Offline verification is useful in the following use cases:

1. When customers are not expected to always be online (e.g. when it's okay for them to be offline for a few days until they need to connect again).
2. Permanently offline (e.g. behind a corporate firewall).

Background

Before we begin, let's review how standard key verification works: https://help.cryptolens.io/examples/key-verification

```
result = Key.activate(token=auth,\
    rsa_pub_key=RSAPubKey,\
    product_id=3349, \
    key="ICVLD-VVSZR-ZTICT-YKGXL",\
    machine_code=Helpers.GetMachineCode(v=2))

if result[0] == None or not Helpers.IsOnRightMachine(result[0], v=2):
    # an error occurred or the key is invalid or it cannot be activated
    # (eg. the limit of activated devices was achieved)
    print("The license does not work: {0}".format(result[1]))
else:
    # everything went fine if we are here!
    print("The license is valid!")
```

When Key.Activate is called, it calls Cryptolens API, which in turn returns a **signed JSON object** placed in the **result** variable. This allows you to proceed as normal and check features, expiry date, etc.

This JSON object can be saved to disk, allowing us to read it again when the customer lacks internet access.

To save to disk in Python (similar principle in other languages):

```
# res is obtained from the code above
if result[0] != None:
    # saving license file to disk
    with open('licensefile.skm', 'w') as f:
        f.write(result[0].save_as_string())
```

Case 1: Handling scenarios when the customer is offline temporarily

The idea is as follows: as the first step, call Key.Activate and check if it returns a successful result. If it does, proceed as you normally would if you have internet access.

If Key.Activate is unable to reach the server, attempt to read the license file from file.

In the code below, we read the license file from licensefile.skm instead of calling Key.Activate.

```python
# read license file from file
with open('licensefile.skm', 'r') as f:
    license_key = LicenseKey.load_from_string(pubKey, f.read())

    if license_key == None or not Helpers.IsOnRightMachine(license_key, v=2):
        print("NOTE: This license file does not belong to this machine.")
    else:
        print("Feature 1: " + str(license_key.f1))
        print("License expires: " + str(license_key.expires))
```

When loading a license from a file, you have flexibility to limit the number of days your customers are allowed to be offline before they need a new license file (obtained by calling Key.Activate, as an example), which can be done in the LoadFromFile method. Note that it works the same in other languages, not just Python. In the code below, the customer will need to connect to the internet 30 days after the last successful activation.

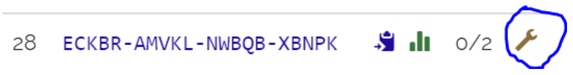
```python
# read license file from file
with open('licensefile.skm', 'r') as f:
    license_key = LicenseKey.load_from_string(pubKey, f.read(), 30)

    if license_key == None or not Helpers.IsOnRightMachine(license_key, v=2):
        print("NOTE: This license file does not belong to this machine.")
    else:
        print("Feature 1: " + str(license_key.f1))
        print("License expires: " + str(license_key.expires))
```

More examples for Python: https://github.com/Cryptolens/cryptolens-python?tab=readme-ov-file#offline-activation-savingloading-licenses

Case 2: No internet access

With no internet access, the call to Key.Actvate will not work (unless a local license server is used, which we will talk about later) and you will have to load it from file. There at least three methods to generate this file:

1. Activation forms hosted by Cryptolens: https://app.cryptolens.io/ActivationForms
2. Calling Key.Activate on you end and sending the file to your customer (possibly as a part of an automated workflow)
3. Obtaining it manually in the dashboard:



28 ECKBR-AMVKL-NWBQB-XBNPK 0/2

If you anticipate that your customer will have many employees requiring your software to run offline, or if you would like to support floating licensing offline, please check out the license server: https://github.com/Cryptolens/license-server.