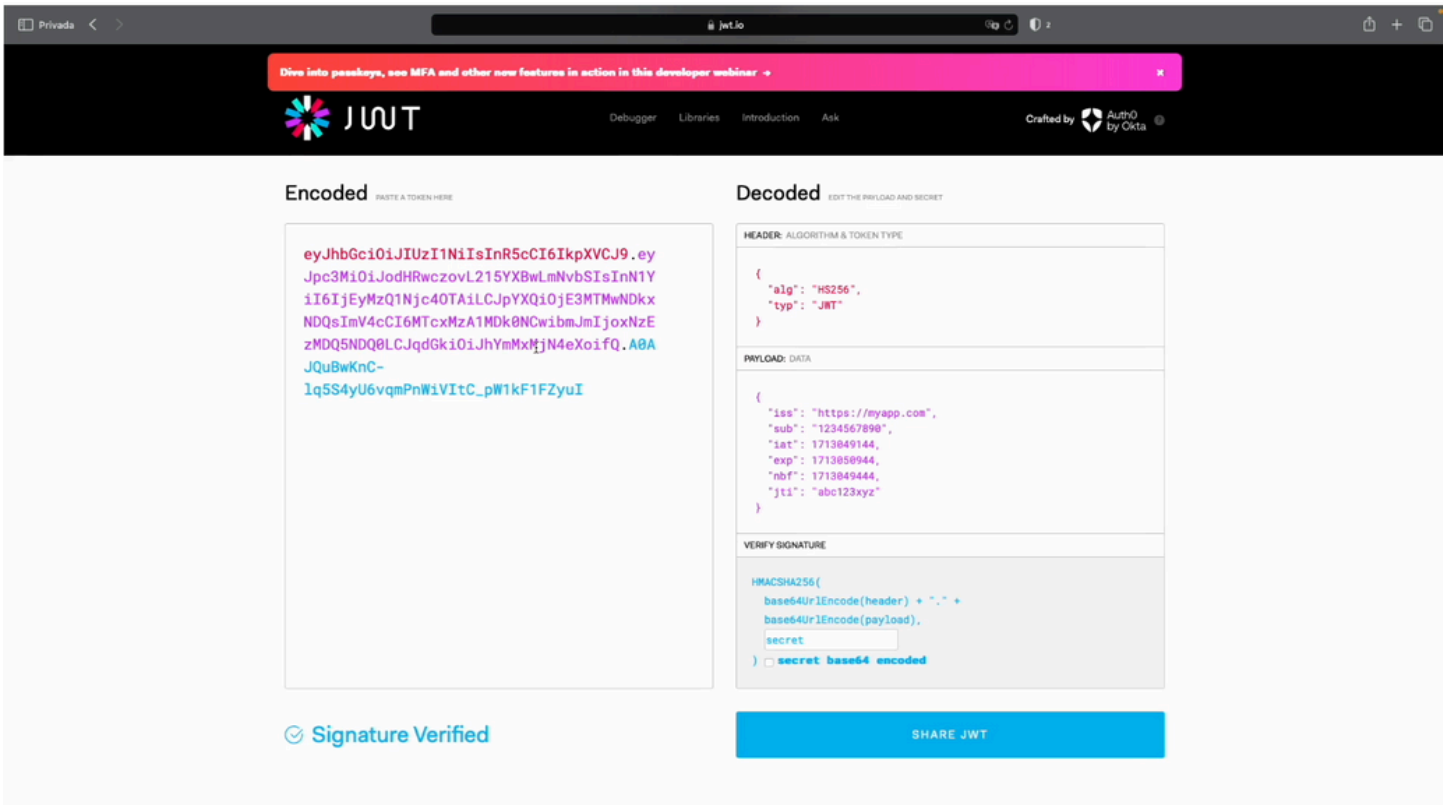


```python
import datetime

import jwt

key = "secret"
encoded = jwt.encode({"some": "payload"}, key, algorithm="HS256")
print(encoded)

encode_fake = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb21lIjoicGG5bG9hZCJ9.4twFt5NiznN84AWoo1d7K01T_yoc0Z6XOpOVswacPZg"
try:
    decode = jwt.decode(encode_fake, key, algorithms="HS256")
    print(decode)
except jwt.exceptions.InvalidSignatureError:
    print("Signature verification failed")
```

```
(JWT_Claim) → JWT_Claim python example_jwt.py
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb21lIjoicGF5bG9hZCJ9.4twFt5NiznN84AWoo1d7K01T_yoc0Z6XOpOVswacPZg
Signature verification failed
(JWT_Claim) → JWT_Claim []
```



```python
import datetime

import jwt

issued_at = datetime.datetime.utcnow()
expiration_time = issued_at + datetime.timedelta(minutes=30)
not_before_time = issued_at + datetime.timedelta(minutes=5)

claims = {
    'iss': 'https://myapp.com',
    'sub': '1234567890',
    'iat': issued_at,
    'exp': expiration_time,
    'nbf': not_before_time,
    'jti': 'abc123x'
}
```


Privada
jwt.io
Dive into passkeys, see MFA and other new features in action in this developer webinar →
JWT
Debugger
Libraries
Introduction
Ask
Crafted by Auth0 by Okta

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
Jpc3MiOiJodHRwczovL215YXBwLmNvbSIsInN1Y
iI6IjEyMzQ1Njc4OTAiLCJpYXQiOjE3MTMwNDkx
NDQsImV4cCI6MTcxMzA1MDk0NCwibmJmIjoxNzE
zMDQ5NDQ0LCJqdGkiOiJhYmMxMjN4eXoifQ.A0A
JQuBwKnC-
lq5S4yU6vqmPnWiVItC_pW1kF1FZyuI
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

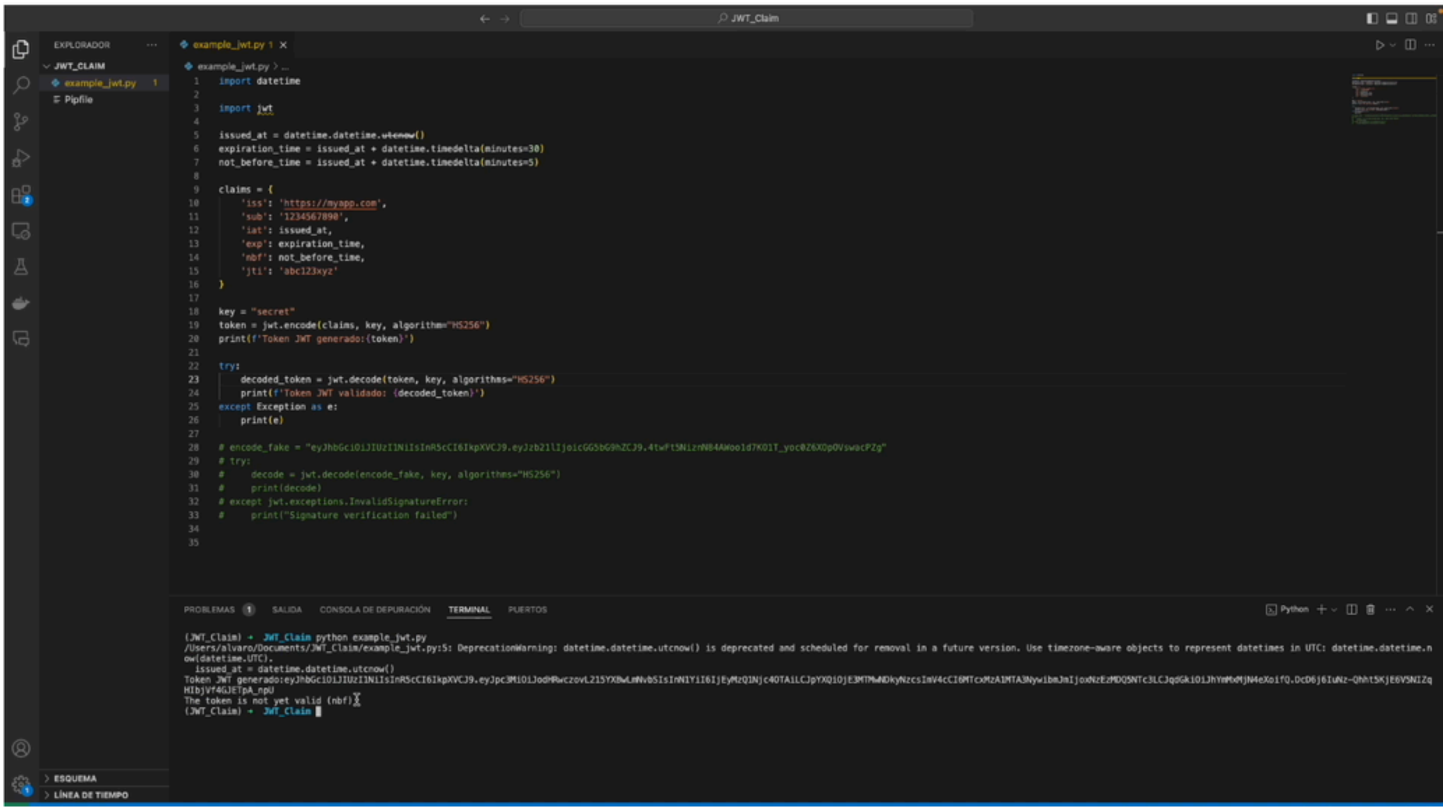
PAYLOAD: DATA

```
{
  "iss": "https://myapp.com",
  "sub": "1234567890",
  "iat": 1713049144,
  "exp": 1713050944,
  "nbf": 1713049444,
  "jti": "abc123xyz"
}
```

VERIFY SIGNATURE

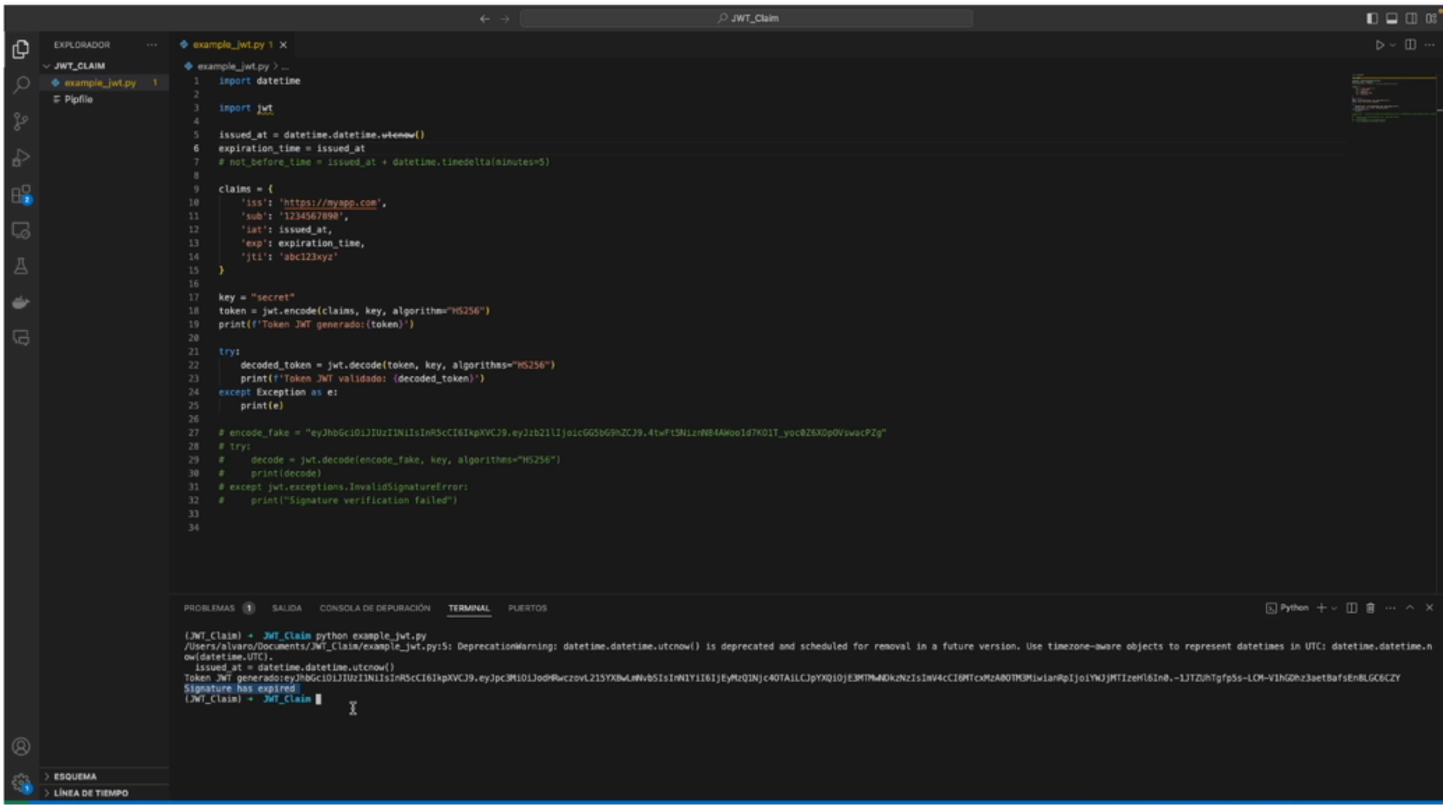
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) □ secret base64 encoded
```

Signature Verified



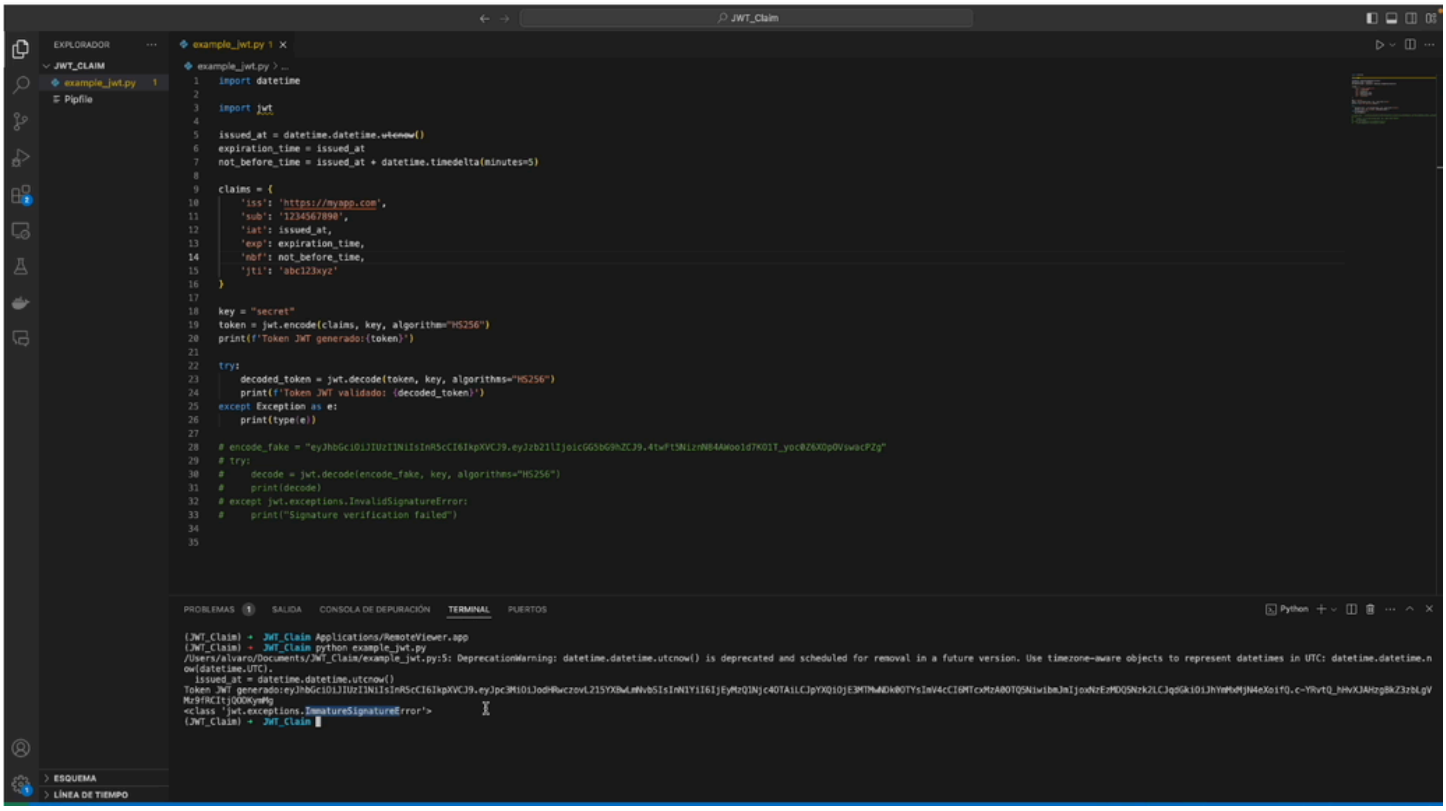
```python
import datetime

import jwt

issued_at = datetime.datetime.utcnow()
expiration_time = issued_at + datetime.timedelta(minutes=30)
not_before_time = issued_at + datetime.timedelta(minutes=5)

claims = {
    'iss': 'https://myapp.com',
    'sub': '1234567890',
    'iat': issued_at,
    'exp': expiration_time,
    'nbf': not_before_time,
    'jti': 'abc123xyz'
}

key = "secret"
token = jwt.encode(claims, key, algorithm="HS256")
print(f'Token JWT generado:{token}')

try:
    decoded_token = jwt.decode(token, key, algorithms="HS256")
    print(f'Token JWT validado: {decoded_token}')
except Exception as e:
    print(e)

# encode_fake = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb21lIjoicGG5bG9hZCJ9.4twFt5NiznN84AWoold7KO1T_yoc0Z6XOpOVswacPZg"
# try:
#     decode = jwt.decode(encode_fake, key, algorithms="HS256")
#     print(decode)
# except jwt.exceptions.InvalidSignatureError:
#     print("Signature verification failed")
```

```
(JWT_Claim) ➜  JWT_Claim python example_jwt.py
/Users/alvaro/Documents/JWT_Claim/example_jwt.py:5: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.n
ow(datetime.UTC).
  issued_at = datetime.datetime.utcnow()
Token JWT generado:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL215YXBwLmNvbSIsInN1YiI6IjEyMzQ1Njc4OTAiLCJpYXQiOjE3MTMwNDkyNzcsImV4cCI6MTcxMzA1MTA3NywibmJmIjoxNzEzMDQ5NTc3LCJqdGkiOiJhYmMxMjN4eXoifQ.DcD6j6IuNz-Qhht5KjE6V5NIZq
HIbjVf4GJETpA_npU
The token is not yet valid (nbf)
(JWT_Claim) ➜  JWT_Claim
```



```python
import datetime

import jwt

issued_at = datetime.datetime.utcnow()
expiration_time = issued_at
# not_before_time = issued_at + datetime.timedelta(minutes=5)

claims = {
    'iss': 'https://myapp.com',
    'sub': '1234567890',
    'iat': issued_at,
    'exp': expiration_time,
    'jti': 'abc123xyz'
}

key = "secret"
token = jwt.encode(claims, key, algorithm="HS256")
print(f'Token JWT generado:{token}')

try:
    decoded_token = jwt.decode(token, key, algorithms="HS256")
    print(f'Token JWT validado: {decoded_token}')
except Exception as e:
    print(e)

# encode_fake = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb21lIjoicGG5bG9hZCJ9.4twFt5NiznN84AWoo1d7KO1T_yoc0Z6XOpOVswacPZg"
# try:
#     decode = jwt.decode(encode_fake, key, algorithms="HS256")
#     print(decode)
# except jwt.exceptions.InvalidSignatureError:
#     print("Signature verification failed")
```

```
(JWT_Claim) ➜  JWT_Claim python example_jwt.py
/Users/alvaro/Documents/JWT_Claim/example_jwt.py:5: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.n
ow(datetime.UTC).
  issued_at = datetime.datetime.utcnow()
Token JWT generado:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL215YXBwLmNvbSIsInN1YiI6IjEyMzQ1Njc4OTAiLCJpYXQiOjE3MTMwNDkzNzIsImV4cCI6MTcxMzA0OTM3MiwianRpIjoiYWJjMTIzeHl6In0.-1JTZUhTgfp5s-LCM-V1hGDhz3aetBafsEnBLGC6CZY
Signature has expired
(JWT_Claim) ➜  JWT_Claim 
```



```python
import datetime

import jwt

issued_at = datetime.datetime.utcnow()
expiration_time = issued_at
not_before_time = issued_at + datetime.timedelta(minutes=5)

claims = {
    'iss': 'https://myapp.com',
    'sub': '1234567890',
    'iat': issued_at,
    'exp': expiration_time,
    'nbf': not_before_time,
    'jti': 'abc123xyz'
}

key = "secret"
token = jwt.encode(claims, key, algorithm="HS256")
print(f'Token JWT generado:{token}')

try:
    decoded_token = jwt.decode(token, key, algorithms="HS256")
    print(f'Token JWT validado: {decoded_token}')
except Exception as e:
    print(type(e))

# encode_fake = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb21lIjoicGG5bG9hZCJ9.4twFt5NiznN84AWoo1d7KO1T_yoc0Z6XOpOVswacPZg"
# try:
#     decode = jwt.decode(encode_fake, key, algorithms="HS256")
#     print(decode)
# except jwt.exceptions.InvalidSignatureError:
#     print("Signature verification failed")
```

```
(JWT_Claim) ➜  JWT_Claim Applications/RemoteViewer.app
(JWT_Claim) ➜  JWT_Claim python example_jwt.py
/Users/alvaro/Documents/JWT_Claim/example_jwt.py:5: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.n
ow(datetime.UTC).
  issued_at = datetime.datetime.utcnow()
Token JWT generado:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL215YXBwLmNvbSIsInN1YiI6IjEyMzQ1Njc4OTAiLCJpYXQiOjE3MTMwNDk0OTYsImV4cCI6MTcxMzA0OTQ5NiwibmJmIjoxNzEzMDQ5Nzk2LCJqdGkiOiJhYmMxMjN4eXoifQ.c-YRvtQ_hHvXJAHzg8kZ3zbLgV
Mz9fRCItjQ0DKymMg
<class 'jwt.exceptions.ImmatureSignatureError'>
(JWT_Claim) ➜  JWT_Claim 
```

Conclusion

- Configuring specific claims in JWT tokens is fundamental for controlling the issuer, audience, expiration, or validity, among others.
- Using specialized libraries, such as PyJWT in Python, facilitates the generation and validation of JWT tokens, allowing us to work more efficiently and securely.
- Proper configuration of the exp (expiration time) and nbf (not before) claims allows us to control the token's lifespan, ensuring it is only valid for the necessary time and preventing misuse after expiration.
- Other claims that may be useful in different scenarios include subject identification (sub) or token identification (jti), which allow us to add additional information to the token and enhance its security.
- Configuring JSON Web Tokens with specific claims has practical applications in the development of web applications and APIs, where authentication and authorization are critical aspects of security.