import pytest

import disruptive import disruptive.errors as dterrors import tests.api_responses as dtapiresponses from disruptive.authentication import Auth

class TestAuth():

```
def test_repr(self, request_mock):
    # Update the response json with a mock token response.
    res = dtapiresponses.auth_token_fresh
    request_mock.json = res

    # Fetch a role.
    x = disruptive.Auth.serviceaccount('key_id', 'secret', 'email')

    # Evaluate __repr__ function and compare copy.
    eval(repr(x))

def test_serviceaccount_auth(self, request_mock):
    # Update the response json with a mock token response.
    res = dtapiresponses.auth_token_fresh
    request_mock.json = res

    # Call the two classmethod constructors.
    auth = disruptive.Auth.serviceaccount('key_id', 'secret', 'email')

    # Assert token post request not sent at construction.
    request_mock.assert_request_count(0)

    # Assert instance of Auth class.
    assert isinstance(auth, Auth)

def test_token_refresh(self, request_mock):
    # Update the response json with an expired token response.
    res = dtapiresponses.auth_token_expired
    request_mock.json = res

    # Create an authentication object.
    auth = disruptive.Auth.serviceaccount('key_id', 'secret', 'email')

    # Verify expired token.
    assert auth._has_expired()

    # Update the response json with a fresh token response.
    res = dtapiresponses.auth_token_fresh
    request_mock.json = res
```

```
        # Call the get_token method to force a refresh.
        auth.get_token()

        # Verify non-expired token.
        assert not auth._has_expired()

    def test_raise_none_credential(self):
        # Verify InvalidTypeError raised at None input credential.
        with pytest.raises(TypeError):
            disruptive.Auth.serviceaccount(None, 'secret', 'email')
        with pytest.raises(TypeError):
            disruptive.Auth.serviceaccount('key_id', None, 'email')
        with pytest.raises(TypeError):
            disruptive.Auth.serviceaccount('key_id', 'secret', None)

    def test_raise_empty_string_credential(self):
        # Verify EmptyStringError raised at missing input credential.
        with pytest.raises(dterrors.EmptyStringError):
            disruptive.Auth.serviceaccount('', 'secret', 'email')
        with pytest.raises(dterrors.EmptyStringError):
            disruptive.Auth.serviceaccount('key_id', '', 'email')
        with pytest.raises(dterrors.EmptyStringError):
            disruptive.Auth.serviceaccount('key_id', 'secret', '')
```